نام و نام خانوادگی: **امیرحسین علی‌بخشی**
شماره‌ی دانشجویی: **۹۷۳۱۰۹۶**

آزمایش شماره ۳

استاد: **آقای معصوم‌زاده**
درس: **آزمایشگاه ریزپردازنده و زبان اسمبلی**

هدف آزمایش

ساخت یک عدد ماشین حساب ساده که برای اعداد صحیح کار میکند و قادر است اعمال زیر را انجام دهد:

- جمع
- تفریق
- ضرب
- تقسیم

بخش های مورد نیاز

برد Arduino

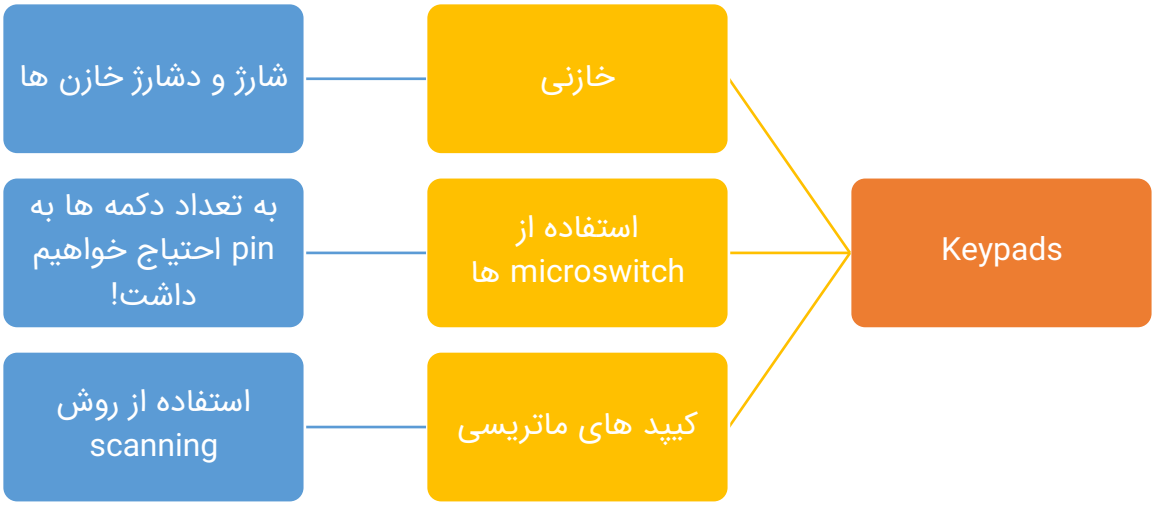
در گزارش شماره ۱ در مورد این برد توضیحات داده شده موجود میباشد.

یک عدد LCD ۱۶×۲

در گزارش شماره ۲ مشخصات فنی و همچنین برخی از توابع مربوط به کتابخانه ی LiquidCrystal ذکر شده است.

یک keypad ۴×۴

انواع keypad



ما در این آزمایش از نوع سوم استفاده خواهیم کرد.

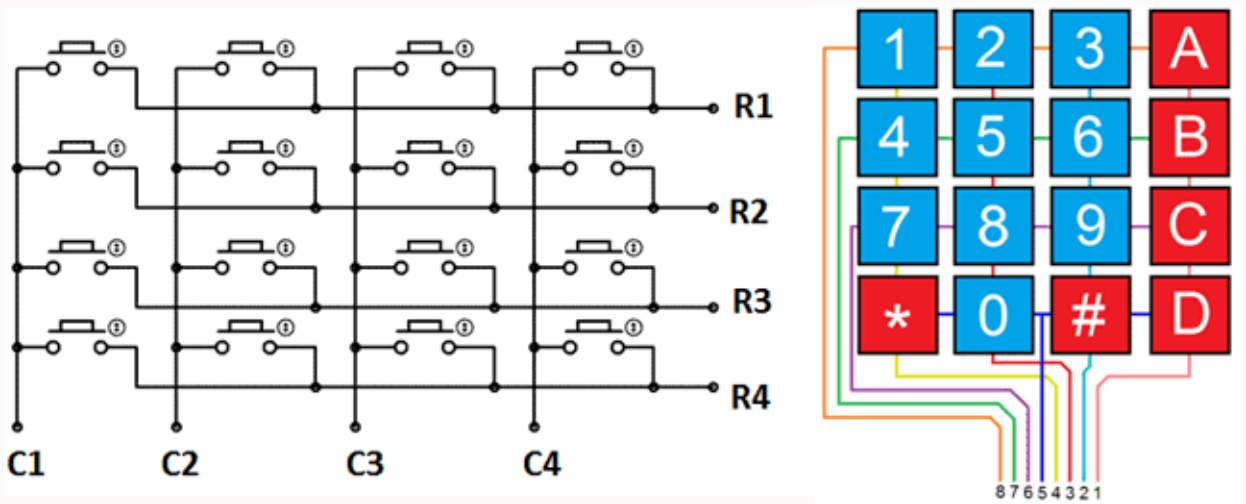
نام و نام خانوادگی: **امیرحسین علی‌بخشی**
شماره‌ی دانشجویی: **۹۷۳۱۰۹۶**

آزمایش شماره ۳

استاد: **آقای معصوم‌زاده**
درس: **آزمایشگاه ریزپردازنده و زبان اسمبلی**

ویژگی ها

در این گونه، تعداد port هایی که از برد اشغال میشوند، برابر تعداد مجموع سطر و ستون های ماتریس مورد نظر خواهد بود. محبوب ترین نوع های کیپد های ماتریسی، کیپد های ۴×۴ و ۴×۳ هستند که به ترتیب به ۸ و ۷ پورت برای برنامه ریزی نیاز دارند. در این آزمایش ما با کیپد ۴×۴ سروکار خواهیم داشت که طرح آن در شکل زیر قابل مشاهده میباشد.



برای استفاده کردن و کنترل کردن این کیپد توسط برد، نیاز داریم از کتابخانه‌ی Keypad استفاده کنیم.

در این کیپد ها دکمه ها در حالت pull-up قرار خواهند داشت. به این معنی که در حالت پیشفرض مقدار خروجی ها را HIGH در نظر میگیریم و در صورت فشرده شدن کلید، مقدار LOW را دریافت خواهیم کرد.

از همین خاصیت میتوان استفاده کرد و مفهومی به نام scanning را معرفی کرد. در این روش برای اینکه بتوان دکمه هایی از کیپد را که فشرده شده اند را شناسایی کرد، در ابتدای هر loop، مقدار سطر ها را یکی پس از دیگری LOW میکنیم و مقادیر ستونی متناظر با سطر مورد نظر را بررسی میکنیم. اگر مقدار برابر با LOW بود یعنی دکمه ای که در سطر و ستون گفته شده قرار دارد فشرده شده است. بدین ترتیب میتوان تمام دکمه هایی که فشرده شده اند را شناسایی کرد.

(تمامی این کارها در کتابخانه انجام میشوند و لازم نیست ما کار خاصی در این مورد انجام دهیم.)

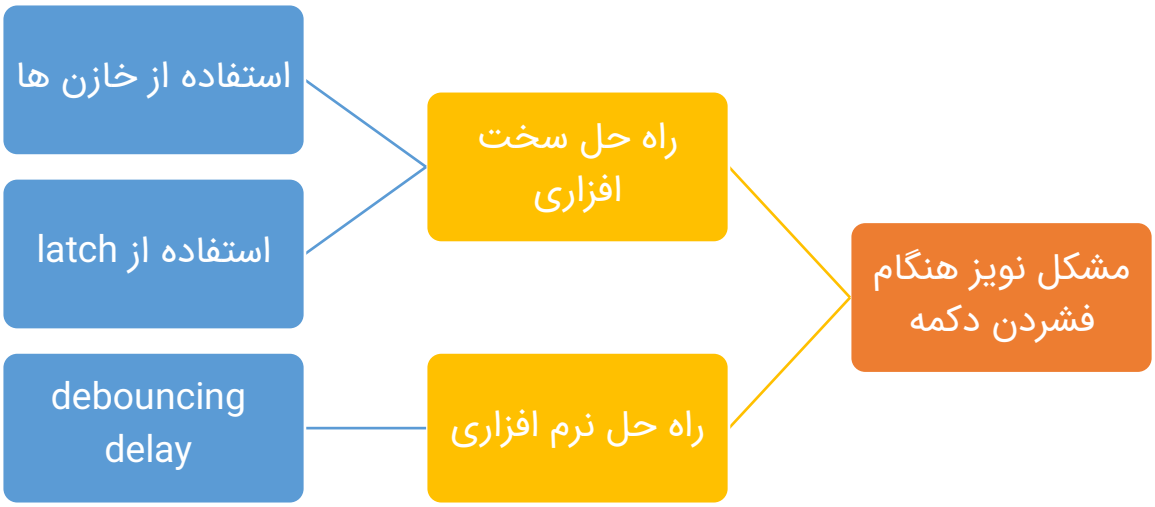
مشکلات پیش رو

از مشکلات این کیپد ها، وجود نویز در هنگام فشردن دکمه ها میباشد. اگر در دنیای واقعی خیلی دقیق به سیگنال های مربوط به فشرده شدن دکمه ها نگاه کنیم میبینیم که انگار دکمه با سرعت زیاد قطع و وصل میشود تا به حالت تعادل برسد. برای این روش راه های گوناگونی وجود دارد.

نام و نام خانوادگی: **امیرحسین علی‌بخشی**
شماره‌ی دانشجویی: **۹۷۳۱۰۹۶**

آزمایش شماره ۳

استاد: **آقای معصوم‌زاده**
درس: **آزمایشگاه ریزپردازنده و زبان اسمبلی**



در مورد روش debouncing delay در قسمت توابع توضیحات بیشتری داده خواهد شد.

توابع کتابخانه keypad

```
Keypad(makeKeymap(userKeymap), row[], col[], rows, cols)
```

توسط این constructor میتوان یک object کیپد را تولید کرد. هر یک از آرگمان های آن به شرح زیر میباشد:

آرگمان	توضیحات
`makeKeymap(userKeymap)`	خود ما میتوانیم مقادیر روی دکمه های کیپد را در قالب آرایه ای دو بعدی به تابع آماده‌ی makeKeymap تحویل بدهیم تا بتوانیم در طول برنامه به کمک نامگذاری های خود از دکمه ها استفاده کنیم
`row[]`	آرایه ای از اعداد که شماره PORT هایی هستند که به قسمت سطر های کیپد را به برد وصل کرده اند
`col[]`	آرایه ای از اعداد که شماره PORT هایی هستند که به قسمت ستون های کیپد را به برد وصل کرده اند
`rows`	تعداد سطر ها
`cols`	تعداد ستون ها

همچنین تابع begin نیز وجود دارد که کار آن تغییر در ساختار کیپدی است که از قبل تولید شده است.

```
getKey()
```

مقدار دکمه ی فشرده شده را مشخص میکند. این تابع حالت non-blocking دارد، یعنی اینکه در صورت نبودن جواب، مقداری برای آن کلید تعریف نمیشود و برنامه بدون معطلی به خط بعدی میرود.

getKeys()

بیان میکند که آیا چند دکمه همزمان فشـــرده اند یا خیر. ولی مشـــخص نمیکند کدام یک از این دکمه ها هستند که فشرده شده اند.

برای فهمیدن این موضوع که کدام دکمه ها فشرده شده اند باید از توابع isPressed() و findInList() استفاده کرد.

waitForKey()

نسخه ی blocking تابع getkey که منتظر میماند تا حتما یک ورودی وارد شود.

getState()

وضعیت کلید ها را به کمک این تابع میتوان مشخص کرد. وضعیت هر کلید میتواند ۴ مقدار میتواند داشته باشد:

- **IDLE**: زمانی که دکمه فشرده نشده
- **PRESSED**: زمانی که دکمه به تازگی فشرده شده است.
- **RELEASED**: زمانی که دکمه تازه رها شده است.
- **HOLD**: زمانی که دکمه برای مدتی نگه داشته شده است؛ میتوان مدت زمانی را که پس از گذشت آن میتوان ادعا کرد دکمه در حالت HOLD قرار گرفته را توسط تابعی به نام setHoldTime() تعیین کرد.

keyStateChanged()

اگر وضعیت کلیدی تغییر کرد مقدار true را به ما برمیگرداند.

setDebounceTime ()

آن راه حل نرم افزاری برای نویز ها که قبلا از آن صـحبت کردیم اسـتفاده از همین تابع اسـت. این تابع کمک میکند تا اگر فاصله بین دو بار فشرده شدن یک کلید از مقدار مشخصی کمتر بود به این تغییر توجهی نشود.

قسمت کد

```
#include <LiquidCrystal.h>
#include <Keypad.h>
#define RS 16
#define EN 17
#define D4 18
#define D5 19
#define D6 20
#define D7 21
#define DELAY_PERIOD 500
const byte ROWS = 4; // Four rows
const byte COLS = 4; //Four columns
```

نام و نام خانوادگی: **امیرحسین علی‌بخشی**
شماره‌ی دانشجویی: **۹۷۳۱۰۹۶**

آزمایش شماره ۳

استاد: **آقای معصوم‌زاده**
درس: **آزمایشگاه ریزپردازنده و زبان اسمبلی**

```
char keys[ROWS][COLS] = {
  {'7','8','9','D'},
  {'4','5','6','C'},
  {'1','2','3','B'},
  {'R','0','=','A'}
};
byte rowPins[ROWS] = { 7, 6, 5, 4 };// Connect keypad ROW0, ROW1, ROW2 and ROW3 t
o these Arduino pins.
byte colPins[COLS] = { 3, 2, 1, 0 }; // Connect keypad COL0, COL1 and COL2 to the
se Arduino pins.
Keypad kpd = Keypad( makeKeymap(keys), rowPins, colPins, ROWS, COLS ); //  Create
 the Keypad
LiquidCrystal lcd(RS, EN, D4, D5, D6, D7);
long Num1,Num2,Number;
char key,action;
boolean result = false;
void setup() {
  lcd.begin(16, 2); //We are using a 16*2 LCD display
  lcd.print("Calculator"); //Display a intro message
  lcd.setCursor(0, 1);   // set the cursor to column 0, line 1
  lcd.print("Lab3"); //Display a intro message
  delay(2000); //Wait for display to show info
  lcd.clear(); //Then clean it
}

void loop() {
  key = kpd.getKey(); //storing pressed key value in a char
  if (key!=NO_KEY)
    DetectButtons();
  if (result==true)
    CalculateResult();
  DisplayResult();
}
void DetectButtons(){
     lcd.clear(); //Then clean it
    if (key=='R') //If cancel Button is pressed
    {Serial.println ("Button Cancel"); Number=Num1=Num2=0; result=false;}
    if (key == '1') //If Button 1 is pressed
    {Serial.println ("Button 1");
    if (Number==0)
    Number=1;
    else
    Number = (Number*10) + 1; //Pressed twice
    }
    .
```

نام و نام خانوادگی: **امیرحسین علی‌بخشی**
شماره‌ی دانشجویی: **۹۷۳۱۰۹۶**

آزمایش شماره ۳

استاد: **آقای معصوم‌زاده**
درس: **آزمایشگاه ریزپردازنده و زبان اسمبلی**

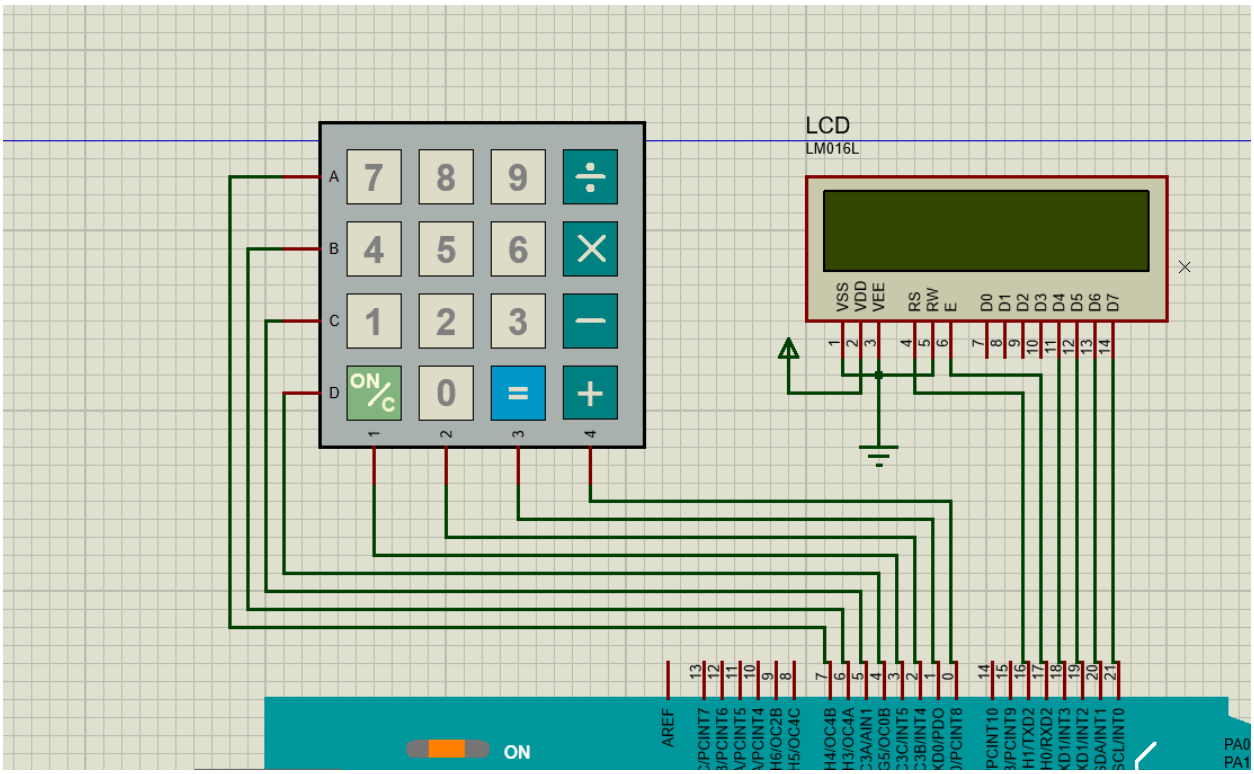
```
    .
    .
    if (key == '9')
    {Serial.println ("Button 9");
    if (Number==0)
    Number=9;
    else
    Number = (Number*10) + 9; //Pressed twice
    }
    if (key == 'A' || key == 'B' || key == 'C' || key == 'D') {
    Num1 = Number;
    Number =0;
    if (key == 'A')
     action = '+';
     if (key == 'B')
      action = '-';
     if (key == 'C')
      action = '*';
     if (key == 'D')
      action = '/';
    delay(100);
  }
}
void CalculateResult(){
  if (action=='+')
    Number = Num1+Num2;
  if (action=='-')
    Number = Num1-Num2;
  if (action=='*')
    Number = Num1*Num2;
  if (action=='/')
    Number = Num1/Num2;
}
void DisplayResult(){
  lcd.setCursor(0, 0);   // set the cursor to column 0, line 1
  lcd.print(Num1);
  lcd.print(action);
  lcd.print(Num2);
  if (result==true)
  {lcd.print(" ="); lcd.print(Number);} //Display the result
  lcd.setCursor(0, 1);   // set the cursor to column 0, line 1
  lcd.print(Number); //Display the result
}
```

نام و نام خانوادگی: **امیرحسین علی‌بخشی**
شماره‌ی دانشجویی: **۹۷۳۱۰۹۶**

آزمایش شماره ۳

استاد: **آقای معصوم‌زاده**
درس: **آزمایشگاه ریزپردازنده و زبان اسمبلی**

مدار



نمونه خروجی