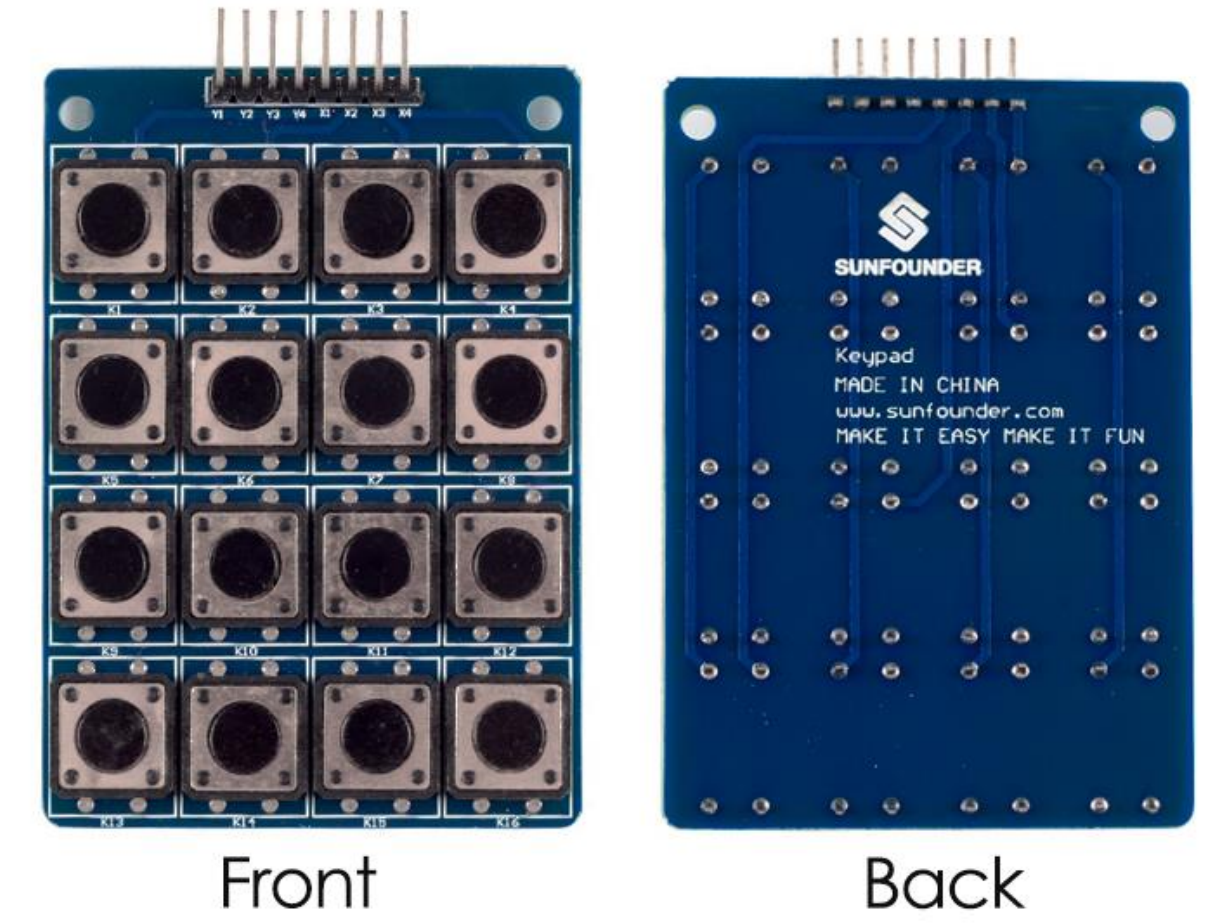
# <u>Lab no 03:</u> Password System Prototype Keypad and Digital-to-Analog Interfaces

The purpose of this Lab is to learn interfaces with a keypad and the Digital-to-Analog Converter (DAC). To do that, you are required to build a prototype for a password system using a keypad. You will use a keypad as input for the password, and the output will be validated, if the password is correct – the green LED turns on and if the password is wrong – the red LED and the buzzer turn on. Then you will learn how to build a simple Digital-to-Analog Converter (DAC) using a resistor ladder network to control alarm sound on a speaker.

## Parts: -

1. Introduction to the keypad (4x4)
   a) How the keypad works & how to scan them.
   b) Wiring 4 x 3 & 4 x 4 keypad with Arduino.
   c) Installing Keypad Library.
   d) Code for 4 x 4 keypad.
2. Password System prototype.
3. DAC with resistor ladder network.

## Part 1. Introduction to the keypad (4x4)

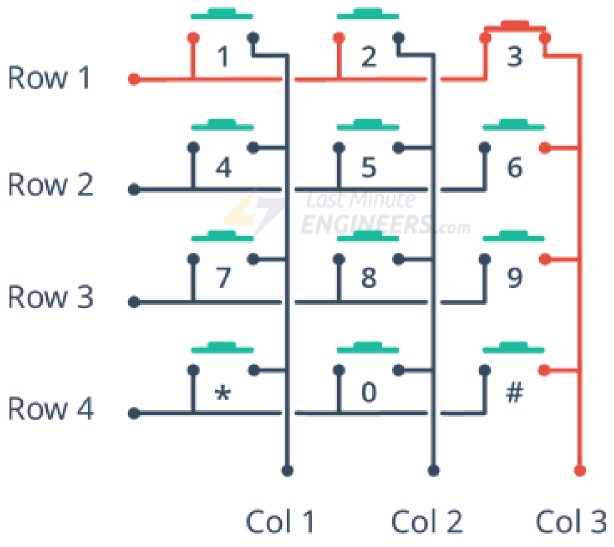


The 4*4 matrix keypad usually is used as input in a project. It has 16 keys in total, which means the same input values.

The 4*4 Matrix Keypad Module is a matrix non-encoded keypad consisting of 16 keys in parallel. The keys of each row and column are connected through the pins outside – pin Y1-Y4 as labeled beside control the rows, when X1-X4, the columns.

## Part 1. a) How keypad works & how to scan them

The working principle is very simple. Pressing a button short one of the row lines to one of the column lines, allows current to flow between them. For example, when key ‘4’ is pressed, column 1 and row 2 are shorted.

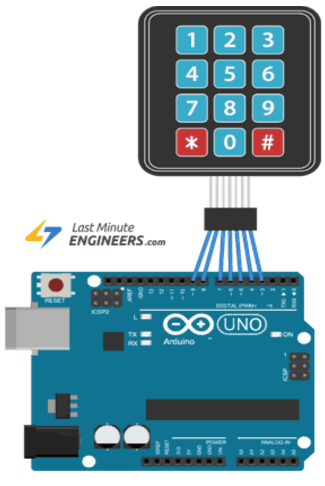


A microcontroller can scan these lines for a button-pressed state. To do this, it follows the below procedure.

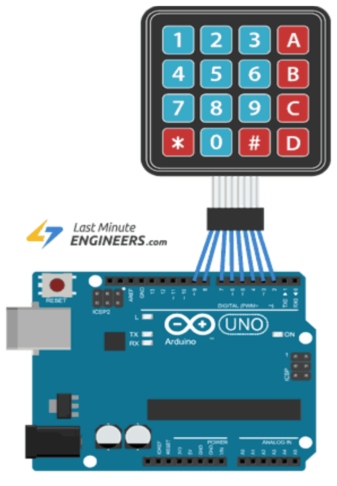
1. Microcontroller sets all the column and row lines to input.
2. Then, it picks a row and sets it HIGH.
3. After that, it checks the column lines one at a time.
4. If the column connection stays LOW, the button on the row has not been pressed.
5. If it goes HIGH, the microcontroller knows which row was set HIGH, and which column was detected HIGH when checked.
6. Finally, it knows which button was pressed that corresponds to the detected row & column.

## Part 1. b) Wiring 4x3 & 4x4 Membrane keypad with Arduino.

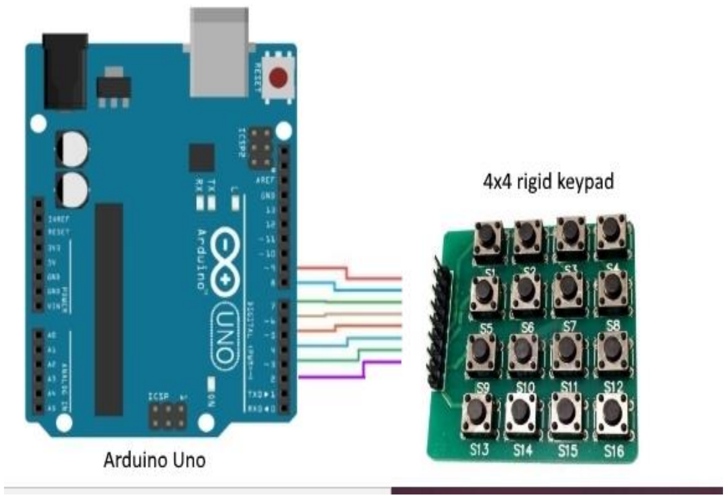
The connections are pretty straightforward. Start by connecting pin 1 of the keypad to digital pin 9 on Arduino. Now keep on connecting the pins leftwards like 2 with 8, 3 with 7, etc.



Wiring 4×3 Membrane Keypad with Arduino



Wiring 4×4 Membrane Keypad with Arduino



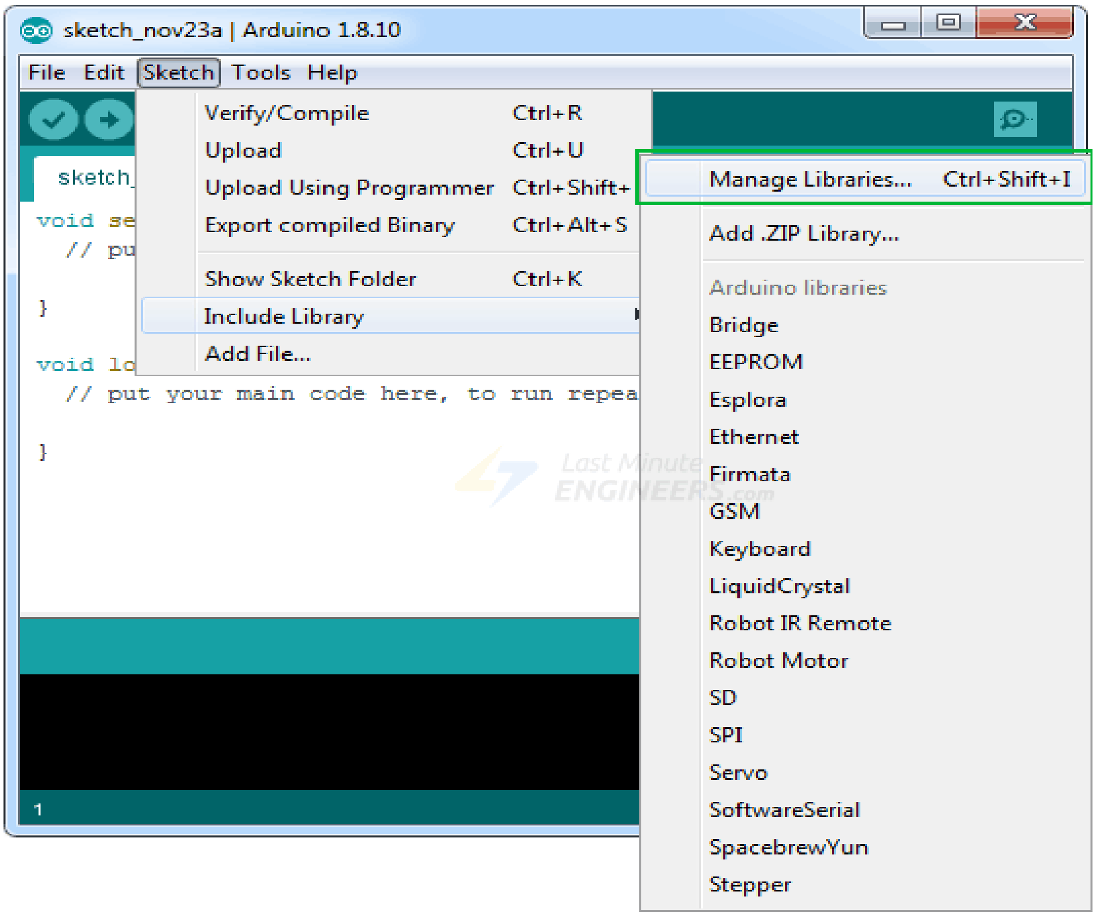
## Part 1. c) Installing Keypad Library

In order to determine which key was pressed, we need to continuously scan rows & columns. Fortunately, Keypad.h was written to hide away this unnecessary complexity so that we can issue simple commands to know which key was pressed.

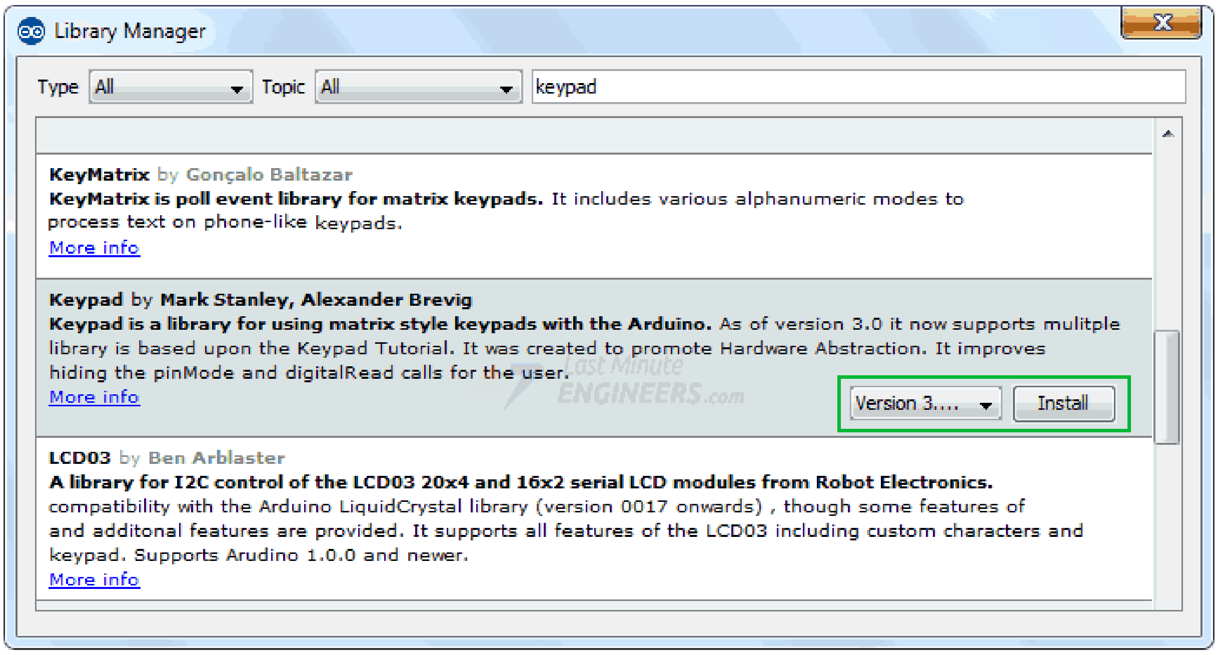
To install the library, navigate to the Sketch > Include Library > Manage Libraries…Wait for the Library Manager to download the libraries index and update the list of installed libraries.

Filter your search by typing ‘keypad’. There should be a couple entries. Look for Keypad by Mark Stanley, Alexander Brevig.

You have to scroll a little bit. Click on that entry, and then select Install



```
void se
  // pu

}

void lo
  // put your main code here, to run repea

}
```



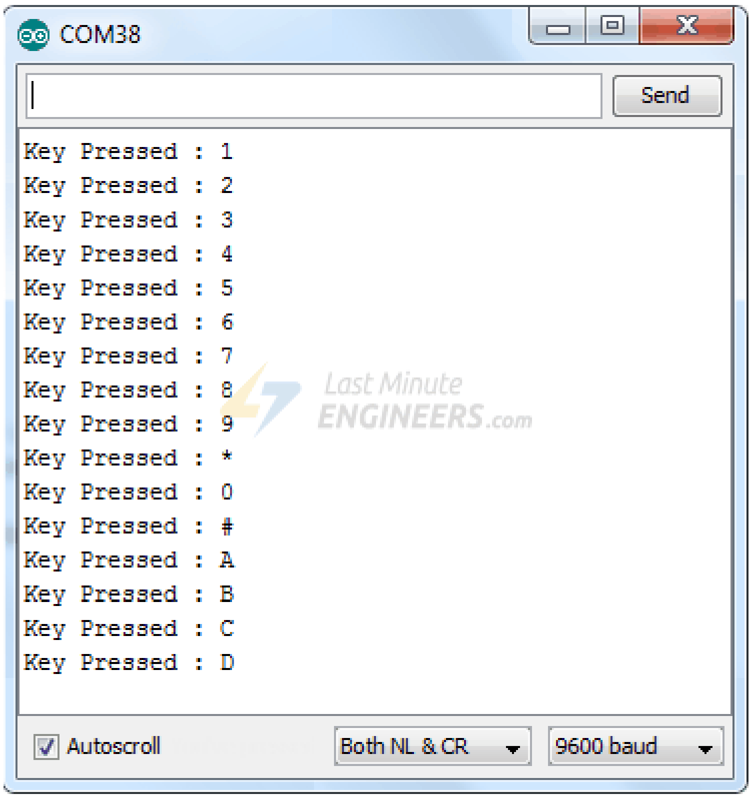
**KeyMatrix** by Gonçalo Baltazar
**KeyMatrix is poll event library for matrix keypads.** It includes various alphanumeric modes to process text on phone-like keypads.
More info

**Keypad** by **Mark Stanley, Alexander Brevig**
**Keypad is a library for using matrix style keypads with the Arduino.** As of version 3.0 it now supports mulitple library is based upon the Keypad Tutorial. It was created to promote Hardware Abstraction. It improves hiding the pinMode and digitalRead calls for the user.
More info

**LCD03** by Ben Arblaster
**A library for I2C control of the LCD03 20x4 and 16x2 serial LCD modules from Robot Electronics.** compatibility with the Arduino LiquidCrystal library (version 0017 onwards) , though some features of and additonal features are provided. It supports all features of the LCD03 including custom characters and keypad. Supports Arudino 1.0.0 and newer.
More info

## Part 1. d) Code for 4 x 4 keypad

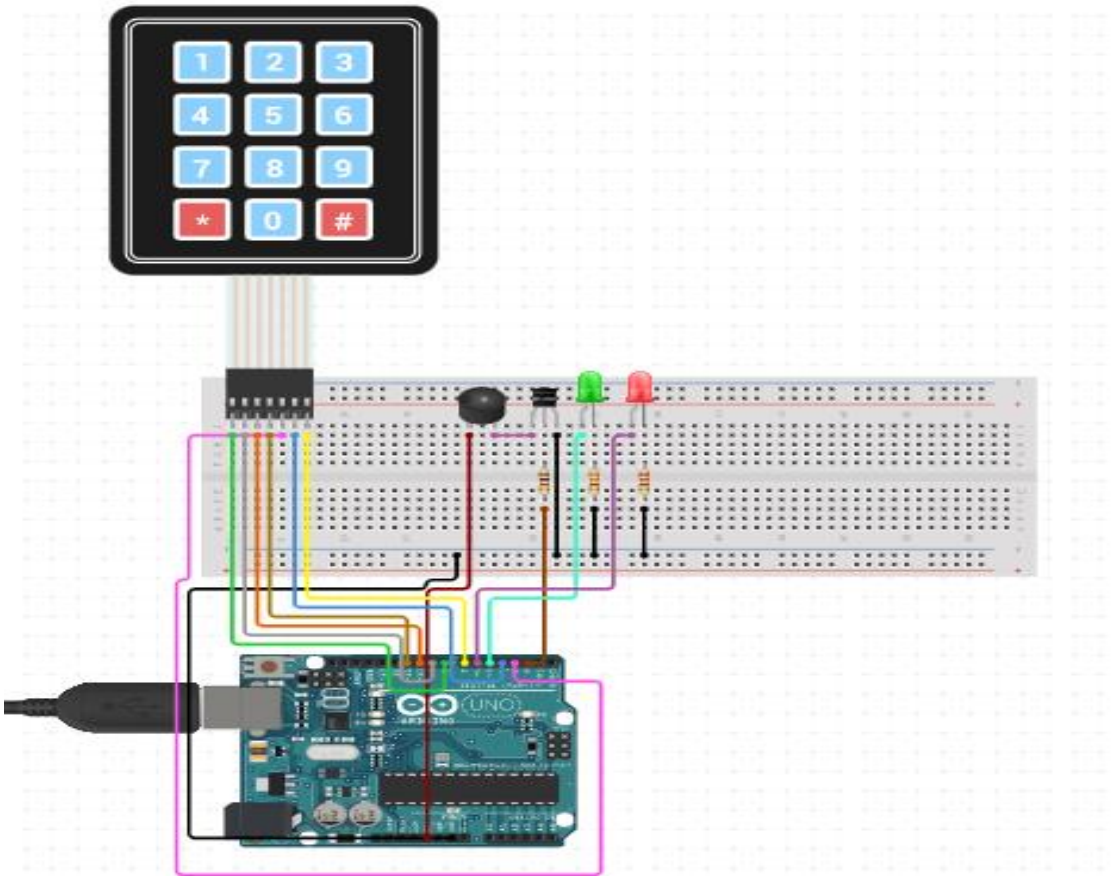
```
#include <Keypad.h>
const byte ROWS = 4; //four rows
const byte COLS = 4; //four columns

char keys[ROWS][COLS] = {
  {'1','2','3','A'},
  {'4','5','6','B'},
  {'7','8','9','C'},
  {'*','0','#','D'}
};

byte rowPins[ROWS] = {9, 8, 7, 6}; //connect to the row pinouts of the
keypad
byte colPins[COLS] = {5, 4, 3, 2}; //connect to the column pinouts of the
keypad

//Create an object of keypad
Keypad keypad = Keypad( makeKeymap(keys), rowPins, colPins, ROWS, COLS );

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

void loop(){
  char key = keypad.getKey();// Read the key

  // Print if key pressed
  if (key){
    Serial.print("Key Pressed : ");
    Serial.println(key);
  }
}
```



```
Key Pressed : 1
Key Pressed : 2
Key Pressed : 3
Key Pressed : 4
Key Pressed : 5
Key Pressed : 6
Key Pressed : 7
Key Pressed : 8
Key Pressed : 9
Key Pressed : *
Key Pressed : 0
Key Pressed : #
Key Pressed : A
Key Pressed : B
Key Pressed : C
Key Pressed : D
```

[✓ Autoscroll] Both NL & CR 9600 baud

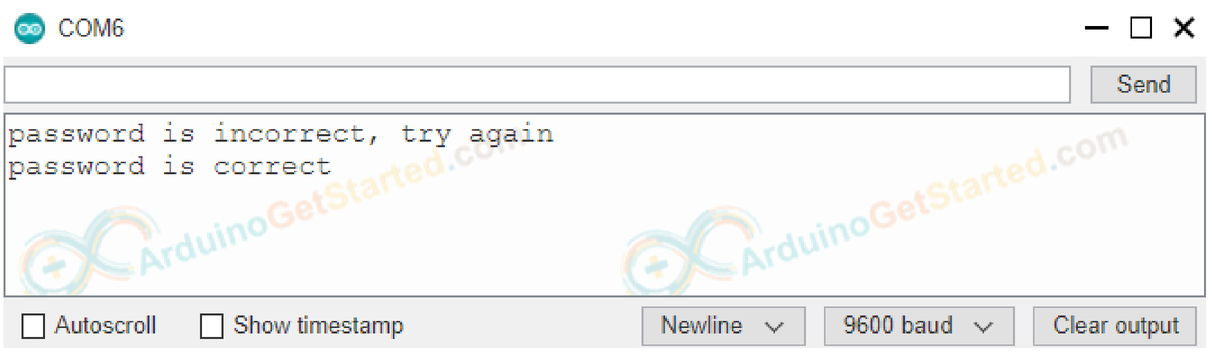
## Part 2. Password System prototype



```
#include <Keypad.h>
int buzzer = 13;
int red = 6;
int green = 7;
const int ROW_NUM = 4; //four rows
const int COLUMN_NUM = 4; //four columns

char keys[ROW_NUM][COLUMN_NUM] = {
  {'1','2','3', 'A'},
  {'4','5','6', 'B'},
  {'7','8','9', 'C'},
  {'*','0','#', 'D'}
};
```

```
byte pin_rows[ROW_NUM] = {9, 10, 11, 12}; //connect to the row pinouts of
the keypad
byte pin_column[COLUMN_NUM] = {5, 4, 3, 2}; //connect to the column pinouts
of the keypad

Keypad keypad = Keypad( makeKeymap(keys), pin_rows, pin_column, ROW_NUM,
COLUMN_NUM );

const String password = "1234"; // change your password here
String input_password;

void setup(){
  Serial.begin(9600);
  pinMode(buzzer, OUTPUT);
  pinMode(green, OUTPUT);
  pinMode(red, OUTPUT);
  input_password.reserve(32); // maximum input characters is 33, change if
needed
}

void loop(){
  char key = keypad.getKey();

  if (key){
    Serial.println(key);

    if(key == '*') {
      input_password = ""; // clear input password
      digitalWrite(green,LOW);
      digitalWrite(red,LOW);
      digitalWrite(buzzer,LOW);
    } else if(key == '#') {
      if(password == input_password) {
        Serial.println("password is correct");
       digitalWrite(green,HIGH);
        // DO YOUR WORK HERE
      } else {
        Serial.println("password is incorrect, try again");
         digitalWrite(buzzer,HIGH);
         digitalWrite(red,HIGH);

      }

      input_password = ""; // clear input password
    } else {
      input_password += key; // append new character to input password
string
    }
  }
}
```

- Run above code
- Open Serial Monitor
- Press “123456” keys and press “#”
- Press “1234” keys and press “#”
- See the result on Serial Monitor



```
password is incorrect, try again
password is correct
```

## Part3. DAC with ladder resistance

- Creating a digital to analog converter (DAC) 8-bit using resistor ladder (R-2R Ladder) and implemented it in Arduino Uno to create signals

- The number of levels are equal to two power the number of bits. In this project, 8 bits means there will be 256 levels. We can calculate the maximum voltage output by this equation.

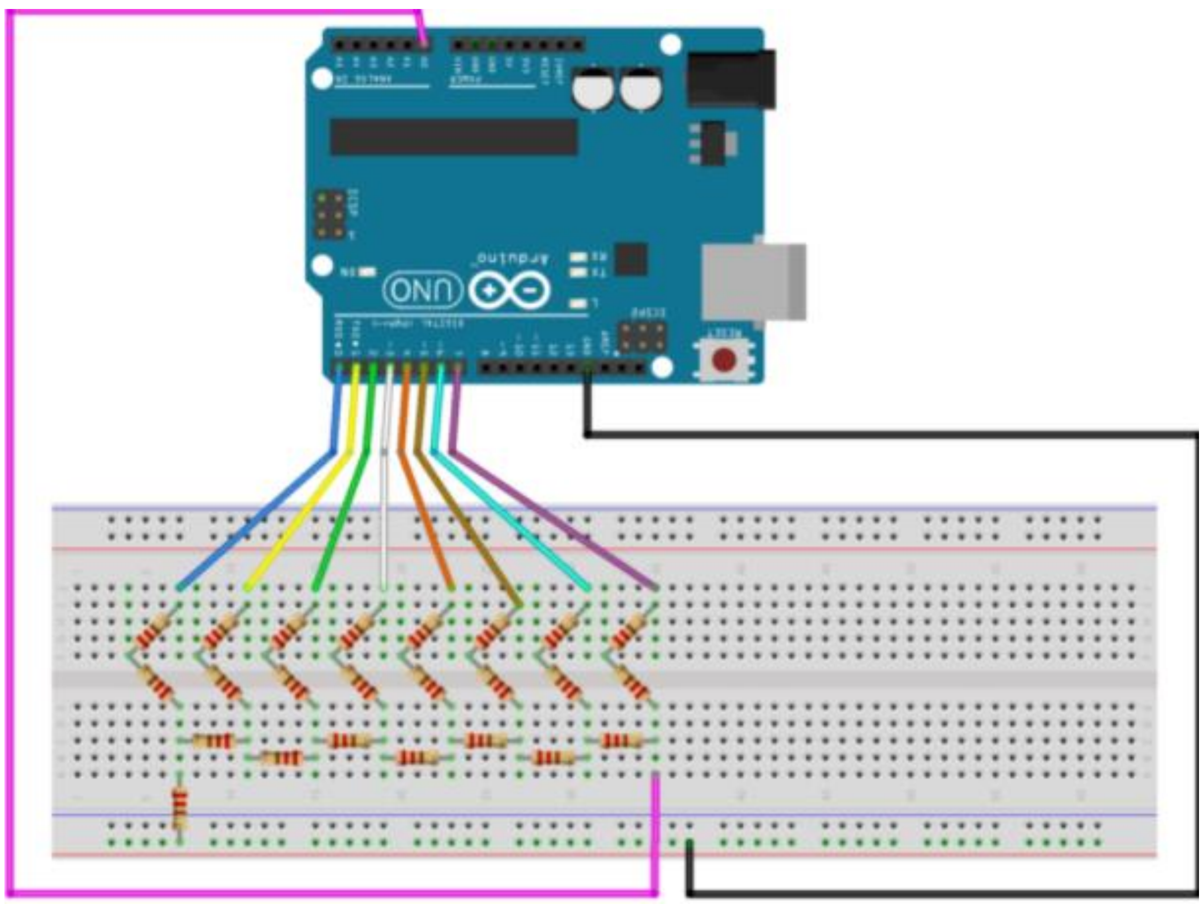
$$Vomax = \frac{Vhigh}{Number\ of\ Levels} * (Number\ of\ Levels - 1)$$

- and the voltage output equation for R-2R DAC itself is this:

$$V_{output} = \frac{V_{b0}}{2^n} + \frac{V_{b1}}{2^{n-1}} + \frac{V_{b2}}{2^{n-2}} + \cdots + \frac{V_{bn}}{2^1}$$

- In our project here, we made an 8-bit R-2R Ladder DAC, so n equals to 8 and the output voltage can be calculated as

$$V_{output} = \frac{V_{b0}}{2^8} + \frac{V_{b1}}{2^7} + \frac{V_{b2}}{2^6} + \frac{V_{b3}}{2^5} + \frac{V_{b4}}{2^4} + \frac{V_{b5}}{2^3} + \frac{V_{b6}}{2^2} + \frac{V_{b7}}{2^1}$$



```
// DAC R-2R Ladder 8 bit tutorials and schematic
void setup()
{
  Serial.begin(9600);
  DDRD = B11111111; // Port D at Arduino Uno (pin 0-7)
}
void loop()
{
  for ( int i = 0; i < 256; i++) {
/* looping the value of i from 0 to 255, creating a ramp wave from 0 to
255, it's more like stairs wave */
    PORTD = i;
/* the integer value of i is automatically translated to binary and assign
to port D, ex 3 is automatically transformed into 00000011 which means only
pin 0 and pin 1 is high (1) and the other pin is low (0)*/
    delay(10); // set the delay of the analog value
    Serial.println(analogRead(A0));
/*read the output of the DAC, translated into digital again by the ADC in
analog read arduino, but we can see the plot using serial plotter without
an oscilloscope. But oscilloscope has the best image to show than serial
plotter ADC Arduino Uno*/
  }
}
```