**Digital Code Lock using Arduino with LCD Display and User Defined Password**

Last Updated on [December 5, 2014](http://www.circuitstoday.com/advanced-digital-code-lock-using-arduino) by [praveen](http://www.circuitstoday.com/author/praveen) in [Arduino](http://www.circuitstoday.com/category/arduino) with [35 Comments](http://www.circuitstoday.com/advanced-digital-code-lock-using-arduino#comments)

We have published a **[digital code lock using arduino](http://www.circuitstoday.com/digital-code-lock-using-arduino%22%20%5Ct%20%22_blank)** some weeks before. This one is a little different. The earlier version was based on a **defined password,** where the user can not change it. Moreover there was no **LCD display** interfaced with the project to output lock status. This project is a much improved version of the same digital code lock which comes with a ***user defined password*** and LCD display. The user will be prompted to ***set a password***at installation. This password inputted at installation will continue to serve the lock until it is changed. The user can change the current password with a single key press. The program will check for current password and allows the user to change password only if the the current password is input correctly.



**Required Knowledge**

You should learn two important device interfacing concepts before attempting this project. The very first one is to **[interface hex keypad with Arduino](http://www.circuitstoday.com/interfacing-hex-keypad-to-arduino)**. The second one is to **[interface LCD with Arduino](http://www.circuitstoday.com/interfacing-lcd-to-arduino)**. Once you understand the concepts behind interfacing LCD module and interfacing hex keypad, its just a matter of adding few lines of code to build the Digital Code Lock. I recommend to read the **[simple digital code lock using arduino](http://www.circuitstoday.com/digital-code-lock-using-arduino)** as well to gain insights into basics of a code lock.

**Note:-** In this program, I have reused the code developed for interfacing hex keypad with arduino. There are two versions of the program in the interfacing tutorial. I used version 2.

**About the Program**

I will be explaining important points about the program only. As mentioned before, you are supposed to know the codes of interfacing hex keypad and lcd module.

**Usage of Device**

**Installation –** You will be asked to input **5** digits as password  at the ***initial boot/reset*** of the device. The first 5 digits you input at installation will be saved as your **SET PASSWORD.**  The device will go LOCKED after setting PASSWORD.

**Key A –** for unlocking the device. Input correct password and **press A** for Unlocking.

**Key B –** for locking any time. Just **press B** and you will see the LOCKED message.

**Key C –** for changing the password. Input the correct password and **Press C**. You will see message asking to ***ENTER NEW PASSWORD***. Enter **5** digits as password. The first 5 digits you enter will be SAVED as NEW PASSWORD.

**Exceptions –** You can not use keys ‘A’, ‘B’ and ‘C’ inside the password combination. These 3 keys are control keys of the device with specific functions. The program checks for these key presses (at the password setting time – you may see the SetPassword() function and look the ***condition to check for invalid key press***) and identifies them as **Invalid Keys**. You will have to ***input 5 new digits*** as password after an Invalid Key press.

**Important Variables and Arrays**

**pass[6]** – is the array used to save and hold the user defined password.

**check[6]** – is the array used to collect & hold user input. This user input data (in **check[]** array) is compared with **pass[]** array to authenticate password.

**entry** – is the variable used to identify initial entry point of the program. User is asked to SET a 5 Digit Password at installation of Lock. Hence we need a variable to ***identify entry***and ***loop 5 times*** to collect **5** digits and save them to **pass[]** array. The same variable is later made use of to Change Password. When the key for Changing Password (here ‘C’) is pressed, this variable is simply assigned a zero value (the initial state of variable). This forces the program control to re enter the Password Setting Loop of the program.

**key\_id** – is the variable used to identify a key press and perform some actions in the program (that should happen only on a key press). By default this variable is set ***zero*** initial value. Whenever a key is pressed in key pad, this variable will be assigned a **value =1.** You may check the **keyscan()** function to see this. This simple trick helps to identify a key press and perform various actions on that key press (based on the value of key press). This variable **is set to zero** at different points in the program (to prevent the value 1 in **key\_id** variable being identified as a false key press). You may check them as well.

**Note:-** **col\_scan** – is the actual variable that gets activated to a **LOW** on key press (hence helps in identifying key press). But this variable is actually a part of the key pad interfacing program (version 2).

**count –** is the variable used to iterate the index of **check[count]** ( user input array ). count variable is initialized to 1. Each user input will be saved to **check[]** array in order of the increment of count variable.

**temp\_press –** is a temporary variable to hold the value of key press. The value of key press is assigned to temp\_press variable as a return result of the **keypress()** function. keypress() is the function defined to identify value of key press.

**lcd\_count –** is a simple counter variable used to iterate the column position of LCD module. This variable helps to display user input data successively in row 2 of LCD module.

**i,j,flag** – are just dummy variables used in the program. i,j are used as counter variables inside for loop. ***flag*** is used to hold status of **checkPassword()** subroutine (the function used to compare user input data and the SET password ). A decision is made based on the value inside flag variable.

**Subroutines used in the Program**

**SetPassword()** – is the subroutine used to SET user defined password. This subroutine is very dependent on the “**Password Setting Loop**” written inside the main program. This password setting loop will be iterated at installation of the device (that is at the boot or reset of the device) for first **5 key presses**. This first 5 key press will be used to SET the Password. These key presses will be saved to **pass[]** array.  As mentioned earlier, ***entry*** is the variable used to iterate the loop **5** times. **key\_id** is the variable used identify key press.

**Note:-** The same “**Password Setting Loop**” is made use of  for Changing the Password as well. When key **‘C’** is pressed, the current password is checked for. If the input password is matching with current SET password, then **entry** variable will be assigned to ***zero*** value. This will simply transfer the control of the program to ENTER the Password Setting Loop again.

**keyscan()** – is the subroutine to scan keypad for a key press. This subroutine is basically same as the version 2 code of interfacing hex keypad to arduino. I have added some lines of code needed for this code lock. Apart from that, the lines of code in this subroutine is same as that of interfacing keypad. keyscan() subroutine scans for a key press (when ever the function is called from Main program or from other sub routines like SetPassword()) and identifies the row and column of the pressed key. If key **‘1’** is pressed, keyscan identifies that key at **row 1** and **column 1** is pressed. Similarly if key ‘6’ is pressed, the keyscan identifies a key is pressed at row 2 and column 3. When ever a key is pressed, another subroutine named **keypress()** is invoked within the keyscan() routine. This keypress() routine is used identify the value of key press (say ‘1’, ‘2’, ‘3’ or ‘A’, ‘C’ or ‘D’ etc)

**keypress()** – as mentioned above is the subroutine to identify value of key press. The **keyscan()** routine identifies which row and column of key pad is pressed. This ***row and column*** number is passed to keypress() routine as parameters (using variable values of i and j ).

**checkPassword()** – is the subroutine to check user input password against the SET User Defined Password. The user input data (password to cross check) is collected in the **check[]** array. This is compared against the SET Password inside **pass[]** array. A for loop is used for comparing. If each digit inside the arrays match, flag variable will remain zero. If any mismatch occurs, the flag will be set to 1 and loop will break.

So that’s all you need to know about the program. Read the program below and if you have any doubt, just drop a comment.

**Program**

#include<LiquidCrystal.h>

LiquidCrystal lcd(7,6,5,4,3,2);

int row[]={A1,A0,8,9};// Defining row pins of keypad connected to Arduino pins

int col[]={10,11,12,13};//Defining column pins of keypad connected to Arduino

int i,j,lcd\_count,count=1,key\_id=0,flag,entry=0;// See About the Program

int col\_scan;// Variable to identify a key press

char temp\_press; // Variable to hold value of key press

char check[6],pass[6]; // See About the Program

void setup()

{

lcd.begin(16,2);

for(i=0;i<=3;i++)

{

pinMode(row[i],OUTPUT);

pinMode(col[i],INPUT);

digitalWrite(col[i],HIGH);

}

lcd.print("SET 5 Digit PASS");

}

/\* Main Program Begins \*/

void loop()

{

 while(entry<=4)// Password Setting Loop begins

{

SetPassword();

}

// Password Setting Loop Ends

key\_id=0;

keyscan(); // Scan for a Key Press

/\* Actions on Key Press begins \*/

if(key\_id==1) // Condition to Check Key is Pressed

{

 check[count]=temp\_press;

count++;

/\* Condition to Unlock Begins\*/

if(temp\_press=='A')

{

  checkPassword();

if(flag==0)

{

lcd.setCursor(0,0);

lcd.print("UNLOCKED");

}else{

lcd.setCursor(0,0);

lcd.print("WRONG PASSWORD");

delay(200);

lcd.clear();

lcd.print("LOCKED");

}

count=1; // Resetting the counter variable

}

/\* Condition to Unlock Ends\*/

/\* Condition to Change Password Begins \*/

else if(temp\_press=='C')

{

checkPassword();

if(flag==0)

{

lcd.setCursor(0,0);

lcd.print("ENTER NEW PASS");

key\_id=0;

entry=0;

}else{

lcd.setCursor(0,0);

lcd.print("WRONG PASSWORD");

}

count=1; // Resetting the counter variable

}

/\* Condition to Change Password Ends \*/

/\* Condition to LOCK Begins\*/

else if(temp\_press=='B')

{

lcd.setCursor(0,0);

lcd.print("LOCKED");

count=1; // Resetting the counter variable

}

/\* Condition to LOCK Ends\*/

}

/\* Actions on Key Press Ends\*/

}

/\* Main Program Ends \*/

void SetPassword() // Subroutine to SET User Defined Password

{

keyscan();

if(key\_id==1)

{

if(temp\_press=='A'||temp\_press=='C'||temp\_press=='B') // Condition to Check for an Invalid Keypress

{

lcd.setCursor(0,0);

lcd.print("INVALID KEYS");

entry=0;

}

else

{

  pass[entry]=temp\_press;

}

}

key\_id=0;

if(entry==5)

{

lcd.clear();

lcd.setCursor(0,0);

lcd.print("PASSWORD SET & LOCKED");

}} // Subroutine to SET Password ENDS

char keyscan()// Subroutine to Scan Keypad for a Keypress

{

for(i=0; i<=3; i++)

{

digitalWrite(row[0],HIGH);

digitalWrite(row[1],HIGH);

digitalWrite(row[2],HIGH);

digitalWrite(row[3],HIGH);

digitalWrite(row[i],LOW);

for(j=0; j<=3; j++)

{

col\_scan=digitalRead(col[j]);

if(col\_scan==LOW)

{

  key\_id=1;

  entry++;

temp\_press=keypress(i,j);

lcd.setCursor(lcd\_count++,1);

lcd.print(temp\_press);

if(temp\_press=='A'||temp\_press=='C'||temp\_press=='B')

{

lcd\_count=0;

lcd.clear();

}

delay(300);

break;

}}

}}// Subroutine to Scan Keypress Ends

char keypress(int i, int j) // Subroutine to Identify the value of Key pressed

{

if(i==0&&j==0)

{

return('1');

}

if(i==0&&j==1)

{

return('2');

}

if(i==0&&j==2)

{

return('3');

}

if(i==0&&j==3)

{

return('A');

}

if(i==1&&j==0)

{

return('4');

}

if(i==1&&j==1)

{

return('5');

}

if(i==1&&j==2)

{

return('6');

}

if(i==1&&j==3)

{

return('B');

}

if(i==2&&j==0)

{

return('7');

}

if(i==2&&j==1)

{

return('8');

}

if(i==2&&j==2)

{

return('9');

}

if(i==2&&j==3)

{

return('C');

}

if(i==3&&j==0)

{

return('\*');

}

if(i==3&&j==1)

{

return('0');

}

if(i==3&&j==2)

{

return('#');

}

if(i==3&&j==3)

{

return('D');

}

} // Subroutine to identify Keypress Ends

void checkPassword() // Subroutine to Check User Input Data with SET Password

{

flag=0;

for(i=1;i<=5&&flag==0;i++)

{

if(check[i]==pass[i])

{

flag=0;

}

else

{flag=1;

}}} // Subroutine to check password ends