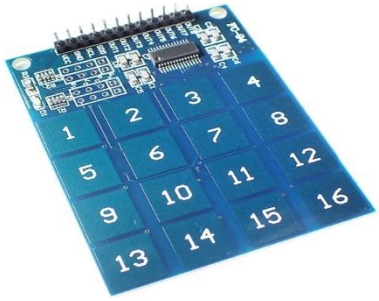
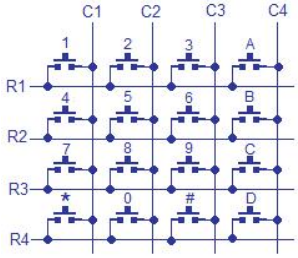
به نام خدا

- کد های مورد نیاز برای برنامه نویسی:
- انواع keypad ماتریسی و چگونگی کارکرد آن‌ها:

صفحه کلیدها(keypad) انواع مختلفی دارد مثل matrix keypad (شکل ۲)و capacitor touch keypad (شکل ۳) و .... matrix keypad ها ساختاری شبیه میکروسوییچ ها دارن با این فرق که آرایشی ماتریسی دارن(شکل ۱) برای کم‌کردن تعداد پین‌ها برای متصل کردن به برد و جاهایی که کاربرد داره و ستون‌ها و سطرها در صورت فشار دادن کلید بهم متصل می شوند وپایه های ستون که در ولتاژ high قرار داشته به low تبدیل می‌شود و برد آردوینو از همین استفاده می‌کند برای تشخیص فشار دادن کلید به این صورت که همه پین‌های یک ردیف را high می‌کند به جز یکی و اگه در همین حال پین ستونی هم low شود خود کتابخانه Keypad تشخیص می‌دهد و اینگونه می‌فهمیم که کدام کلید فشار داده شده است



شکل ۳



شکل ۱

شکل ۲

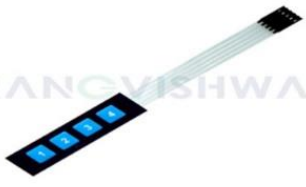
Matrix keypad های استاندارد هم به ۳ درسته تقسیم می شود که فقط در تعداد سطر و ستون ها تفاوت دارن که درزیر آورده شده است:


شکل ٥

4x4 Universal Matrix Membrane Switch Keypad

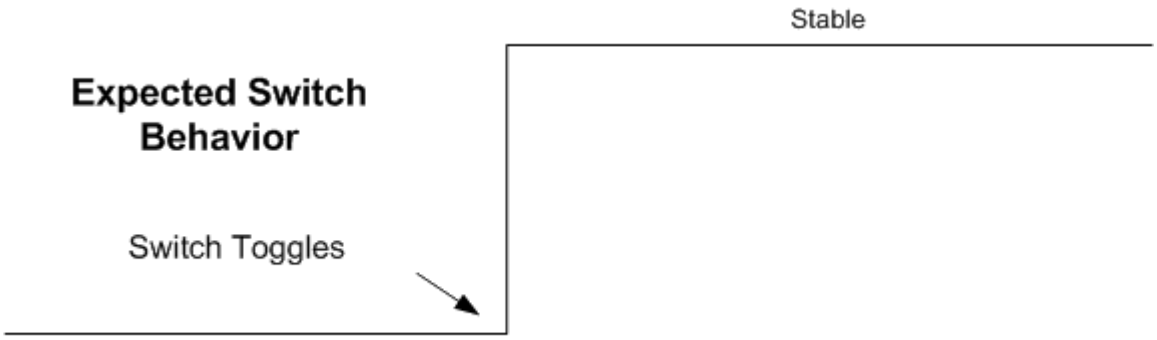

شکل ٤

3x4 Universal Matrix Membrane Switch Keypad

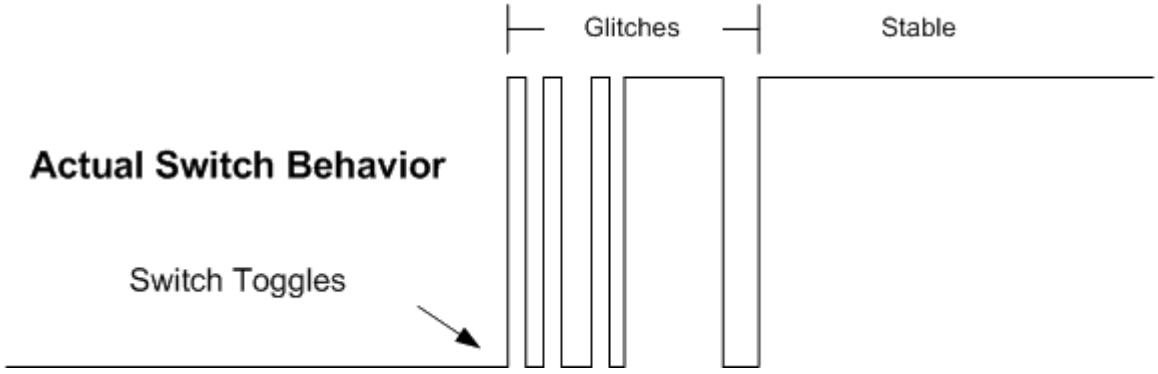

شکل ٦

1x4 Universal Matrix Membrane Switch Keypad

- پدیده‌ی نوسان (bounce) کلید چیست و چگونه می‌توان از بروز اشکالات ناشی از آن جلوگیری کرد؟
منشا پدیده نوسان در فشار دادن کلید است که چون کلید از دو صفحه خازن به وجودآمده یا اتصال دو هادی به وسیله یک هادی دیگر و در حالت فشار داده شده کلید یعنی نزدیک شدن دو صفحه خازن ، این دو صفحه به دقیقا دریک فاصله نمی‌مانند یا در حالت اتصال هادی‌ها سطح هادی ها مدام تماس پیدا کرده و جدا شدن که به جای پالسی به شکل ۷ ، پالسی به شکل ۸ می‌فرستند و پردازنده اینگونه برداشت میکند که چندین و چند بار کلید فشار داده شده و رها شده است



شکل ۷



شکل ۸

و برای جلو گیری از این پدیده می شود چند کار انجام داد:

۱. استفاده از فیلتر هایی دیجیتال یا آنالوگ که سیگنال ارسالی شکل ۸ به صورت سیگنال شکل ۷ به دست پردازنده برسد
۲. برای پردازنده یک بازه زمانی برای هر پالس در نظر بگیریم و مثلا برایش تعریف شود پالسی که طول آن بیشتر از ۱۰ میلی ثانیه است را در نظر بگیر و هر پالسی که طول آن کمتر از آن باشد را نادیده بگیرد

۳. استفاده از خازن به صورت موازی با کلید ها که با شارژ و دشارژ شدن در مواقع نیاز از این پدیده جلو گیری کند

- تعریف مختصر توابع مورد نیاز از کتابخانه keypad.h مانند:
  - Keypad(makeKeymap(userKeymap) , row[] , col[] , rows , cols)

این تابع برای تعریف keypad در آروینو استفاده می‌شود که ورودی اول آن با استفاده از تابع makekeymap که از خود کتابخانه keypad.h است آرایه ای که از صفحه کلید درست کردیم را بهش می دهیم و در ورودی دو پین‌هایی از برد که به سطر ها متصل شده اند را می دهیم و در ورودی سوم پین‌هایی از برد که به ستون ها متصل است را می‌دهیم و در دو ورودی بعدی تعداد ردیف ها و ستون ها را می‌دهیم

  - Char getKey()

این تابع اگر کلیدی فشار داده شده باشد کاراکتر آن کلید را به ما برمیگرداند وگرنه کاراکتر null را می‌فرستد و عملا non-blocking است

  - Char getKeys()

برای زمانی از این تابع استفاده میکنیم که چندتا کلید همزمان فشار داده شود

  - char waitForKey()
  - این تابع هم مثل getkey است اما با این تفاوت که از نوع blocking است یعنی برنامه را نگه می‌دارد تا زمانی که یک کلید فسرده شود
  - KeyState getState()

این تابع وضعیت کلید را به ما برمی‌گردانه. و خروجی این تابع ۴ حالت داره که عبارت است از:

۱. IDLE : هیچ اتفاقی برای کلید نیوفتاده
۲. PRESSED : کلید فشارداده شده
۳. RELEASED : کلید تازه رها شده
۴. HOLD : کلید نگه داشته شده است(که با تابع HoldTime هم میتوان مقدار تایمی که لازم است تا معنی نگه داشته شدن کلید تعریف شود رو بهش میگیم)

  - boolean keyStateChanged()

این تابع فقط به ما میگه که وضعیت کلید تغییر کرده یا خیر

- نحوه و کاربردهای ارتباط سریال در آردوینو:

ار تباط سریالی برای انتقال و گرفتن داده‌ها است که نیاز به ۳ سیم است اول باید GND دو میکرو کنترلر یا هر دو ابزاری که می‌خواهند در ارتباط باشند باید یا ارتباط برد با یک USB و ... دارای یک

زمین مشترک باشند و دو سیم دیگر هم برای اتصال پورت های RX , TX است که TX یکی به RX آن یکی و بلعکس متصل می‌شود

- تعریف مختصر و نحوه کار با توابع ارتباط سریال مانند:
  - begin()

این تابع میاد اون پورت مودر نظر برای برقراری ارتباط را باز میکنه و وقتی begin را صدا میزنیم پورت باز میشه و آماده برای در یافت و یا ارسال داده می‌شود و در آرگومانش سرعت این ارتباط را می‌دهیم که به طور معمول این سرعت رو روی ۹۶۰۰ می گذارند

  - end()

این تابع کاری برعکس begin انجام می‌دهد و پورت ارتباطی را می‌بندد و به اصطلاح آزاد میکند

  - find()

این دستور چیزی که در ورودی دادیم را در بافر میگرد تا آن را پیدا کند که آن را در آرگومان اول قرار میدهیم و اگر ورودی چیزی باشد که لازم است طول آن مشخص باشد در آرگومان دوم این طول را هم می‌دهیم و در خروجی یک Boolean برمیگرداند

  - parseInt()

این تابع میاد و از ورودی یک عدد int میخوند و قابلیت های دیگه ای هم دارد که مثلا آرگومان اول این تابع (lookahead) میتوان بهش گفت که برای مثال اگه a12 را گرفتی a را در نظر نگیر و ۱۲ رو بخون و به عنوان آرگومان دوم(ignore) می‌توان گفت که این کاراکتر ها را کلا در نظر نگیر

  - println()

این تابع مثل تابع print() عمل می‌کند و یک مقداری را میگیرد و آن را چاپ میکند و این دستور به این صورت عمل می‌کند که دونه دونه تمام ورودی ها را می‌خونه و کد اسکی آن را میفرستد فقط در تابع println() مثل این است که یک \n در آخر print قرار داده باشیم که به خط بعد برود

  - read()

ای تابع اولین بایت ورودی به بافر های پین‌های سریالی را می‌خوند را برمی‌گرداند و اگه ورودی نیامده باشد مقدار -۱ را برمی‌گرداند

  - readStringUntil()

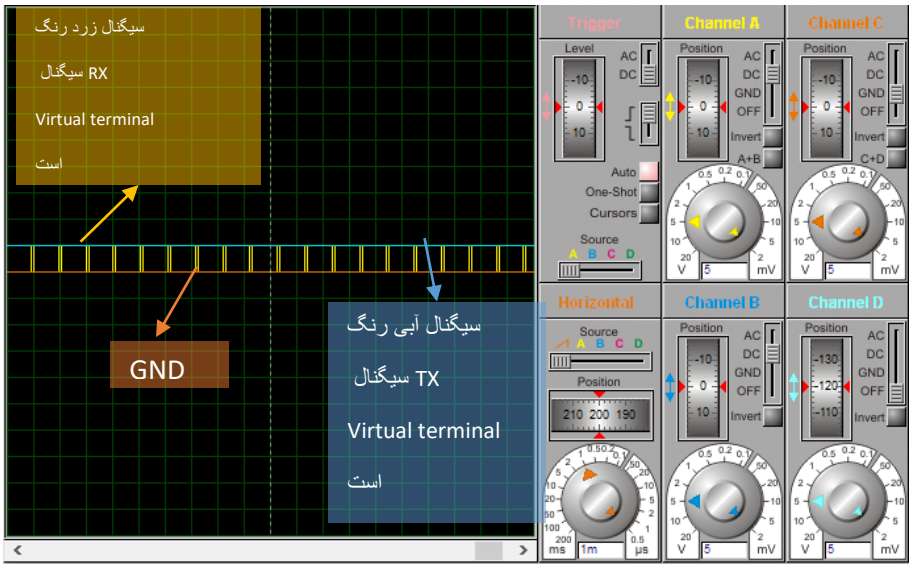
این دیستور انقدر از ورودی می خوند تا به کاراکتری که در ورودی این تابع داده ایم برسیم و خروجی آن یک string است

- write()

این تابع هم برای چاپ در خروجی انجام می شود ولی کل ورودی را به صورت یک بایت در نظر میگیرد و این فرقی است که با دستور print دارد مثلا اگه بنویسیم(Serial.print(68 می‌آید و اول کد اسکی ۶ رو میفرسته و بعد کد اسکی ۸ را ولی اگه همین دسور را با write بنویسیم میاد و کاراکتری را با کد اسکی ۶۸ را میفرستد

سیگنال فرستاده شده در ترمینال:

همان طور که در اسیلوسکوپ در هنگام آزمایش میبینیم در حالتی که دیتایی فرستاده نمیشود یک قطار پالس باریکی دارد(شکل ۱۰) و وقتی سیگنال ارسال می‌شود اول یکسری نوسانات می‌کند و بعد سیگنال در لول high می ماند(شکل ۹) و وقتی دریافتش تموم شد به حالت اولیه بر میگردد



شکل ۱۰