












Thank you for purchasing the Advance Starter Kit!

KIT CONTENTS

The following are the contents of the kit. Pictures are for reference only. Actual items may vary slightly (e.g. color, size, brand, form factor/shape, etc). Certain component characteristic may also vary (e.g. resistor values, carrier boards, hardware versions):


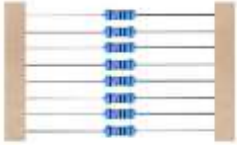



ITEM NAME	QUANTITY	PICTURES
Arduino Uno microcontroller board Option A: Arduino Uno DIP version Option B: Arduino Uno SMD version This is the heart of your kit. This board contains the microcontroller and its supporting circuits to be able to perform the exercises designed for this kit.	1PC	 <p>Arduino Uno DIP version</p> <p>Arduino Uno SMD version</p>
USB CABLE Use this connect your Arduino Uno to your computer for Uploading and for viewing serial port data from the Serial Monitor.	1PC	
LCD 16X2 WITH I2C BACKPACK This provides the Arduino the capability to display 2 lines of 16 alphanumeric characters (total of 32 characters).	1 SET	
MFRC522 RFID MODULE This is a 13.56MHz RFID reader module. Its most basic function is to read the unique identification number from compatible RFID cards and tags	1PC	

<p>13.56MHz RFID CARD</p> <p>A sample RFID card for the MFRC522 reader</p>	<p>1PC</p>	
<p>13.56MHz RFID KEYCHAIN TAG</p> <p>A sample RFID tag for the MFRC522 reader</p>	<p>1PC</p>	
<p>ANALOG JOYSTICK MODULE</p> <p>This module consists of a 2-axis potentiometer enclosed as one. Usage would be the same as if there were 2 potentiometers. The stick also serves as a pushbutton when pressed</p>	<p>1PC</p>	
<p>4X4 MATRIX KEYBOARD</p> <p>A basic keypad module for data entry or controls</p>	<p>1PC</p>	
<p>RTC MODULE</p> <p>A DS1302 based real time clock module that keeps track of the date and time even after a power off. A CR2032 coin cell battery is required.</p>	<p>1PC</p>	
<p>RAIN/WATER LEVEL SENSOR</p> <p>Used for detecting water level or water presence. A simple logic change at its output pin occurs on water detection</p>	<p>1PC</p>	
<p>DHT11 HUMIDITY SENSOR</p>	<p>1PC</p>	

RGB LED MODULE	1PC	
28BYJ-48 STEPPER MOTOR DRIVER MODULE	1PC	
28BYJ-48 STEPPER MOTOR	1PC	
1 CHANNEL RELAY MODULE, 5V coil, 10A contacts	1PC	
MB-102 BREADBOARD, 830-points	1PC	
JUMPER WIRES	30 PCS	
MALE-FEMALE CONNECTOR	10 PCS	

SOUND SENSOR MODULE	1PC	
INFRARED REMOTE CONTROLLER	1PC	
10Kohm POTENTIOMETER	1PC	
SEVEN SEGMENT MODULE	1PC	
4 DIGIT SEVEN SEGMENT MODULE	1PC	
8x8 LED MATRIX MODULE	1PC	

9G SERVO MOTOR	1PC	
BUZZER	1PC	
TILT SWITCH	1PC	
LIGHT DEPENDENT RESISTOR (LDR)	2PCS	
TACTILE PUSH BUTTONS WITH CAPS	5PCS	
9V BATTERY CLIP WITH DC PLUG	1PC	

LEDs	15PCS	
RESISTOR	30PCS	
38KHz IR RECEIVER	1PC	
74HC595 IC	1PC	
LM35DZ	1PC	

INSTALL THE SERIAL PORT DRIVER

Skip this portion if you have the DIP version of the Arduino Uno. This only applies to board with CH340G USB chips.

The Arduino Uno SMD version uses the popular CH340G chip as its USB-UART interface controller. This is the most reliable, cost-effective and proven solution that allows the ATmega328P chip on the board to communicate with the computer. Arduino Uno DIP version users need not perform this step. Follow these steps to install the driver:

- 1) Download the driver installation files from the following links:
Windows Users: <http://layadcircuits.com/ds/SalengUno/drivers/CH341SER.ZIP>
Mac OS Users: http://layadcircuits.com/ds/SalengUno/drivers/CH341SER_MAC.ZIP
- 2) Uncompress the downloaded file
- 3) Go to the CH341SER folder
- 4) Run setup.exe
- 5) On the window that pops up, click on the Install button
- 6) Wait until installation is complete. Note that if you already previously installed the driver, the installation may report a “failed” installation. If there is a need to reinstall, please uninstall first before attempting a reinstall.
- 7) Restart the computer
- 8) After restart, insert the Arduino Uno into a USB port of the computer. The computer should now recognize the hardware.

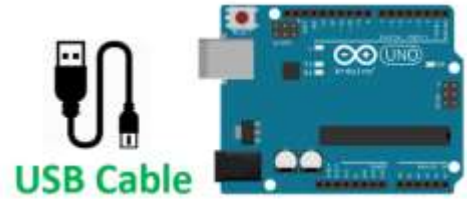
INSTALL THE ARDUINO IDE

Before you can start using the kit, you will need to install the “Arduino IDE”. The Arduino Integrated Development Environment is a package that includes all the necessary software tools such as a compiler, linker, editor, programmer, a serial monitor (terminal) and others. Download the latest Arduino IDE from the Arduino website: <https://www.arduino.cc/en/Main/Software> . The installer (not the non-admin zip file) is recommended.

Once downloaded, run the installer and follow the onscreen instructions.

BASIC SETUP

There are two main parts of the basic Arduino development setup. You will need a computer running the Arduino IDE with the serial driver installed, and the Arduino with the necessary hardware attached to it. The Arduino and the computer is then connected via the USB cable provided in the kit.



USB Cable

Arduino Board

**Sketch built from
the Arduino IDE**

Below are some pointers:

- You will build your program/code or “sketch” in the Arduino IDE software. These are the “instructions” that you want the Arduino to perform.
- After writing your code, you will “upload” or burn the program into the Arduino. Once programmed, the Arduino will run this code until you upload another one.
- Note that the Arduino is powered from the USB cable when connected to the computer. Although this will work for majority of the experiments you can do with the kit, there may be some applications where you might exceed the 500mA limit of the USB port. In such a case, use a 7-9V (recommended) or up to 12V power supply with a higher current rating. A 9V battery or two Li Ion batteries in series will also suffice.

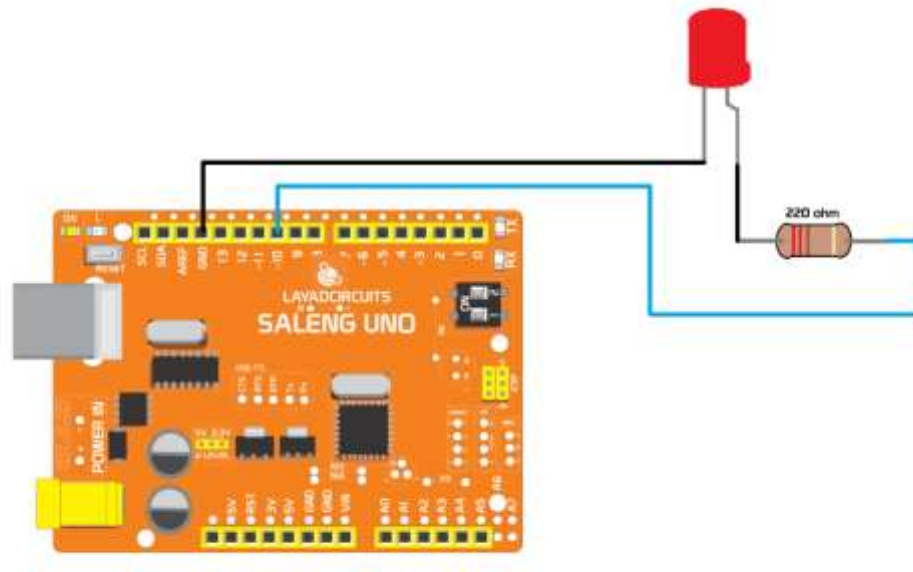
WHAT'S NEXT?

At this point, you are all set. True to the open-source concept of the Arduino, there are numerous basic experiments/tutorials you may perform with this kit from the internet. You may also [follow some of the experiments for this kit](#) that may have been provided to you. You may also go through the examples under File>Examples in the Arduino IDE, the example sketches would have enough information on how to wire the hardware. There are also a lot of resources you can find from the official Arduino website: www.arduino.cc. The website has a “Resources” section and a “Community” or Forums where you may get specific information.

EXERCISES

BLINKING LED

This exercise turns on and off an LED that is wired externally. If you change the LED to pin 13, the LED on the board of Saleng/Arduino will blink. You can also change the Delay of the LED on the code.



CODE:

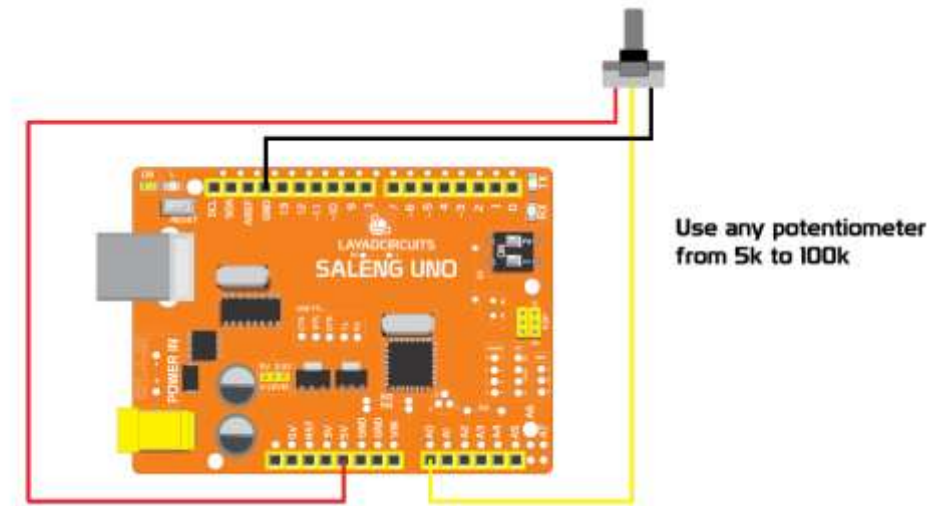
```
int LED = 10; // LED pin

void setup()
{
  pinMode(LED, OUTPUT); // set LED pin as an Output
}

void loop()
{
  digitalWrite(LED, HIGH); // turn on the LED
  delay(500); // wait for 500ms
  digitalWrite(LED, LOW); // turn on the LED
  delay(500); // wait for 500ms
}
```

DIGITAL VOLTMETER

This exercise measures the voltage applied to pin A0 (apply 0-5V) and displays it on the Serial monitor. The potentiometer varies this voltage between 0V and 5V. Open the serial monitor and set the baud rate to 9600. The LED built into the board will blink every 100ms.



CODE:

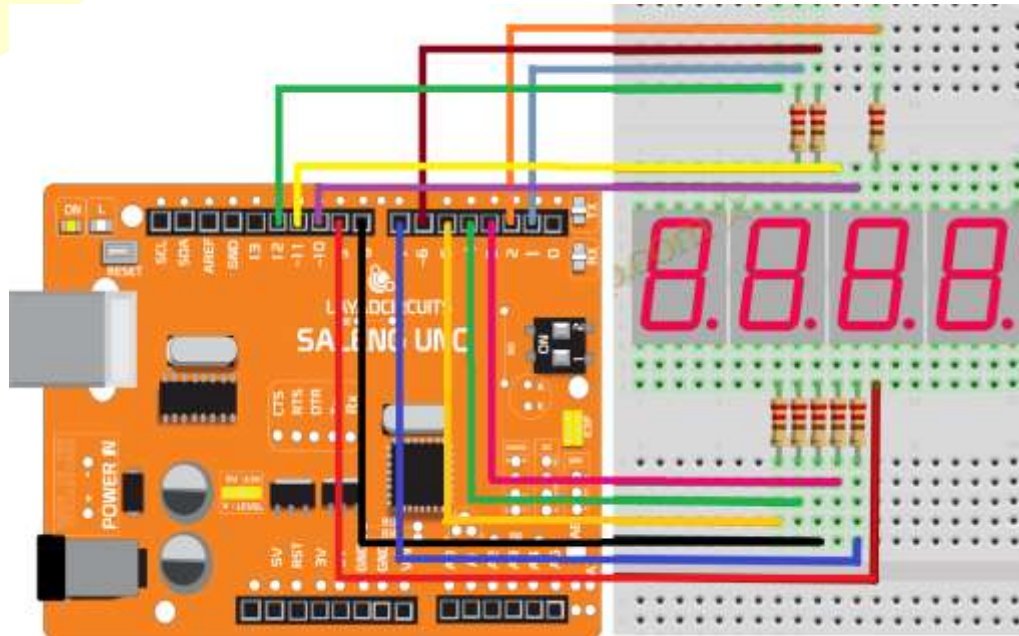
```
int potpin=0; // Use pin A0 as the voltage input. limit this to 5V
int ledpin=13; //
int val=0; //
int v;
void setup()
{
  pinMode(ledpin,OUTPUT); //
  Serial.begin(9600); //
}
void loop()
{
  // blink the LED within the board every 100ms
  digitalWrite(ledpin,HIGH);
  delay(50);
  digitalWrite(ledpin,LOW);
  delay(50);

  val=analogRead(potpin); // read the raw ADC reading
  v=map(val,0,1023,0,500); // convert the readings in millivolts x 10

  Serial.println((float)v/100.00); // print on screen
}
```

FOUR DIGIT 7 SEGMENT

This exercise uses a 4 – digit 7 – segment LED that can display numbers from 0 to 9 using Arduino. Seven segment displays consist of 7 LEDs, called segments, arranged in the shape of an “8”. Most 7-segment displays actually have 8 segments, with a dot on the right side of the digit that serves as a decimal point. The Numbers that will be displayed is based on the code.



CODE:

```
int a = 1;
int b = 2;
int c = 3;
int d = 4;
int e = 5;
int f = 6;
int g = 7;
int dp = 8;
int d4 = 9;
int d3 = 10;
int d2 = 11;
int d1 = 12;
long n = 1230;
int x = 100;
int del = 55;

void setup()
{
  pinMode(d1, OUTPUT);
  pinMode(d2, OUTPUT);
  pinMode(d3, OUTPUT);
  pinMode(d4, OUTPUT);
}
```

```

pinMode(a, OUTPUT);
pinMode(b, OUTPUT);
pinMode(c, OUTPUT);
pinMode(d, OUTPUT);
pinMode(e, OUTPUT);
pinMode(f, OUTPUT);
pinMode(g, OUTPUT);
pinMode(dp, OUTPUT);
}
/////////////////////////////////////////////////////////////////
void loop()
{
  Display(1, 1);
  Display(2, 2);
  Display(3, 3);
  Display(4, 4);
}
/////////////////////////////////////////////////////////////////
void WeiXuan(unsigned char n)//
{
  switch(n)
  {
case 1:
  digitalWrite(d1,LOW);
  digitalWrite(d2, HIGH);
  digitalWrite(d3, HIGH);
  digitalWrite(d4, HIGH);
  break;
case 2:
  digitalWrite(d1, HIGH);
  digitalWrite(d2, LOW);
  digitalWrite(d3, HIGH);
  digitalWrite(d4, HIGH);
  break;
case 3:
  digitalWrite(d1,HIGH);
  digitalWrite(d2, HIGH);
  digitalWrite(d3, LOW);
  digitalWrite(d4, HIGH);
  break;
case 4:
  digitalWrite(d1, HIGH);
  digitalWrite(d2, HIGH);
  digitalWrite(d3, HIGH);
  digitalWrite(d4, LOW);
  break;
  default :
    digitalWrite(d1, HIGH);
    digitalWrite(d2, HIGH);
    digitalWrite(d3, HIGH);
    digitalWrite(d4, HIGH);
    break;
}
}

```

```
}  
void Num_0 ()  
{  
    digitalWrite (a, HIGH);  
    digitalWrite (b, HIGH);  
    digitalWrite (c, HIGH);  
    digitalWrite (d, HIGH);  
    digitalWrite (e, HIGH);  
    digitalWrite (f, HIGH);  
    digitalWrite (g, LOW);  
    digitalWrite (dp, LOW);  
}  
void Num_1 ()  
{  
    digitalWrite (a, LOW);  
    digitalWrite (b, HIGH);  
    digitalWrite (c, HIGH);  
    digitalWrite (d, LOW);  
    digitalWrite (e, LOW);  
    digitalWrite (f, LOW);  
    digitalWrite (g, LOW);  
    digitalWrite (dp, LOW);  
}  
void Num_2 ()  
{  
    digitalWrite (a, HIGH);  
    digitalWrite (b, HIGH);  
    digitalWrite (c, LOW);  
    digitalWrite (d, HIGH);  
    digitalWrite (e, HIGH);  
    digitalWrite (f, LOW);  
    digitalWrite (g, HIGH);  
    digitalWrite (dp, LOW);  
}  
void Num_3 ()  
{  
    digitalWrite (a, HIGH);  
    digitalWrite (b, HIGH);  
    digitalWrite (c, HIGH);  
    digitalWrite (d, HIGH);  
    digitalWrite (e, LOW);  
    digitalWrite (f, LOW);  
    digitalWrite (g, HIGH);  
    digitalWrite (dp, LOW);  
}  
void Num_4 ()  
{  
    digitalWrite (a, LOW);  
    digitalWrite (b, HIGH);  
    digitalWrite (c, HIGH);  
    digitalWrite (d, LOW);  
    digitalWrite (e, LOW);  
    digitalWrite (f, HIGH);  
    digitalWrite (g, HIGH);  
}
```

```
    digitalWrite(dp, LOW);
}
void Num_5()
{
    digitalWrite(a, HIGH);
    digitalWrite(b, LOW);
    digitalWrite(c, HIGH);
    digitalWrite(d, HIGH);
    digitalWrite(e, LOW);
    digitalWrite(f, HIGH);
    digitalWrite(g, HIGH);
    digitalWrite(dp, LOW);
}
void Num_6()
{
    digitalWrite(a, HIGH);
    digitalWrite(b, LOW);
    digitalWrite(c, HIGH);
    digitalWrite(d, HIGH);
    digitalWrite(e, HIGH);
    digitalWrite(f, HIGH);
    digitalWrite(g, HIGH);
    digitalWrite(dp, LOW);
}
void Num_7()
{
    digitalWrite(a, HIGH);
    digitalWrite(b, HIGH);
    digitalWrite(c, HIGH);
    digitalWrite(d, LOW);
    digitalWrite(e, LOW);
    digitalWrite(f, LOW);
    digitalWrite(g, LOW);
    digitalWrite(dp, LOW);
}
void Num_8()
{
    digitalWrite(a, HIGH);
    digitalWrite(b, HIGH);
    digitalWrite(c, HIGH);
    digitalWrite(d, HIGH);
    digitalWrite(e, HIGH);
    digitalWrite(f, HIGH);
    digitalWrite(g, HIGH);
    digitalWrite(dp, LOW);
}
void Num_9()
{
    digitalWrite(a, HIGH);
    digitalWrite(b, HIGH);
    digitalWrite(c, HIGH);
    digitalWrite(d, HIGH);
    digitalWrite(e, LOW);
    digitalWrite(f, HIGH);
```

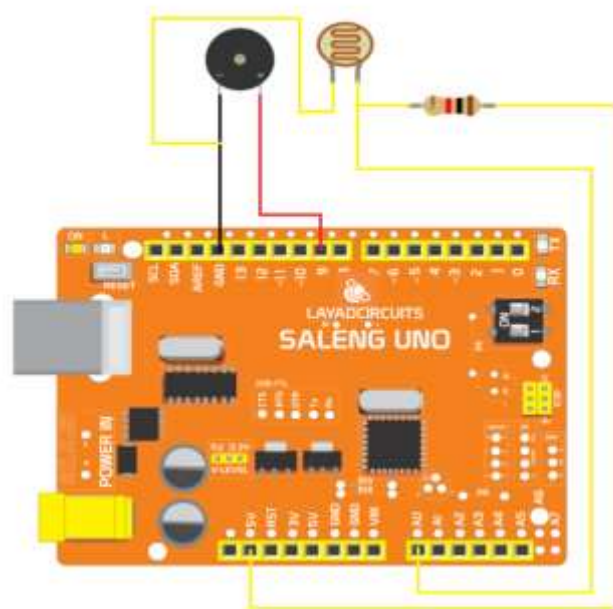
```

    digitalWrite(g, HIGH);
    digitalWrite(dp, LOW);
}
void Clear() //æ,...å±□
{
    digitalWrite(a, LOW);
    digitalWrite(b, LOW);
    digitalWrite(c, LOW);
    digitalWrite(d, LOW);
    digitalWrite(e, LOW);
    digitalWrite(f, LOW);
    digitalWrite(g, LOW);
    digitalWrite(dp, LOW);
}
void pickNumber(unsigned char n)
{
    switch(n)
    {
        case 0:Num_0();
        break;
        case 1:Num_1();
        break;
        case 2:Num_2();
        break;
        case 3:Num_3();
        break;
        case 4:Num_4();
        break;
        case 5:Num_5();
        break;
        case 6:Num_6();
        break;
        case 7:Num_7();
        break;
        case 8:Num_8();
        break;
        case 9:Num_9();
        break;
        default:Clear();
        break;
    }
}
void Display(unsigned char x, unsigned char Number)
{
    WeiXuan(x);
    pickNumber(Number);
    delay(1);
    Clear();
}

```

LIGHT ACTIVATED BUZZER

This exercise is a simple demonstration of an alarm that uses a buzzer and an LDR sensor. The buzzer will sound/alarm when there is an adequate light in the ambient environment. As long as there is an ambient light that is sensed by the LDR sensor, the buzzer will automatically turn on and will remain on as long as the light remains. The LDR has a series resistance with a value that will depend on the particular LDR. This value may be from 220 up to 100K. Change it accordingly. The LDR value may be viewed from Tools>Serial Monitor with 9600 set at the lower right drop down menu



CODE:

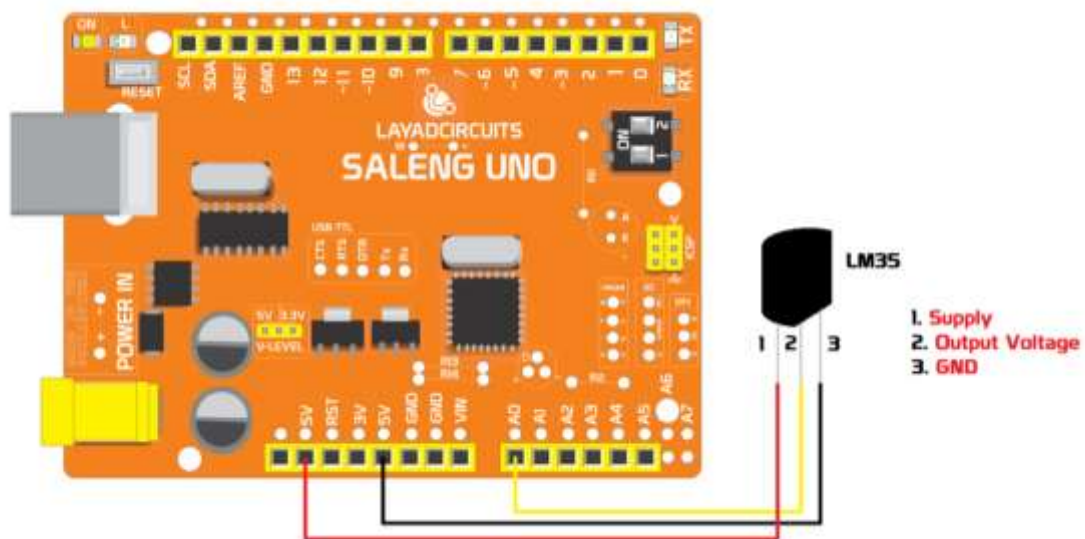
```
int buzzer = 9;
int LDR = 0;
int val;
void setup()
{
  Serial.begin(9600);
  pinMode(buzzer, OUTPUT);
}
void voice_out(int del)
{
  delay(del);
  digitalWrite(buzzer, HIGH);
  delay(del);
  digitalWrite(buzzer, LOW);
}
void loop()
```



```
{
  val=analogRead(LDR);
  Serial.println(val);
  if(val < 700 ) // change this light threshold according to your hardware. this is
  a value from 0 to 1023
  {
    voice_out(val);
  }
  delay(100);
}
```

LM35 TEMPERATURE SENSOR

This exercise uses the LM35 analog temperature sensor to display the temperature on the Serial Monitor. The output voltage varies, based on the temperature around it. Set the serial monitor (Tools>Serial Monitor) baud rate at the lower right drop down menu to 9600.



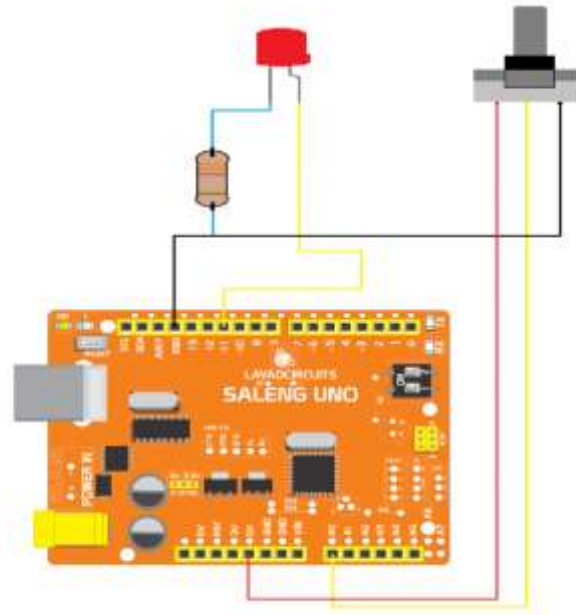
CODE:

```
int potPin = 0;
void setup()
{
  Serial.begin(9600);
}
void loop()
{
  int val;
  int dat;
  val=analogRead(0);
  dat=(125*val)>>8;
  Serial.print("Tep:");
  Serial.print(dat);
}
```

```
Serial.println("C");
delay(500);
}
```

PWM LED BRIGHTNESS

This exercise lets you control the brightness of the LED using the potentiometer by turning its knob.



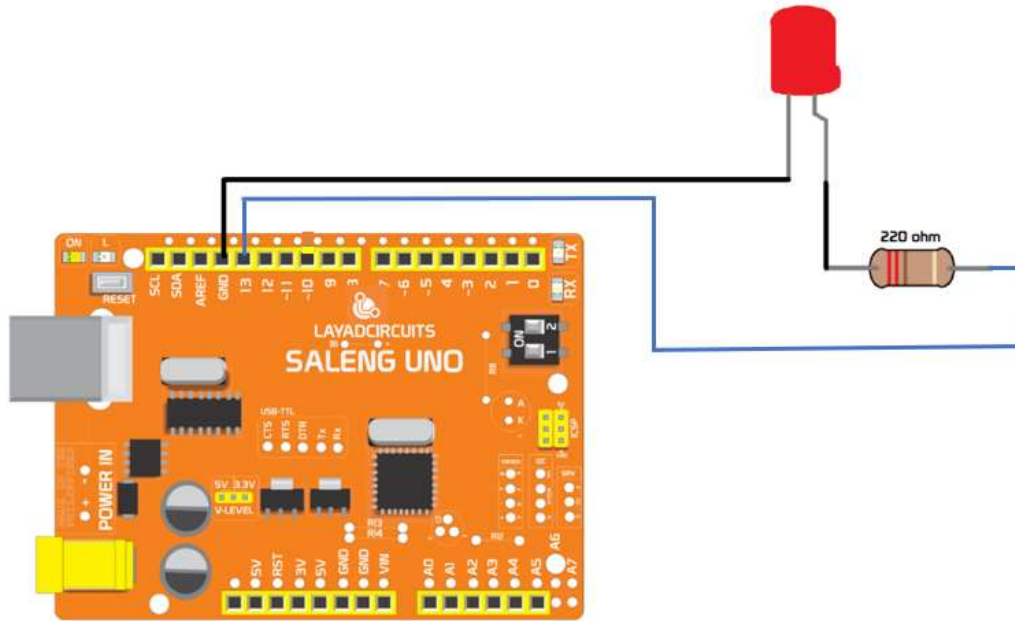
CODE:

```
int potpin=0;
int ledpin=11;
int val=0;

void setup()
{
  pinMode(ledpin,OUTPUT);
  Serial.begin(9600);
}
void loop()
{
  val=analogRead(potpin);
  Serial.println(val);
  analogWrite(ledpin,val/4);
  delay(10);
}
```

SERIAL MONITOR LED CONTROL

Upload the code and open the serial monitor from tools>serial monitor. You may or may not connect external LED and resistor as pin 13 has a built-in LED connected to it labelled "L" on the board. Set the baud rate of the serial monitor to 9600 at the drop-down menu at the lower right corner. Now type 1 in the send field and click on the send button. This will turn on the LED. Sending a 0 turns the LED off. The status is also printed on the serial monitor.



CODE:

```
int val;
int ledpin=13;
void setup()
{
  Serial.begin(9600);
  pinMode(ledpin,OUTPUT);
  Serial.println("Hello World!");
}
void loop()
{
  if(Serial.available()) // wait until we receive a character from the computer
  {
    val=Serial.read(); // read the character
    if(val=='1') // a 1 was received, therefore we turn on the LED
    {
      digitalWrite(ledpin,HIGH);
      Serial.println("LED is ON");
    }
    else if(val=='0') // a 0 was received, therefore we turn off the LED
    {
```

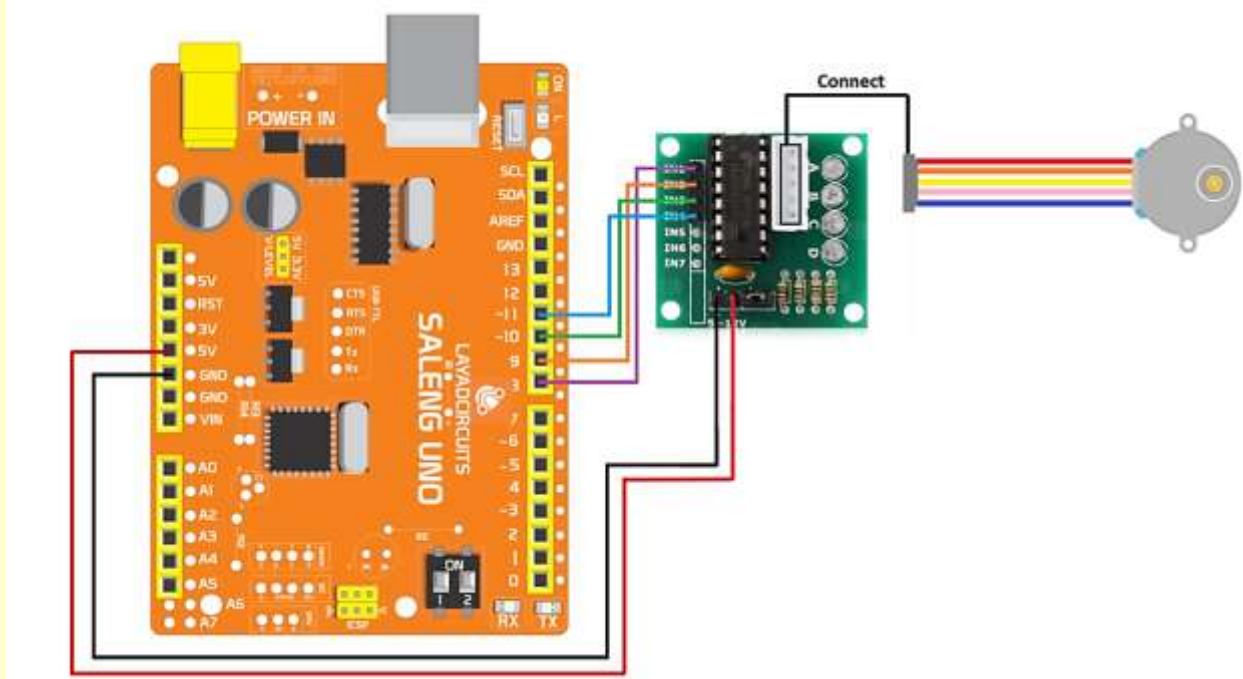
```

digitalWrite(ledpin,LOW);
Serial.println("LED is OFF");
}
}
}

```

STEPPER MOTOR

This control stepper motor with ULN2003 driver that slowly rotate continuously with a set speed parameter of 90



CODE:

```

#include <Stepper.h>

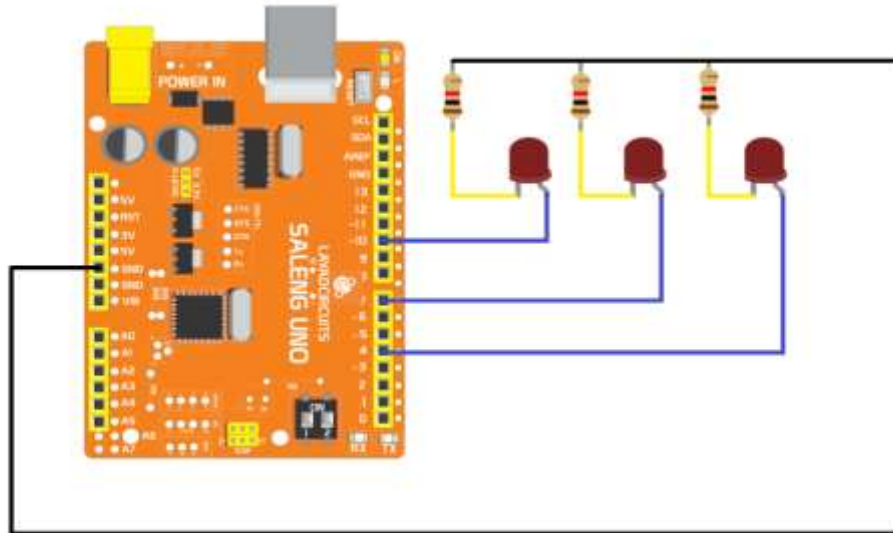
#define STEPS 100
Stepper stepper(STEPS, 8, 9, 10, 11);
int previous = 0;
void setup()
{
  stepper.setSpeed(90);
}
void loop()
{
  int val = analogRead(0);

```

```
stepper.step(val - previous);
previous = val;
}
```

TRAFFIC LIGHTS

This is a simulation of a traffic light. It is simply a sequence of LED's turning on and off.



CODE:

```
const int redled =10;
const int yellowled =7;
const int greenled =4;

void setup()
{
  pinMode(redled, OUTPUT);
  pinMode(yellowled, OUTPUT);
  pinMode(greenled, OUTPUT);
}
void loop()
{
  //turn on the green LED for 5 seconds
  digitalWrite(greenled, HIGH);
  delay(5000);
  digitalWrite(greenled, LOW);

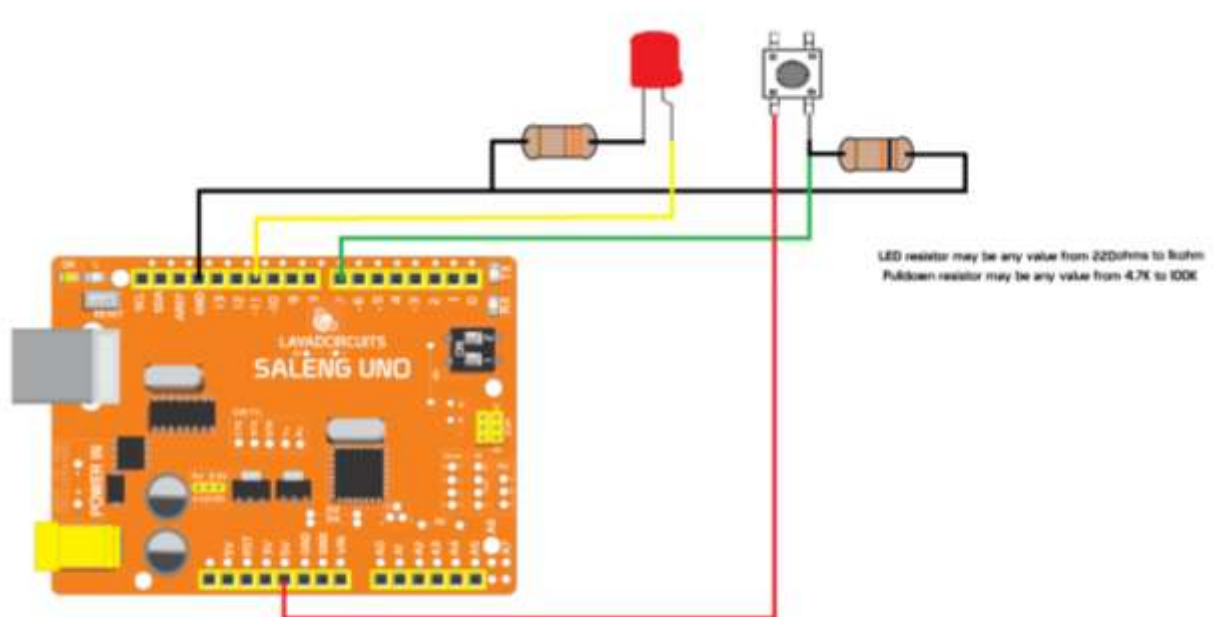
  //Blink the yellow LED 3 times
  for(int i=0;i<3;i++)
  {
    delay(500);
```

```
digitalWrite(yellowled, HIGH);
delay(500);
digitalWrite(yellowled, LOW);
}
delay(500);

// turn on the red LED for 5 seconds
digitalWrite(redled, HIGH);
delay(5000);
digitalWrite(redled, LOW);
}
```

CONTROL LED WITH PUSH BUTTON

A push button controls the LED at the output. Press and un-press it to turn the LED on and off. This example uses an external pull-down resistor on the button. Another solution though is to enable the internal pullup resistor on the digital pin using INPUT_PULLUP parameter instead of INPUT when calling pinMode().



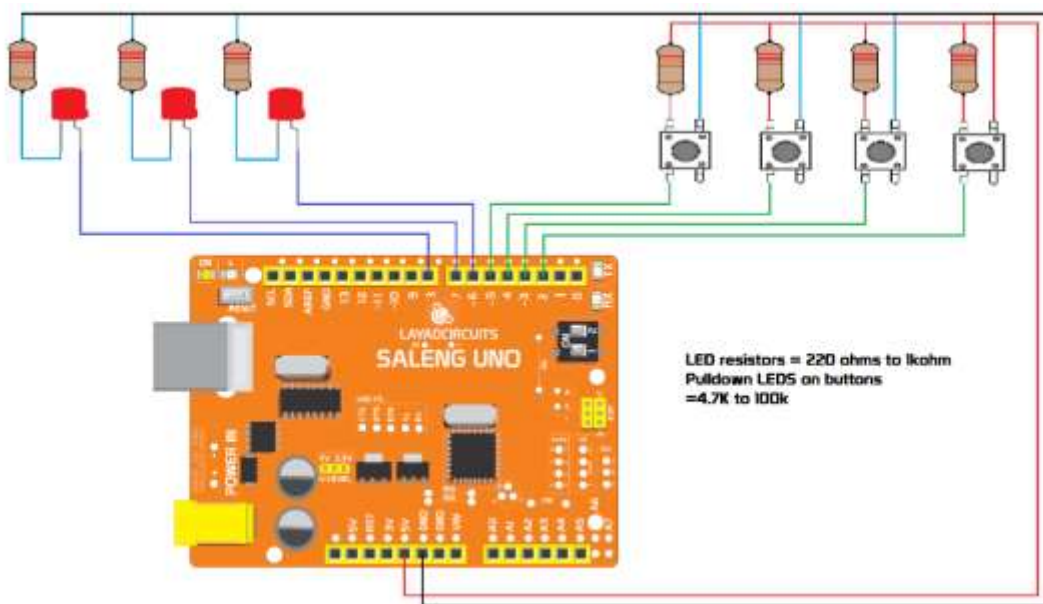
CODE:

```
int ledpin=11;
int inpin=7;
int val;
void setup()
{
  pinMode(ledpin,OUTPUT);
  pinMode(inpin,INPUT);
}
void loop()
{
```

```
val=digitalRead(inpin);
if (val==HIGH)
{
  digitalWrite(ledpin,LOW);
}
else
{
  digitalWrite(ledpin,HIGH);}
}
```

INDIVIDUAL LED CONTROL

This allows the user to control the LEDs individually that are wired externally.



CODE:

```
int redled=8;
int yellowled=7;
int greenled=6;
int redpin=5;
int yellowpin=4;
int greenpin=3;
int resetpin=2;
int red;
int yellow;
int green;
int reset;

void setup()
{
```

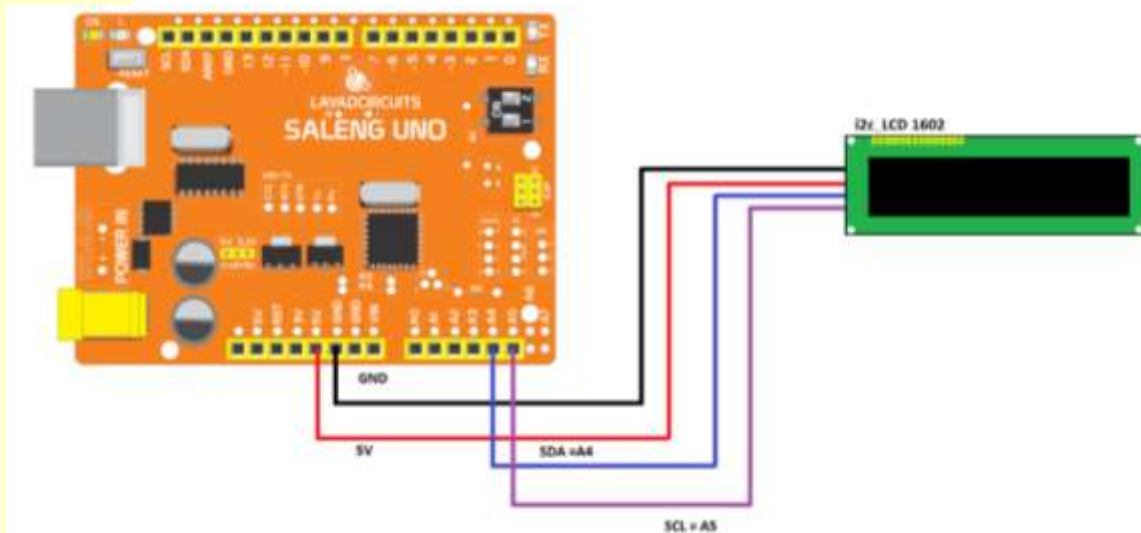
```
pinMode(redled, OUTPUT);
pinMode(yellowled, OUTPUT);
pinMode(greenled, OUTPUT);
pinMode(redpin, INPUT_PULLUP);
pinMode(yellowpin, INPUT_PULLUP);
pinMode(greenpin, INPUT_PULLUP);
pinMode(resetpin, INPUT_PULLUP);
}

void loop()
{
  red=digitalRead(redpin);
  yellow=digitalRead(yellowpin);
  green=digitalRead(greenpin);
  reset = digitalRead(resetpin);

  if (red==LOW)
  {
    digitalWrite(redled, HIGH);
    digitalWrite(greenled, LOW);
    digitalWrite(yellowled, LOW);
  }
  if (green==LOW)
  {
    digitalWrite(redled, LOW);
    digitalWrite(greenled, HIGH);
    digitalWrite(yellowled, LOW);
  }
  if (yellow==LOW)
  {
    digitalWrite(redled, LOW);
    digitalWrite(greenled, LOW);
    digitalWrite(yellowled, HIGH);
  }
  if (reset == LOW)
  {
    digitalWrite(redled, LOW);
    digitalWrite(greenled, LOW);
    digitalWrite(yellowled, LOW);
  }
}
```


LCD I2C

This is a 16x2 LCD display screen with I2C interface. It can display 16x2 characters on 2 lines, white characters on blue background or black characters on yellow-green background. that a library "<LiquidCrystal_I2C.h>" must be added to run the code. Download it from [here](#). Some modules may use the 0x27 address while some use 0x3F.



CODE:

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h> // to add this library, search
"<LiquidCrystal_I2C.h> df robot", click the suitable library you will find
and download the zip file. Open the Arduino ide, then click sketch, go to
include library and click the add .zip library.

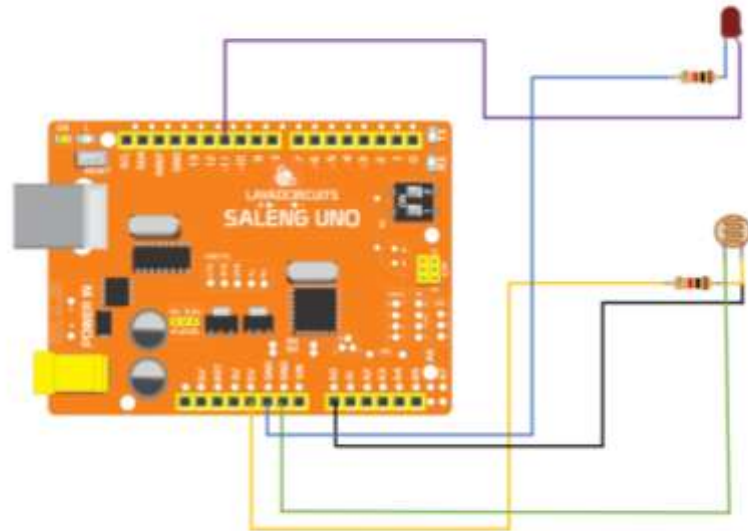
LiquidCrystal_I2C lcd(0x27,16,2); // change the 0x27 to 0x3F if you do not
see text.
//do note that you may also need to adjust the potentiometers at the back of
the I2C board.
//this controls the contrast of the LCD module.
void setup()
{
  lcd.init(); // initialize the lcd

  // Print a message to the LCD.
  lcd.backlight(); // turn on the backlight
  lcd.setCursor(0,0); // set the position (column, row) where we
should start printing
  lcd.print("Hello world!"); // print the data
}
```

```
void loop()
{
}
```

PHOTO SENSITIVE LAMP

An Arduino controlled lamp that is used not only to detect light but also to measures the brightness/illuminance level of the ambient light.

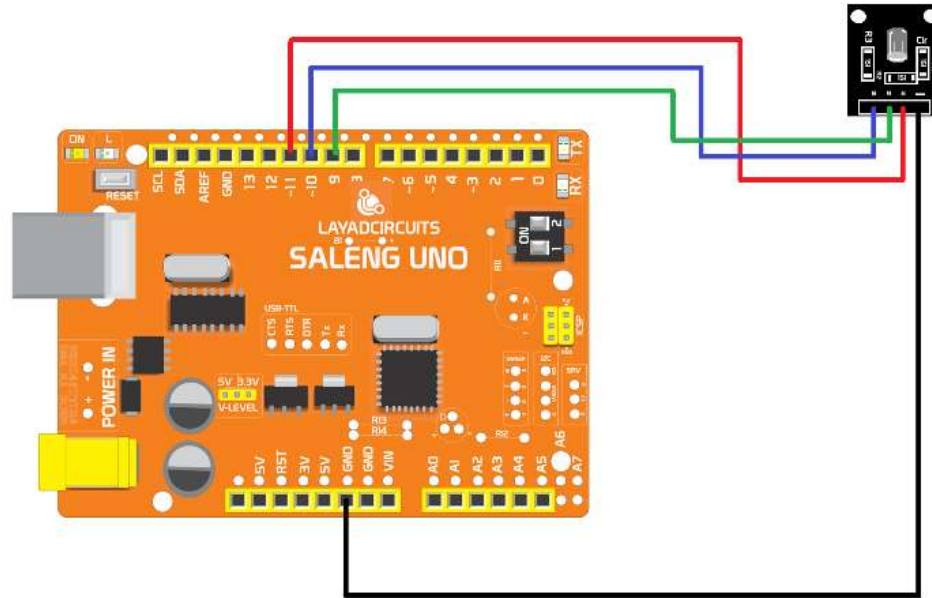


CODE:

```
int potpin=0;
int ledpin=11;
int val=0;
void setup()
{
  pinMode(ledpin,OUTPUT);
  Serial.begin(9600);
}
void loop()
{
  val=analogRead(potpin);
  Serial.println(val);
  analogWrite(ledpin,val);
  delay(10);
}
```

RGB LED

An RGB LED is basically a led package with the three colors:red, green and blue. By varying the brightness of each LED, we can produce other colors. The Arduino has an analogWrite() function which generates PWM signals on PWM-capable pins of the Arduino/Saleng Uno. This PWM action causes the variations in the brightness of each individual LED.



CODE:

```
int ledPin = 13;
int redPin = 11;
int greenPin = 9;
int bluePin = 10;

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

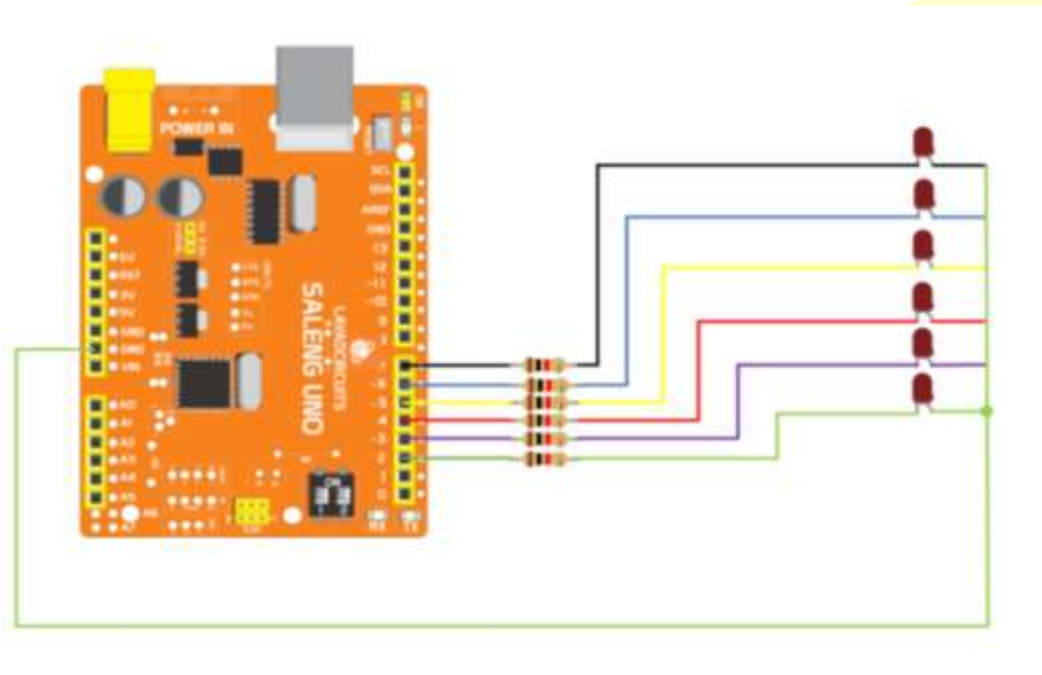
void loop() // run over and over again
{
```

```
// Basic colors:
color(255, 0, 0);
delay(1000);
color(0,255, 0);
delay(1000);
color(0, 0, 255);
delay(1000);

// Example blended colors:
color(255,255,0);
delay(1000);
color(255,255,255);
delay(1000);
color(128,0,255);
delay(1000);
color(0,0,0);
delay(1000);
}

void color (unsigned char red, unsigned char green, unsigned char blue)
{
    analogWrite(redPin, 255-red);
    analogWrite(bluePin, 255-blue);
    analogWrite(greenPin, 255-green);
}
}
```

RUNNING LED To make running LEDs, put code in the loop section so that it runs repeatedly. The running LED turns on each LED one at a time after delay elapses.



CODE:

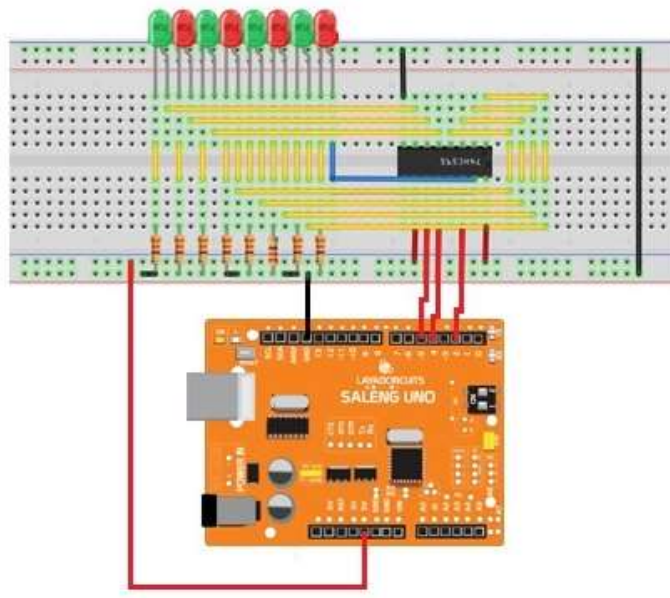
```
int BASE = 2 ; // starting pin/1st LED.
int NUM = 6;   // number of LEDs

void setup()
{
  for (int i = BASE; i < BASE + NUM; i ++)
  {
    pinMode(i, OUTPUT); //set all pins as outputs
  }
}

void loop()
{
  for (int i = BASE; i < BASE + NUM; i ++)
  {
    digitalWrite(i, LOW); // turn off the LED's in sequence
    delay(200);           // wait for 200ms
  }
  for (int i = BASE; i < BASE + NUM; i ++)
  {
    digitalWrite(i, HIGH); // turn off the LED's in sequence
    delay(200);           // wait for 200ms
  }
}
```

SHIFT REGISTER

A shift register allows you to expand the number of I/O pins you can use from your Arduino (or any microcontroller for that matter). And the 74HC595 shift register (nicknamed '595') is one of the most famous among all.



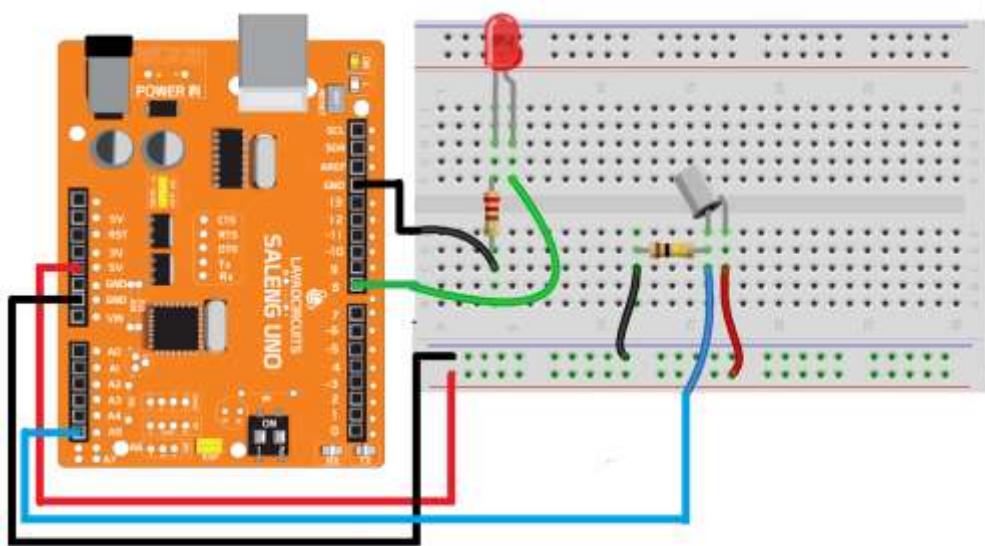
CODE:

```
int data = 2;
int clock = 5;
int latch = 4;
int ledState = 0;
const int ON = HIGH;
const int OFF = LOW;

void setup()
{
  pinMode(data, OUTPUT);
  pinMode(clock, OUTPUT);
  pinMode(latch, OUTPUT);
}
void loop()
{
  for(int i = 0; i < 256; i++)
  {
    updateLEDs(i);
    delay(500);
  }
}
void updateLEDs(int value)
{
  digitalWrite(latch, LOW);
  shiftOut(data, clock, MSBFIRST, ~value);
  digitalWrite(latch, HIGH);
}
```

TILT SWITCH LED CONTROL

A Tilt Sensor switch is an electronic device that detects the orientation of an object and gives its output High or Low accordingly. Tilt sensor can turn on or off the circuit based on the orientation.

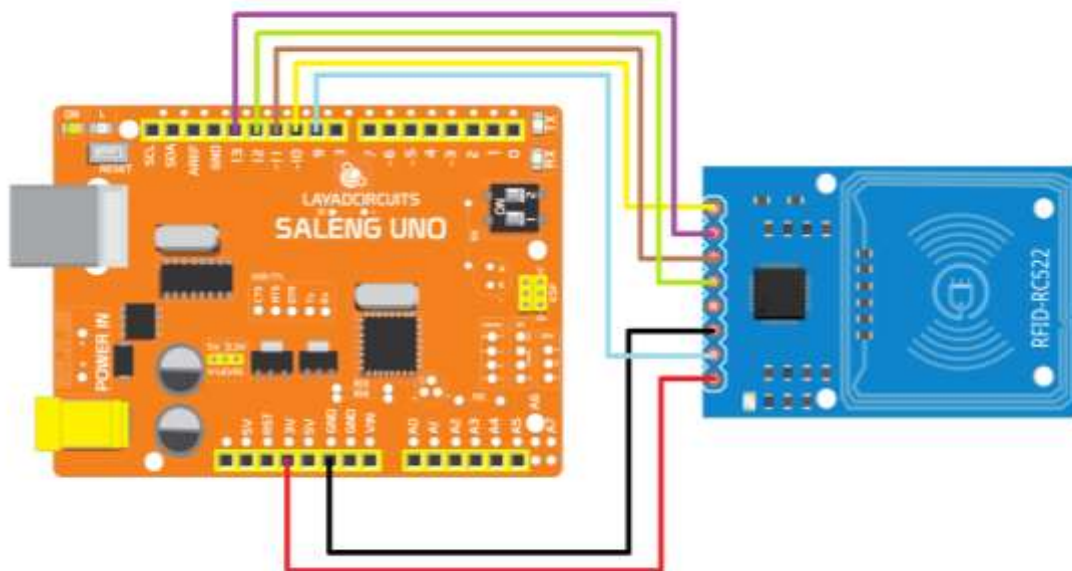


CODE:

```
void setup()
{
  pinMode(8,OUTPUT);
}
void loop()
{
  int i;
  while(1)
  {
    i=analogRead(5);
    if(i>512)
    {
      digitalWrite(8,LOW);
    }
    else
    {
      digitalWrite(8,HIGH);
    }
  }
}
```

RFID

This exercise uses RFID sensor that uses electromagnetic fields that transfer data over short distances. Every time you tap the RFID tag/card, you should see the RFID tag/card serial number on the serial monitor. Note that a library "<RFID.h>" must be added to run the code.

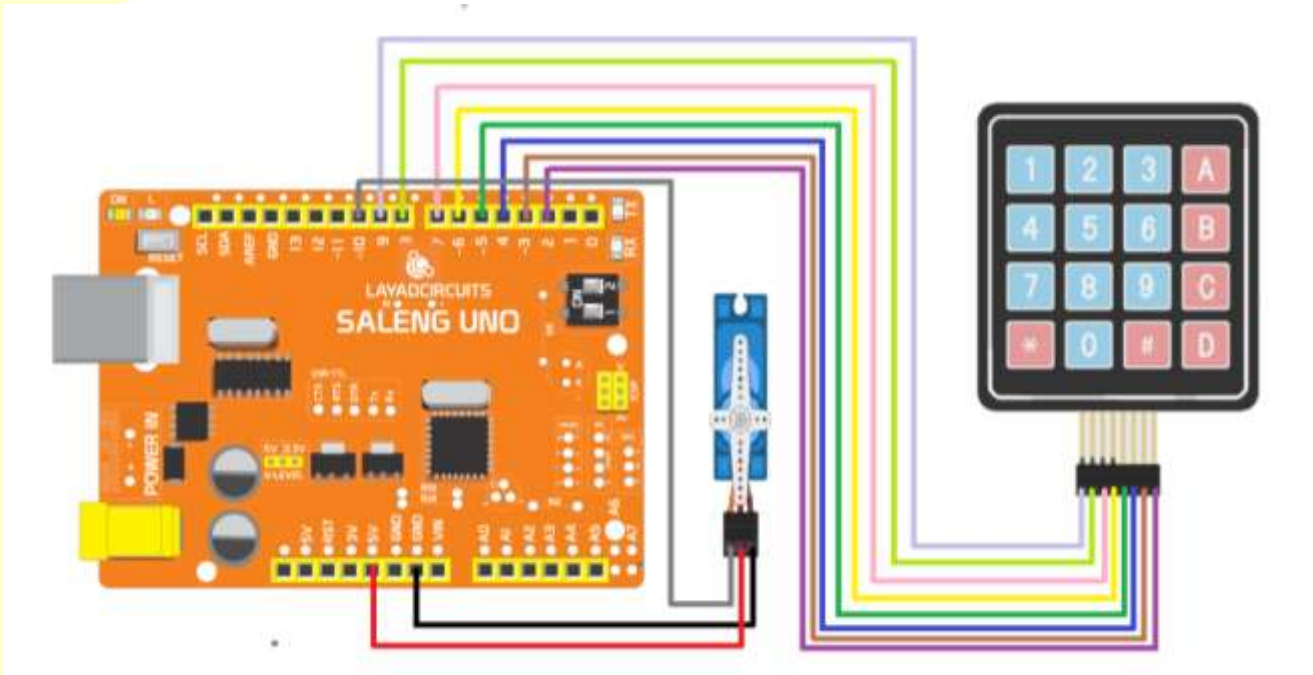


CODE:

```
#include <SPI.h>
#include <RFID.h>
#define SS_PIN 10
#define RST_PIN 9
RFID rfid(SS_PIN, RST_PIN);
String rfidCard;
void setup() {
  Serial.begin(9600);
  Serial.println("Starting the RFID Reader...");
  SPI.begin();
  rfid.init();
}
void loop() {
  if (rfid.isCard()) {
    if (rfid.readCardSerial()) {
      rfidCard = String(rfid.serNum[0]) + " " + String(rfid.serNum[1]) + " " +
String(rfid.serNum[2]) + " " + String(rfid.serNum[3]);
      Serial.println(rfidCard);
    }
    rfid.halt();
  }
}
```


KEYPAD DOOR LOCK

In this exercise, we want to turn a servo motor to a certain position after entering the right three-digit code on the keypad. The three-digit code can be modified in the program. Note that "<keypad.h>" library must be added to run the code.



CODE:

```
#include <Keypad.h>
#include <Servo.h>

#define codeLength 4
Servo myservo;

char Code[codeLength];
char PassW[codeLength]="123"; //password
byte keycount=0;

const byte ROWS = 3;
const byte COLS = 3;

char hexaKeys[ROWS][COLS] = {
  {'1', '2', '3'},
  {'4', '5', '6'},
  {'7', '8', '9'}
};
};
```

www.layadcircuits.com

Layad Circuits Electronics Engineering Supplies & Services, B314 Lopez Bldg., Session Rd. cor. Assumption Rd., Baguio City, Philippines
General inquiries: info@layadcircuits.com Sales: sales@layadcircuits.com FB: facebook.com/layadcircuits Mobile: +639164428565

An IMPORTANT NOTICE: at the end of this guide addresses availability, warranty, changes, use in safety-critical applications, intellectual property matters and other important disclaimers.

```
byte rowPins[ROWS] = {9, 8, 7};
byte colPins[COLS] = {5, 4, 3};
```

```
Keypad customKeypad = Keypad(makeKeymap(hexaKeys), rowPins, colPins, ROWS, COLS);
```

```
void setup(){
  Serial.begin(9600);
  myservo.attach(10);
}

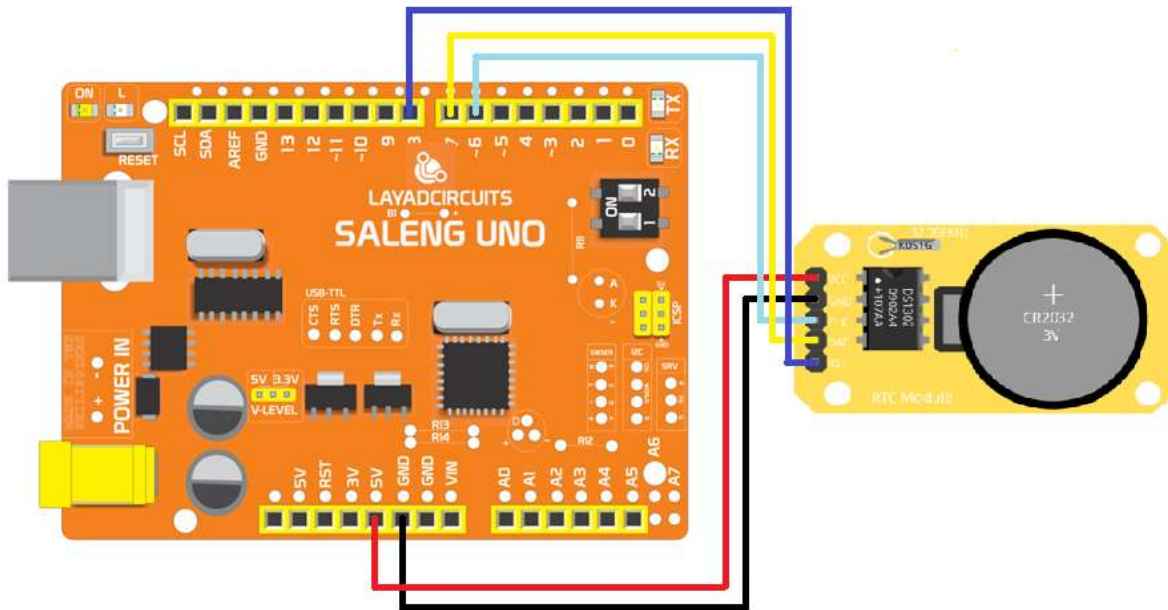
void loop(){
  char customKey = customKeypad.getKey();
  if(customKey){
    //Serial.println(customKey);
    Code[keycount]=customKey;
    Serial.print(Code[keycount]);
    keycount++;
  }
  if(keycount==codeLength-1){
    Serial.println(" ");

    if(!strcmp(Code,PassW)){
      Serial.println("Correct");
      myservo.write(80); //unlock
      delay(4000);
    }
    else{
      Serial.println("Incorrect");
      myservo.write(10); //lock
      delay(1000);
    }
  }
  deleteCount();
}

void deleteCount(){
  while(keycount !=0){
    Code[keycount--]=0;
  }
  return;
}
```

RTC MODULE

This exercise uses the DS1302 Real-Time Clock module. Set the current date and time in the following format: “seconds, minutes, hours, day of the week, day of the month, month, year”. Timestamps are observed on the serial monitor. Note that a library “<virtuabotixRTC.h>” must be added to run the code.



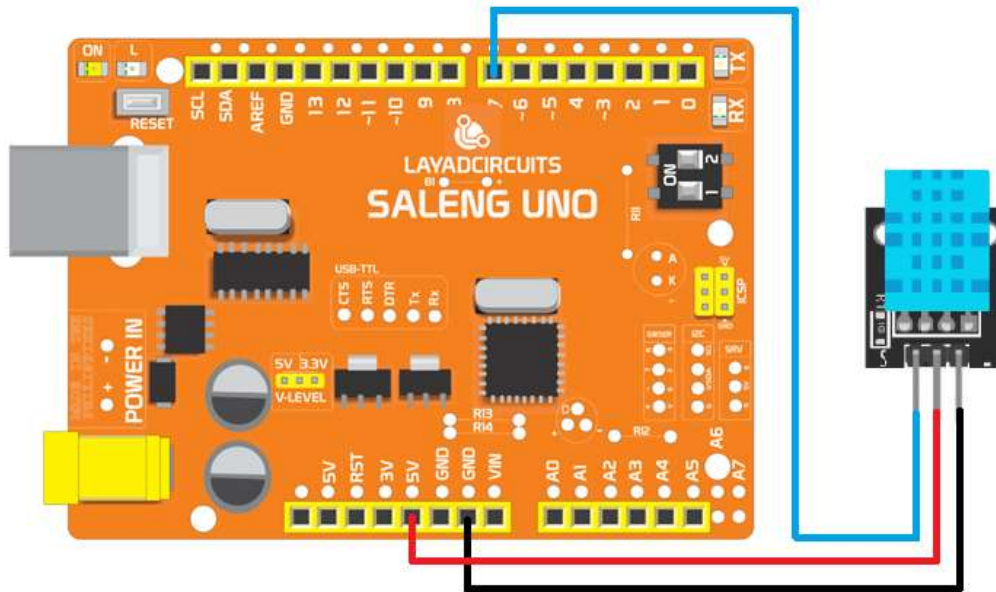
CODE:

```
#include <virtuabotixRTC.h>
// Creation of the Real Time Clock Object
virtuabotixRTC myRTC(6, 7, 8);
void setup() {
  Serial.begin(9600);
  // Set the current date, and time in the following format:
  // seconds, minutes, hours, day of the week, day of the month, month, year
  myRTC.setDS1302Time(18, 10, 10, 7, 18, 9, 2020);
}
void loop() {
  // This allows for the update of variables for time or accessing the individual elements.
  myRTC.updateTime();
  // Start printing elements as individuals
  Serial.print("Current Date / Time: ");
  Serial.print(myRTC.dayofmonth);
  Serial.print("/");
  Serial.print(myRTC.month);
  Serial.print("/");
  Serial.print(myRTC.year);
  Serial.print(" ");
}
```

```
Serial.print(myRTC.hours);  
Serial.print(":");  
Serial.print(myRTC.minutes);  
Serial.print(":");  
Serial.println(myRTC.seconds);  
// Delay so the program doesn't print non-stop  
delay(500);  
}
```

HUMIDITY AND TEMPERATURE SENSOR

DHT11 sensor is a commonly used humidity and temperature sensor. It can measure temperature from 0 °C to 50 °C. The humidity and temperature changes can be observed on the serial monitor. Note that a library "<dht.h>" must be added to run the code.



CODE:

```
#include <dht.h>

dht DHT;

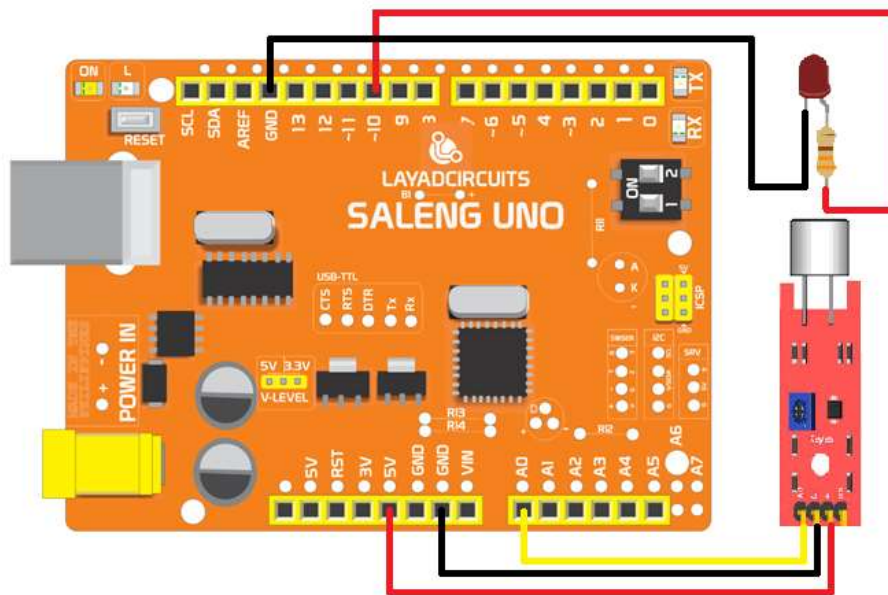
#define DHT11_PIN 7

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

void loop() {
  int chk = DHT.read11(DHT11_PIN);
  Serial.print("Temperature = ");
  Serial.println(DHT.temperature);
  Serial.print("Humidity = ");
  Serial.println(DHT.humidity);
  delay(1000);
}
```

CONTROL LED USING SOUND SENSOR

This exercise shows how you can control LED by clapping with the help of the sound sensor. Note that any sound may activate the LED. The sound sensor has a built-in multi-turn potentiometer to adjust the sensitivity of the digital output pin.



CODE:

```
int Sensor = A0;
int clap = 0;
long detection_range_start = 0;
long detection_range = 0;
boolean status_lights = false;

void setup() {
  pinMode(Sensor, INPUT);
  pinMode(10, OUTPUT);
}

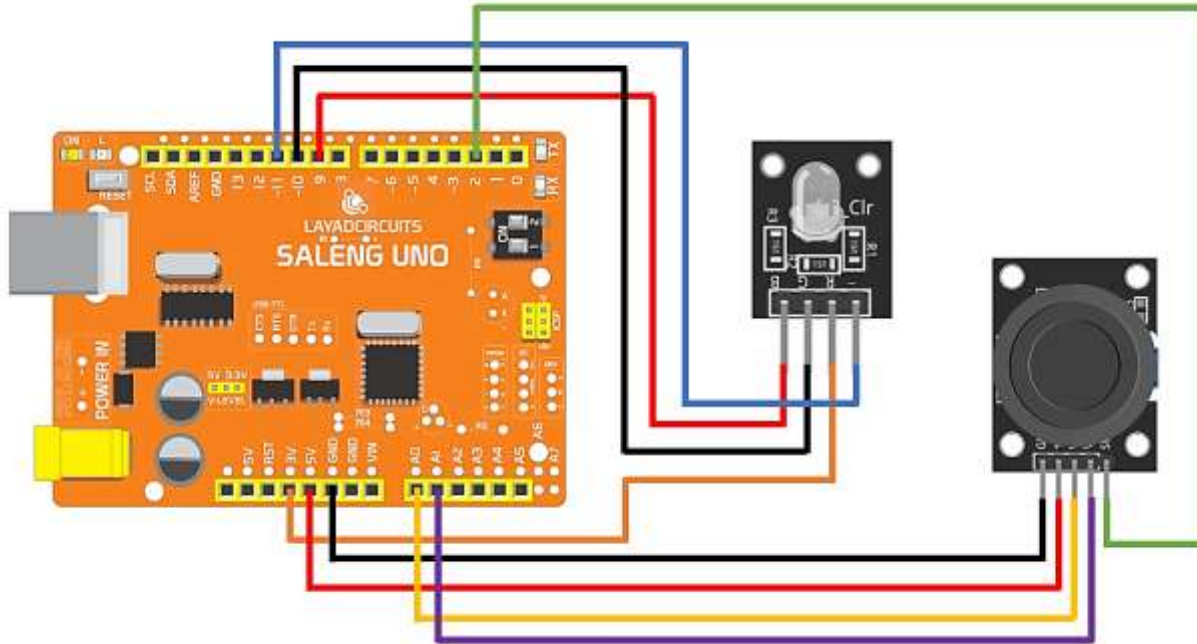
void loop() {
  int status_sensor = digitalRead(Sensor);
  if (status_sensor == 0)
  {
    if (clap == 0)
    {
      detection_range_start = detection_range = millis();
      clap++;
    }
    else if (clap > 0 && millis()-detection_range >= 50)

```

```
{
detection_range = millis();
clap++;
}
}
if (millis()-detection_range_start >= 400)
{
if (clap == 2)
{
if (!status_lights)
{
status_lights = true;
digitalWrite(10, HIGH);
}
else if (status_lights)
{
status_lights = false;
digitalWrite(10, LOW);
}
}
}
clap = 0;
}
}
```

CONTROL RGB LED WITH JOYSTICK

In this exercise, we will use an analog joystick module to control the RGB LED lights. The analog joystick is similar to two potentiometers connected together, one for the vertical movement (Y-axis) and other for the horizontal movement (X-axis). The joystick also comes with a select switch. Here, the RGB led lights' color will depend on the analog joystick's orientation.



CODE:

```
const int PUSHBUTTON_PIN = 2;
const int RED_PIN = 11;
const int GREEN_PIN = 10;
const int BLUE_PIN = 9;

const int redX = 512;
const int redY = 1023;
const int greenX = 1023;
const int greenY = 0;
const int blueX = 0;
const int blueY = 0;

void setup() {
  Serial.begin(9600);
  // Set the Joystick button as an input
  pinMode(PUSHBUTTON_PIN, INPUT);
  digitalWrite(PUSHBUTTON_PIN, HIGH);

  pinMode(RED_PIN, OUTPUT);
  pinMode(GREEN_PIN, OUTPUT);
  pinMode(BLUE_PIN, OUTPUT);
}

void loop() {
  int xAxis = analogRead(A0);
  int yAxis = analogRead(A1);

  // Flip orientation (if needed)
  xAxis = map(xAxis, 0, 1023, 0, 1023);
  yAxis = map(yAxis, 0, 1023, 1023, 0);
}
```



```

int distanceRed = sqrt(pow(abs(redX - xAxis), 2) + pow(abs(redY - yAxis), 2));
int distanceGreen = sqrt(pow(abs(greenX - xAxis), 2) + pow(abs(greenY - yAxis), 2));
int distanceBlue = sqrt(pow(abs(blueX - xAxis), 2) + pow(abs(blueY - yAxis), 2));

int brightRed = 255 - constrain(map(distanceRed, 0, 1023, 0, 255), 0, 255);
int brightGreen = 255 - constrain(map(distanceGreen, 0, 1023, 0, 255), 0, 255);
int brightBlue = 255 - constrain(map(distanceBlue, 0, 1023, 0, 255), 0, 255);

if (digitalRead(PUSHBUTTON_PIN) == 0) {
  brightRed = 255;
  brightGreen = 255;
  brightBlue = 255;
}

analogWrite(RED_PIN, brightRed);
analogWrite(GREEN_PIN, brightGreen);
analogWrite(BLUE_PIN, brightBlue);

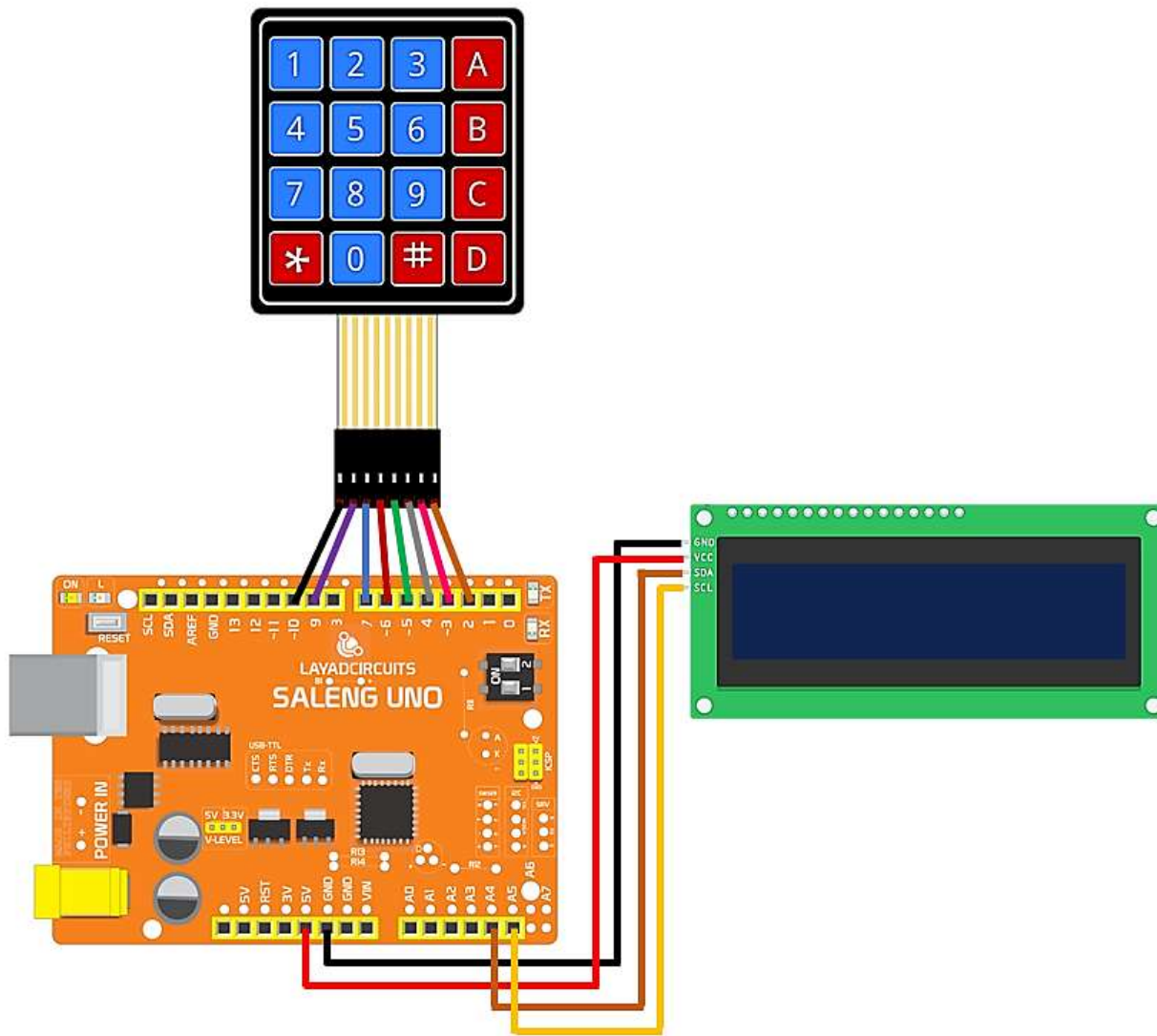
Serial.print("KEY: ");
Serial.print(digitalRead(PUSHBUTTON_PIN));
Serial.print(", X: ");
Serial.print(xAxis);
Serial.print(", Y: ");
Serial.print(yAxis);
Serial.print(", R: ");
Serial.print(brightRed);
Serial.print(", G: ");
Serial.print(brightGreen);
Serial.print(", B: ");
Serial.print(brightBlue);
Serial.println("\n");

delay(100);
}

```

PRINT 4X4 KEYPAD INPUT TO LCD

This calculator can perform basic addition, subtraction, multiplication, and division calculations. In this exercise, we will use an Arduino, LCD and keypad to create a calculator. The results can be displayed on an LCD screen (16x2 Dot-matrix) when the values are entered using a keypad (4x4 keypad). With whole numbers, this calculator could do basic operations like addition, subtraction, multiplication, and division.



CODE:

```
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27, 20, 2); // set the LCD address to 0x27 for a 16 chars and 2
line display

#include <Keypad.h>

const byte ROWS = 4;
const byte COLS = 4;

char keys [ROWS] [COLS] = {
  {'1', '2', '3', '+'},
  {'4', '5', '6', '-'},
  {'7', '8', '9', '*'},
  {'C', '0', '=', '/'}
};

byte rowPins[ROWS] = {10, 9, 7, 6};
```

```
byte colPins[COLS] = {5, 4, 3, 2};
```

```
Keypad myKeypad = Keypad( makeKeymap(keys), rowPins, colPins, ROWS, COLS );
```

```
boolean presentValue = false;
boolean next = false;
boolean final = false;
String num1, num2;
int answer = 0;
char op;
```

```
void setup()
```

```
{
  lcd.init();
  lcd.backlight();
  lcd.setCursor(3,0);
  lcd.print("4x4 Keypad");
  lcd.setCursor(3,1);
  lcd.print("Calculator");
  delay(3000);
  lcd.clear();
}
```

```
void loop() {
```

```
  char key = myKeypad.getKey();

  if (key != NO_KEY && (key == '1' || key == '2' || key == '3' || key == '4' || key ==
'5' || key == '6' || key == '7' || key == '8' || key == '9' || key == '0'))
  {
    if (presentValue != true)
    {
      num1 = num1 + key;
      int numLength = num1.length();
      lcd.setCursor(0, 0);
      lcd.print(num1);
    }
    else
    {
      num2 = num2 + key;
      int numLength = num2.length();
      int numLength1 = num1.length();
      lcd.setCursor(1 + numLength1, 0);
      lcd.print(num2);
      final = true;
    }
  }

  else if (presentValue == false && key != NO_KEY && (key == '/' || key == '*' || key ==
'-' || key == '+'))
  {
    if (presentValue == false)
    {
      int numLength = num1.length();
```

```

    presentValue = true;
    op = key;
    lcd.setCursor(0 + numLength, 0);
    lcd.print(op);
  }
}

else if (final == true && key != NO_KEY && key == '=') {

  if (op == '+') {
    answer = num1.toInt() + num2.toInt();
  }
  else if (op == '-') {
    answer = num1.toInt() - num2.toInt();
  }
  else if (op == '*') {
    answer = num1.toInt() * num2.toInt();
  }
  else if (op == '/') {
    answer = num1.toInt() / num2.toInt();
  }
  lcd.clear();
  lcd.setCursor(16, 1);
  lcd.autoscroll();
  lcd.print(answer);
  lcd.noAutoscroll();

}

else if (key != NO_KEY && key == 'C') {
  lcd.clear();
  presentValue = false;
  final = false;
  num1 = "";
  num2 = "";
  answer = 0;
  op = ' ';
}
}

```

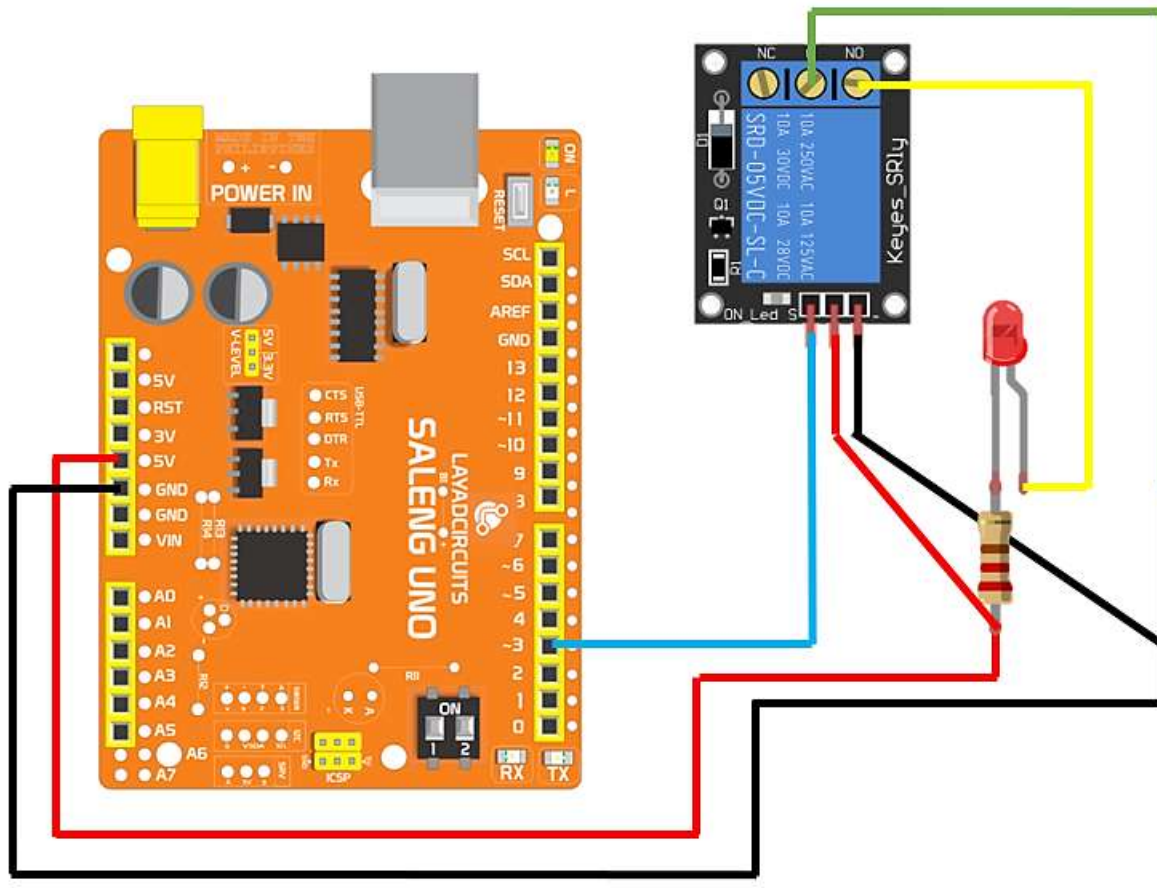
BASIC SETUP FOR ARDUINO WITH RELAY

In this exercise, we will execute the basic setup for Arduino relay.

Take note:

- High voltage connections can be made to this screw terminal. For example: Bulb, ceiling fan etc., But in this project we are just using an LED.
- When you make the connection between a) and b), the connected LED is always ON until it receives a signal from the Arduino to turn it OFF.

- When you make the connection between b) and c), the connected LED is OFF until it receives a signal from the Arduino to turn it ON.



CODE:

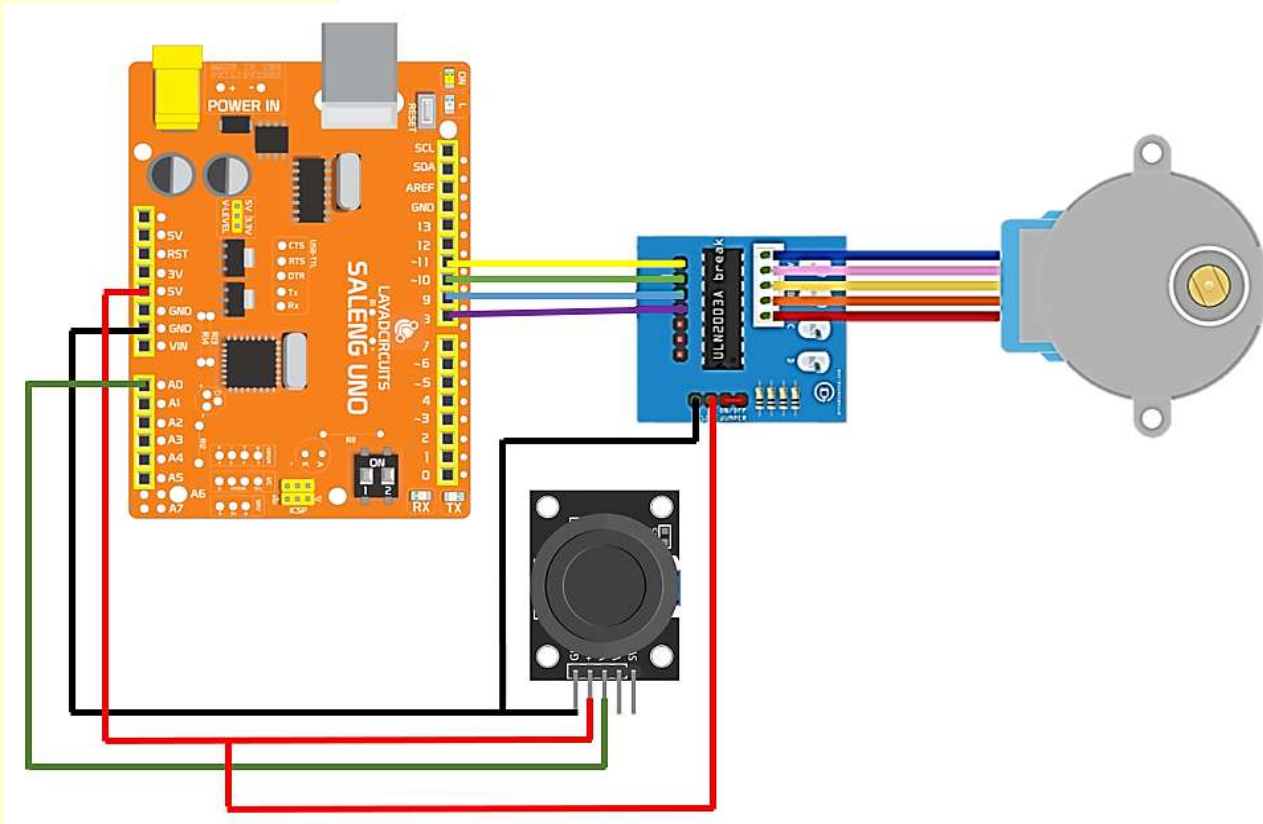
```
int Relaypin= 3; // Define input pin for relay

void setup() {
  // put your setup code here, to run once:
  pinMode(Relaypin, OUTPUT); // Define the Relaypin as output pin
}

void loop() {
  // put your main code here, to run repeatedly:
  digitalWrite(Relaypin, HIGH); // Sends high signal
  delay(1000); // Waits for 1 second
  digitalWrite(Relaypin, LOW); // Makes the signal low
  delay(1000); // Waits for 1 second
}
```

DRIVING STEPPER MOTOR CONTROL WITH JOYSTICK

This exercise shows how to control stepper motor speed and direction of rotation using Arduino UNO board and Analog PS2 joystick module. Note that an Arduino library "<Stepper.h >" must be added to run the code.



CODE:

```
// include Arduino stepper motor library
#include <Stepper.h>

// define number of steps per revolution
#define STEPS 32

// define stepper motor control pins
#define IN1 11
#define IN2 10
#define IN3 9
#define IN4 8

// initialize stepper library
Stepper stepper(STEPS, IN4, IN2, IN3, IN1);

// joystick pot output is connected to Arduino A0
```

www.layadcircuits.com

Layad Circuits Electronics Engineering Supplies & Services, B314 Lopez Bldg., Session Rd. cor. Assumption Rd., Baguio City, Philippines
General inquiries: info@layadcircuits.com Sales: sales@layadcircuits.com FB: facebook.com/layadcircuits Mobile: +639164428565

An IMPORTANT NOTICE: at the end of this guide addresses availability, warranty, changes, use in safety-critical applications, intellectual property matters and other important disclaimers.

```
#define joystick A0

void setup()
{
}

void loop()
{
  // read analog value from the potentiometer
  int val = analogRead(joystick);

  // if the joystick is in the middle ==> stop the motor
  if( (val > 500) && (val < 523) )
  {
    digitalWrite(IN1, LOW);
    digitalWrite(IN2, LOW);
    digitalWrite(IN3, LOW);
    digitalWrite(IN4, LOW);
  }

  else
  {
    // move the motor in the first direction
    while (val >= 523)
    {
      // map the speed between 5 and 500 rpm
      int speed_ = map(val, 523, 1023, 5, 500);
      // set motor speed
      stepper.setSpeed(speed_);

      // move the motor (1 step)
      stepper.step(1);

      val = analogRead(joystick);
    }

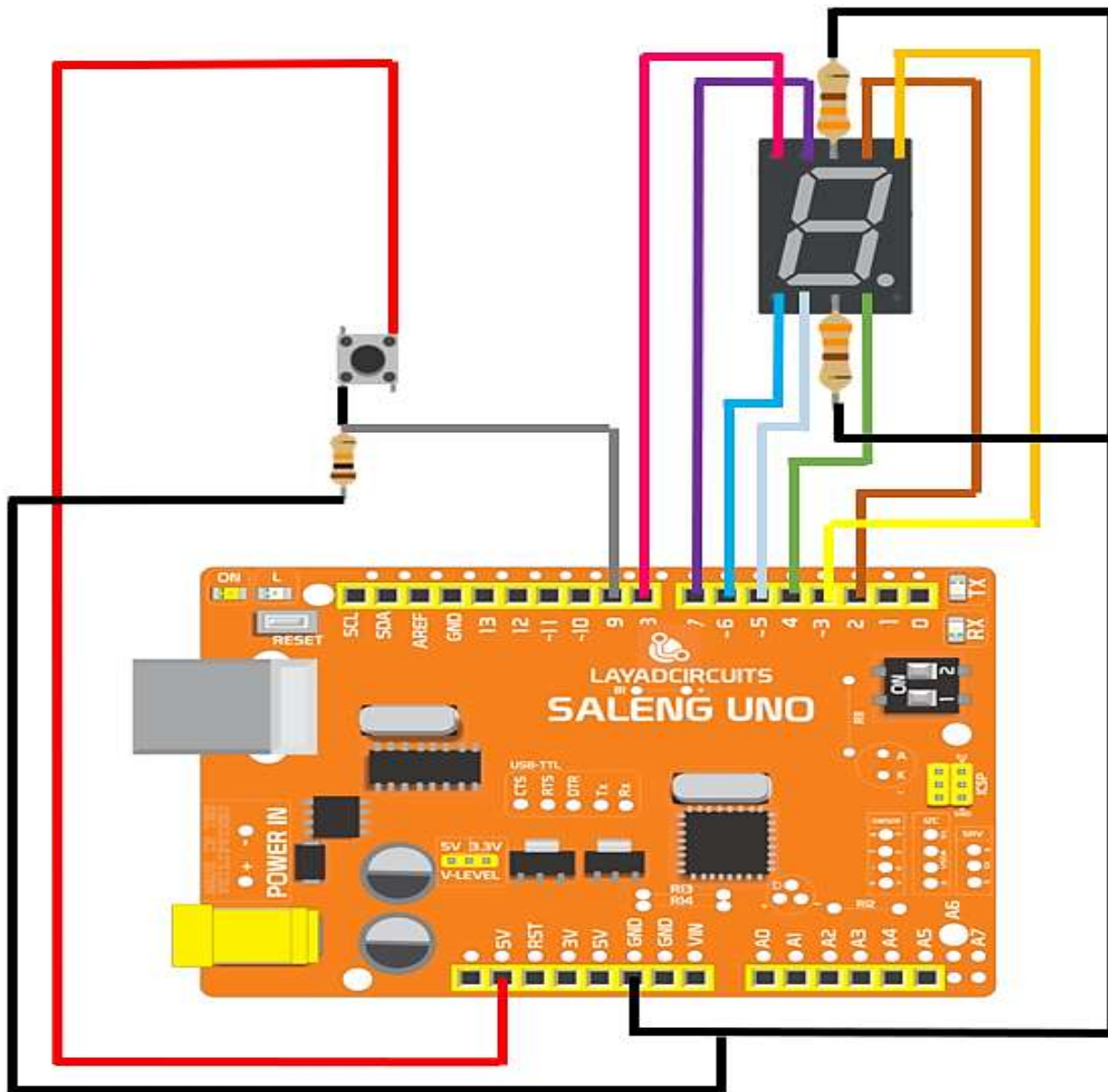
    // move the motor in the other direction
    while (val <= 500)
    {
      // map the speed between 5 and 500 rpm
      int speed_ = map(val, 500, 0, 5, 500);
      // set motor speed
      stepper.setSpeed(speed_);

      // move the motor (1 step)
      stepper.step(-1);

      val = analogRead(joystick);
    }
  }
}
```

SEGMENT DISPLAY CONTROLLED BY PUSHBUTTON

In this exercise, the 7-segment display will count automatically from 1 up to the last character programmed, however pushing the push button will change how it operates. If pressed, it will automatically enter a number counted by twos. Additionally, it counts in thirds if you press it a second time, and so on. Letters are also encoded as another illustration to show of how the button works.



CODE:

www.layadcircuits.com

Layad Circuits Electronics Engineering Supplies & Services, B314 Lopez Bldg., Session Rd. cor. Assumption Rd., Baguio City, Philippines
General inquiries: info@layadcircuits.com Sales: sales@layadcircuits.com FB: facebook.com/layadcircuits Mobile: +639164428565

An IMPORTANT NOTICE: at the end of this guide addresses availability, warranty, changes, use in safety-critical applications, intellectual property matters and other important disclaimers.


```

int counter_value=1;
int number=0;
int table[16][7]={
  {1,1,1,1,1,1,0}, //0
  {0,1,1,0,0,0,0}, //1
  {1,1,0,1,1,0,1}, //2
  {1,1,1,1,0,0,1}, //3
  {0,1,1,0,0,1,1}, //4
  {1,0,1,1,0,1,1}, //5
  {1,0,1,1,1,1,1}, //6
  {1,1,1,0,0,0,0}, //7
  {1,1,1,1,1,1,1}, //8
  {1,1,1,0,0,1,1}, //9
  {1,1,1,1,1,0,1}, //a
  {0,0,1,1,1,1,1}, //b
  {0,0,0,1,1,0,1}, //c
  {0,1,1,1,1,0,1}, //d
  {1,1,0,1,1,1,1}, //e
  {1,0,0,0,1,1,1} //f
};
unsigned long previousMillis = 0,currentMillis;
int prev_button=0;

void setup()
{
  for(int i=2; i<9; i++)
    pinMode(i,OUTPUT);
  pinMode(9, INPUT);
  for(int pin=0; pin<7; pin++)
    digitalWrite(pin+2,table[number][pin]);
}

void loop()
{
  currentMillis = millis();
  if(digitalRead(9) && !prev_button)
  {
    prev_button=1;
    ++counter_value%=16;
    delay(30);
  }
  else if(!digitalRead(9) && prev_button)
    prev_button=0;

  if (currentMillis - previousMillis >= 1000)
  {
    previousMillis = currentMillis;
    number+=counter_value;
    number%=16;

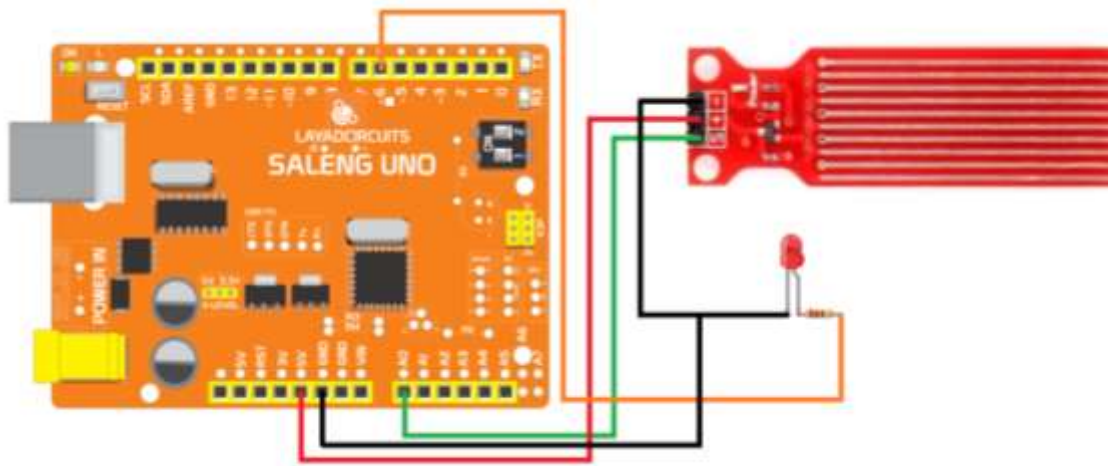
    for(int pin=0; pin<7; pin++)
      digitalWrite(pin+2,table[number][pin]);
  }

  delay(20);
}

```

WATER LEVEL SENSOR

The PCB of the water level sensor is made out of long conductive plates. When the water reaches a certain level, the conductivity between the two plates changes, and by measuring the changes we can measure the approximate water level.



CODE:

```
#define ledPin 6

#define sensorPin A0

void setup() {

  Serial.begin(9600);

  pinMode(ledPin, OUTPUT);

  digitalWrite(ledPin, LOW);

}

void loop()

{

  unsigned int sensorValue = analogRead(sensorPin);
```

www.layadcircuits.com

Layad Circuits Electronics Engineering Supplies & Services, B314 Lopez Bldg., Session Rd. cor. Assumption Rd., Baguio City, Philippines
General inquiries: info@layadcircuits.com Sales: sales@layadcircuits.com FB: facebook.com/layadcircuits Mobile: +639164428565

An IMPORTANT NOTICE: at the end of this guide addresses availability, warranty, changes, use in safety-critical applications, intellectual property matters and other important disclaimers.

```

if (sensorValue < 540)
    return;

uint8_t outputValue = map(sensorValue, 540, 800, 0, 255);

Serial.print(sensorValue);

Serial.print(" ");

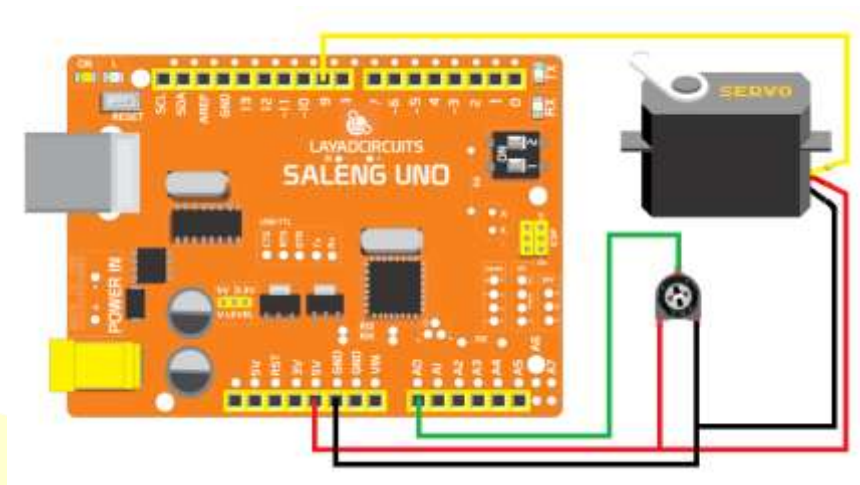
Serial.println(outputValue);

analogWrite(ledPin, outputValue); // generate PWM signal
}

```

SERVO MOTIOR CONTROLLED BY POTENTIOMETER

This shows how to control a servo position using a potentiometer. Note that an Arduino library "<Servo.h >" must be added to run the code.



CODE:

```

#include <Servo.h> // include the required Arduino library

#define servoPin 9 // Arduino pin for the servo
#define potPin A0 // Arduino pin for the potentiometer

int angle = 0; // variable to store the servo position in degrees
int reading = 0; // variable to store the reading from the analog input

Servo myservo; // create a new object of the servo class

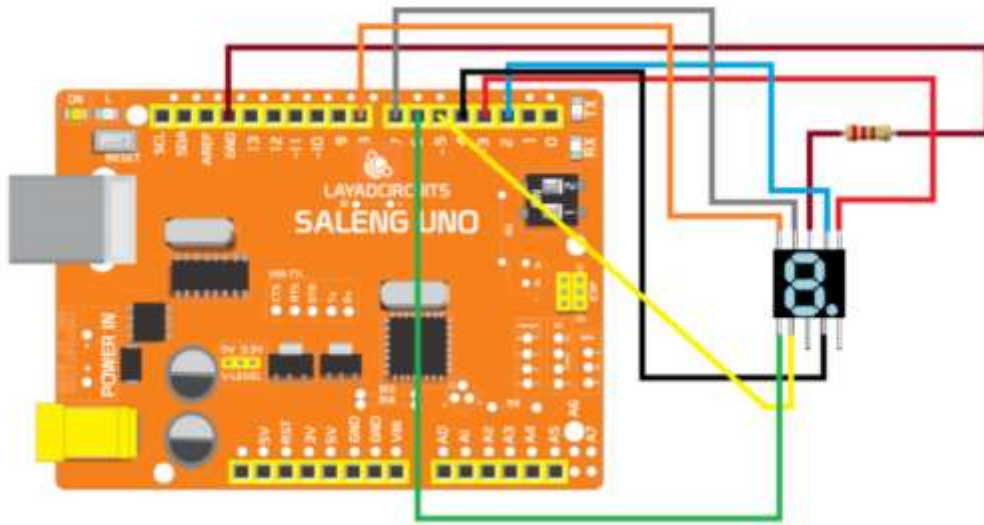
```

```
void setup() {
  myservo.attach(servoPin);
}

void loop() {
  reading = analogRead(potPin); // read the analog input
  angle = map(reading, 0, 1023, 0, 180); // map the input to a value between 0 and 180
  degrees
  myservo.write(angle); // tell the servo to go to the set position
  delay(15); // wait 15 ms for the servo to reach the position
}
```

7-SEGMENT DISPLAY COUNTER

In this exercise we are going to interface a seven segment display to ARDUINO UNO. The display counts from 0-9 and resets itself to zero.



CODE:

```
#define segA 2//connecting segment A to PIN2

#define segB 3// connecting segment B to PIN3

#define segC 4// connecting segment C to PIN4

#define segD 5// connecting segment D to PIN5
```

www.layadcircuits.com

Layad Circuits Electronics Engineering Supplies & Services, B314 Lopez Bldg., Session Rd. cor. Assumption Rd., Baguio City, Philippines
General inquiries: info@layadcircuits.com Sales: sales@layadcircuits.com FB: facebook.com/layadcircuits Mobile: +639164428565

An IMPORTANT NOTICE: at the end of this guide addresses availability, warranty, changes, use in safety-critical applications, intellectual property matters and other important disclaimers.

```
#define segE 6// connecting segment E to PIN6

#define segF 7// connecting segment F to PIN7

#define segG 8// connecting segment G to PIN8

int COUNT=0;//count integer for 0-9 increment

void setup()

{
for (int i=2;i<9;i++)

{
pinMode(i, OUTPUT);// taking all pins from 2-8 as output

}
}

void loop()

{
switch (COUNT)
{

case 0://when count value is zero show"0" on disp

digitalWrite(segA, HIGH);

digitalWrite(segB, HIGH);

digitalWrite(segC, HIGH);

digitalWrite(segD, HIGH);
```

```
digitalWrite(segE, HIGH);  
digitalWrite(segF, HIGH);  
digitalWrite(segG, LOW);  
break;
```

```
case 1:// when count value is 1 show"1" on disp
```

```
digitalWrite(segA, LOW);  
digitalWrite(segB, HIGH);  
digitalWrite(segC, HIGH);  
digitalWrite(segD, LOW);  
digitalWrite(segE, LOW);  
digitalWrite(segF, LOW);  
digitalWrite(segG, LOW);  
break;
```

```
case 2:// when count value is 2 show"2" on disp
```

```
digitalWrite(segA, HIGH);  
digitalWrite(segB, HIGH);  
digitalWrite(segC, LOW);  
digitalWrite(segD, HIGH);  
digitalWrite(segE, HIGH);  
digitalWrite(segF, LOW);  
digitalWrite(segG, HIGH);  
break;
```

```
case 3:// when count value is 3 show"3" on disp
```

```
digitalWrite(segA, HIGH);  
digitalWrite(segB, HIGH);  
digitalWrite(segC, HIGH);  
digitalWrite(segD, HIGH);  
digitalWrite(segE, LOW);  
digitalWrite(segF, LOW);  
digitalWrite(segG, HIGH);  
  
break;
```

```
case 4:// when count value is 4 show"4" on disp
```

```
digitalWrite(segA, LOW);  
digitalWrite(segB, HIGH);  
digitalWrite(segC, HIGH);  
digitalWrite(segD, LOW);  
digitalWrite(segE, LOW);  
digitalWrite(segF, HIGH);  
digitalWrite(segG, HIGH);  
  
break;
```

```
case 5:// when count value is 5 show"5" on disp
```

```
digitalWrite(segA, HIGH);  
digitalWrite(segB, LOW);  
digitalWrite(segC, HIGH);  
digitalWrite(segD, HIGH);  
digitalWrite(segE, LOW);  
digitalWrite(segF, HIGH);
```

```
digitalWrite(segG, HIGH);  
break;
```

```
case 6:// when count value is 6 show"6" on disp
```

```
digitalWrite(segA, HIGH);  
digitalWrite(segB, LOW);  
digitalWrite(segC, HIGH);  
digitalWrite(segD, HIGH);  
digitalWrite(segE, HIGH);  
digitalWrite(segF, HIGH);  
digitalWrite(segG, HIGH);  
break;
```

```
case 7:// when count value is 7 show"7" on disp
```

```
digitalWrite(segA, HIGH);  
digitalWrite(segB, HIGH);  
digitalWrite(segC, HIGH);  
digitalWrite(segD, LOW);  
digitalWrite(segE, LOW);  
digitalWrite(segF, LOW);  
digitalWrite(segG, LOW);  
break;
```

```
case 8:// when count value is 8 show"8" on disp
```

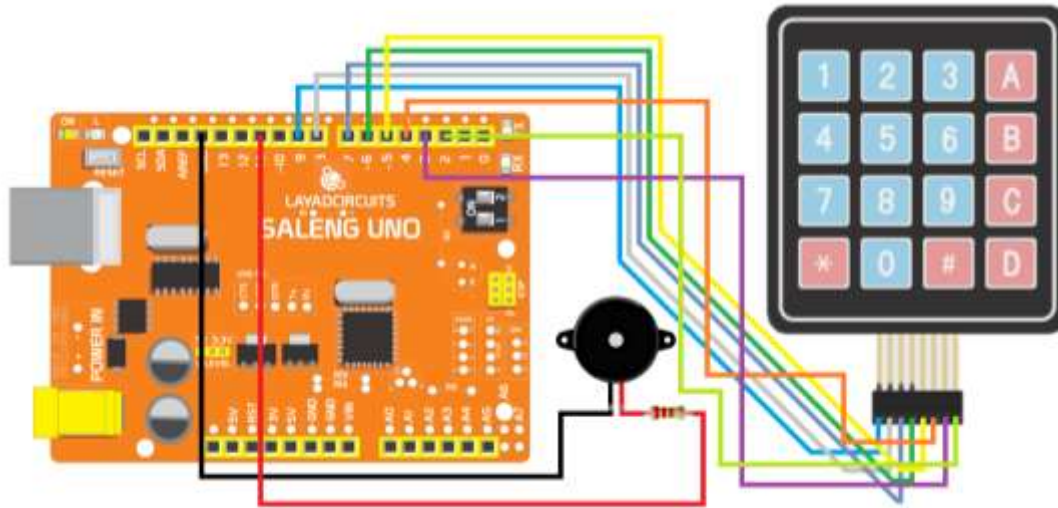
```
digitalWrite(segA, HIGH);  
digitalWrite(segB, HIGH);
```



```
digitalWrite(segC, HIGH);  
digitalWrite(segD, HIGH);  
digitalWrite(segE, HIGH);  
digitalWrite(segF, HIGH);  
digitalWrite(segG, HIGH);  
  
break;  
  
case 9:// when count value is 9 show"9" on disp  
  
digitalWrite(segA, HIGH);  
digitalWrite(segB, HIGH);  
digitalWrite(segC, HIGH);  
digitalWrite(segD, HIGH);  
digitalWrite(segE, LOW);  
digitalWrite(segF, HIGH);  
digitalWrite(segG, HIGH);  
  
break;  
  
}  
  
if (COUNT<10)  
{  
    COUNT++;  
    delay(1000);///increment count integer for every second  
}  
  
if (COUNT==10)  
{  
    COUNT=0;// if count integer value is equal to 10, reset it to zero.  
    delay(1000);  
}  
  
}
```

4x4 KEYPAD MUSIC PLAYER

Create music using Arduino, buzzer, 4x4 keypad and connecting cables. Note that an Arduino library "<Keypad.h>" must be added to run the code.



CODE:

```
#include <Keypad.h>

// Defines how many rows and columns there are
const byte ROWS = 4;
const byte COLS = 4;

// Define name of keys
char hexaKeys[ROWS][COLS] = {
  {'1', '2', '3', '4'},
  {'5', '6', '7', '8'},
  {'A', 'B', 'C', 'D'},
  {'E', 'F', 'G', 'H'}
};

byte colPins[ROWS] = {9, 8, 7, 6};
byte rowPins[COLS] = {5, 4, 3, 2};

Keypad customKeypad = Keypad(makeKeymap(hexaKeys), colPins, rowPins, ROWS, COLS);

// Define notes
#define NOTE_C 523
#define NOTE_CS 554
#define NOTE_D 587
#define NOTE_E 659
#define NOTE_F 698
```



```
#define NOTE_FS 740
#define NOTE_G 784
#define NOTE_A 880
#define NOTE_B 988
#define NOTE_C2 1047
#define NOTE_CS2 1109
#define NOTE_D2 1175
#define NOTE_E2 1319
#define NOTE_F2 1397
#define NOTE_FS2 1480
#define NOTE_G2 1568

// Change this depending on your pin on the Arduino
const int buzzer = 11;

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

void loop() {
  int customKey = customKeypad.getKey();

  if (customKey == '1') {
    tone(buzzer, NOTE_C, 200);
  }
  if (customKey == '2') {
    tone(buzzer, NOTE_CS, 200);
  }
  if (customKey == '3') {
    tone(buzzer, NOTE_D, 200);
  }
  if (customKey == '4') {
    tone(buzzer, NOTE_E, 200);
  }
  if (customKey == '5') {
    tone(buzzer, NOTE_F, 200);
  }
  if (customKey == '6') {
    tone(buzzer, NOTE_FS, 200);
  }
  if (customKey == '7') {
    tone(buzzer, NOTE_G, 200);
  }
  if (customKey == '8') {
    tone(buzzer, NOTE_A, 200);
  }
  if (customKey == 'A') {
    tone(buzzer, NOTE_B, 200);
  }
  if (customKey == 'B') {
    tone(buzzer, NOTE_C2, 200);
  }
  if (customKey == 'C') {
    tone(buzzer, NOTE_CS2, 200);
  }
}
```

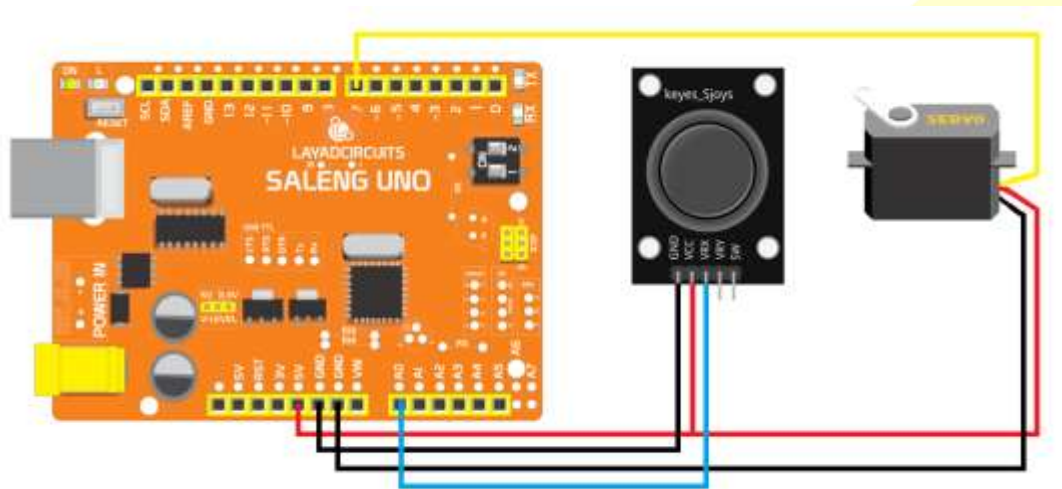
```

}
if (customKey == 'D') {
  tone(buzzer, NOTE_D2, 200);
}
if (customKey == 'E') {
  tone(buzzer, NOTE_E2, 200);
}
if (customKey == 'F') {
  tone(buzzer, NOTE_F2, 200);
}
if (customKey == 'G') {
  tone(buzzer, NOTE_FS2, 200);
}
if (customKey == 'H') {
  tone(buzzer, NOTE_G2, 200);
}
}
}
}

```

JOYSTICK CONTROLLED SERVO MOTOR

This project aims to control servo motor using Saleng UNO and Joystick. As seen on the circuit diagram, the jumper on the joystick is connected to the Vrx pin, therefore you can only control the servo motor by moving the joystick button on the upward and downward direction. You can also change the connection of the jumper to the Vry pin. And by that, you will observe that you can only control the servo motor by moving the joystick button on the sideways direction.



CODES:

www.layadcircuits.com

Layad Circuits Electronics Engineering Supplies & Services, B314 Lopez Bldg., Session Rd. cor. Assumption Rd., Baguio City, Philippines
General inquiries: info@layadcircuits.com Sales: sales@layadcircuits.com FB: facebook.com/layadcircuits Mobile: +639164428565

An IMPORTANT NOTICE: at the end of this guide addresses availability, warranty, changes, use in safety-critical applications, intellectual property matters and other important disclaimers.

```
#include <Servo.h>

Servo myservo;

int potpin = 0;
int val;

void setup() {
  myservo.attach(7);
}

void loop() {
  val = analogRead(potpin);
  val = map(val, 0, 1023, 0, 180);
  myservo.write(val);
  delay(15);
}
```

FREQUENTY ASKED QUESTIONS

Q: Is this kit the same as other kits?

A: There are a wide variety of Arduino Kits available in the market from numerous suppliers. This kit is very similar to some “Advanced Arduino Starter Kits” available in the open market, however, there may be minor differences like color of PCB, LED or perhaps in the part number used. This kit is our own version.

Q: Does this kit have everything?

A: No kit can include every component or module that can be used with the Arduino as there maybe millions. However, this kit contains some of the frequently used components and modules for Arduino projects.

Q: My Arduino Uno SMD or Saleng Uno board is not recognized by the Computer

A: this is usually because the USB driver has not been installed. [Download](#) from here then install. Reboot your computer after installation of the driver. You may also want to remove anything on pins D0 and D1.

Q: Does the difference in USB-serial circuit affect the programming or usage of the microcontroller?

A: No. The USB-serial circuit simply converts the UART of the microcontroller to USB for easy PC interfacing. Functionally, both circuits are the same.

Q: Despite my efforts, I am still having trouble with my project, what can I do?

A: You may avail of the Layad Circuits' Engineering services. Contact us for more information.

Q: I get errors while trying to upload code when something is connected to pins 0 and 1, what should I do?

A: This occurs since pins 0 and 1 are the UART pins used by the microcontroller to communicate with the PC. Circuits connected to this may have to be disconnected first before uploading code. In the Arduino, you would have to disconnect the external circuit manually.

Contributors:

E. Binay-an

K. Cariño

B. Aliangga

N. Callado

C. Cruz

K. Dela-Torre

IMPORTANT NOTICE

Layad Circuits Electronics Engineering Supplies & Services (Layad Circuits) reserves the right to make corrections, enhancements, improvements and other changes to its products, services and documentations, and to discontinue any product or service. Buyers or clients should obtain the latest relevant information before placing orders and should verify that such information is current and complete. Additional terms may apply to the use or sale of Layad Circuits products and services.

Reproduction of significant portions of Layad Circuits information in Layad Circuits datasheets or user guides is permissible only if reproduction is without alteration, displays the Layad Circuits logo and is accompanied by all associated warranties, conditions, limitations, and notices. Layad Circuits is not responsible or liable for such reproduced documentation. Information of third parties may be subject to additional restrictions. Resale of Layad Circuits products or services with statements different from or beyond the parameters stated by Layad Circuits for that product or service voids all express and any implied warranties for the associated Layad Circuits product or service. Layad Circuits is not responsible or liable for any such statements.

Buyers and others who are developing systems that incorporate Layad Circuits products (collectively, "Designers") understand and agree that Designers remain responsible for using their independent analysis, evaluation and judgment in designing their applications and that Designers have full and exclusive responsibility to assure the safety of Designers' applications and compliance of their applications (and of all Layad Circuits products used in or for Designers' applications) with all applicable regulations, laws and other applicable requirements. Designer represents that, with respect to their applications, Designer has all the necessary expertise to create and implement safeguards that (1) anticipate dangerous consequences of failures, (2) monitor failures and their consequences, and (3) lessen the likelihood of failures that might cause harm and take appropriate actions. Designer agrees that prior to using or distributing any applications that include Layad Circuits products, Designer will thoroughly test such applications and the functionality of such Layad Circuits products as used in such applications. Layad Circuits' provision of technical, application or other design advice, quality characterization, reliability data or other services or information, including, but not limited to, reference designs and materials relating to evaluation modules, (collectively, "Layad Circuits Resources") are intended to assist designers who are developing applications that incorporate Layad Circuits products; by downloading, accessing or using Layad Circuits Resources in any way, Designer (individually or, if Designer is acting on behalf of a company, Designer's company) agrees to use any particular Layad Circuits Resource solely for this purpose and subject to the terms of this Notice.

Layad Circuits' provision of Layad Circuits Resources does not expand or otherwise alter Layad Circuits' applicable published warranties or warranty disclaimers for Layad Circuits products, and no additional obligations or liabilities arise from Layad Circuits providing such Layad Circuits Resources.

Layad Circuits reserves the right to make corrections, enhancements, improvements and other changes to its Layad Circuits Resources. Layad Circuits has not conducted any testing other than that specifically described in the published documentation for a particular Layad Circuits Resource.

NO OTHER LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE TO ANY OTHER LAYAD CIRCUITS INTELLECTUAL PROPERTY RIGHT, AND NO LICENSE TO ANY TECHNOLOGY OR INTELLECTUAL PROPERTY RIGHT OF LAYAD CIRCUITS OR ANY THIRD PARTY IS GRANTED HEREIN, including but not limited to any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which Layad Circuits products or services are used. Information regarding or referencing third-party products or services does not constitute a license to use such products or services, or a warranty or endorsement thereof. Use of Layad Circuits Resources may require a license from a third party under the patents or other intellectual property of the third party, or a license from Layad Circuits under the patents or other intellectual property of Layad Circuits. LAYAD CIRCUITS RESOURCES ARE PROVIDED "AS IS" AND WITH ALL FAULTS. LAYAD CIRCUITS DISCLAIMS ALL OTHER WARRANTIES OR REPRESENTATIONS, EXPRESS OR IMPLIED, REGARDING RESOURCES OR USE THEREOF, INCLUDING BUT NOT LIMITED TO ACCURACY OR COMPLETENESS, TITLE, ANY EPIDEMIC FAILURE WARRANTY AND ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT OF ANY THIRD PARTY INTELLECTUAL PROPERTY RIGHTS. LAYAD CIRCUITS SHALL NOT BE LIABLE FOR AND SHALL NOT DEFEND OR INDEMNIFY DESIGNER AGAINST ANY CLAIM, INCLUDING BUT NOT LIMITED TO ANY INFRINGEMENT CLAIM THAT RELATES TO OR IS BASED ON ANY COMBINATION OF PRODUCTS EVEN IF DESCRIBED IN LAYAD CIRCUITS RESOURCES OR OTHERWISE. IN NO EVENT SHALL LAYAD CIRCUITS BE LIABLE FOR ANY ACTUAL, DIRECT, SPECIAL,

www.layadcircuits.com**Layad Circuits Electronics Engineering Supplies & Services, B314 Lopez Bldg., Session Rd. cor. Assumption Rd., Baguio City, Philippines**
General inquiries: info@layadcircuits.com Sales: sales@layadcircuits.com FB: facebook.com/layadcircuits Mobile: +639164428565

An IMPORTANT NOTICE: at the end of this guide addresses availability, warranty, changes, use in safety-critical applications, intellectual property matters and other important disclaimers.

COLLATERAL, INDIRECT, PUNITIVE, INCIDENTAL, CONSEQUENTIAL OR EXEMPLARY DAMAGES IN CONNECTION WITH OR ARISING OUT OF LAYAD CIRCUITS RESOURCES OR USE THEREOF, AND REGARDLESS OF WHETHER LAYAD CIRCUITS HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. Unless Layad Circuits has explicitly designated an individual product as meeting the requirements of a particular industry standard, Layad Circuits is not responsible for any failure to meet such industry standard requirements. Where Layad Circuits specifically promotes products as facilitating functional safety or as compliant with industry functional safety standards, such products are intended to help enable customers to design and create their own applications that meet applicable functional safety standards and requirements. Using products in an application does not by itself establish any safety features in the application. Designers must ensure compliance with safety-related requirements and standards applicable to their applications. Designer may NOT use any Layad Circuits products in life-critical applications. Life-critical medical equipment is medical equipment where failure of such equipment would cause serious bodily injury or death (e.g., life support, pacemakers, defibrillators, heart pumps, neurostimulators, and implantables). Designers agree that it has the necessary expertise to select the product with the appropriate qualification designation for their applications and that proper product selection is at Designers' own risk. Designers are solely responsible for compliance with all legal and regulatory requirements in connection with such selection. Designer will fully indemnify Layad Circuits and its representatives against any damages, costs, losses, and/or liabilities arising out of Designer's noncompliance with the terms and provisions of this Notice.

