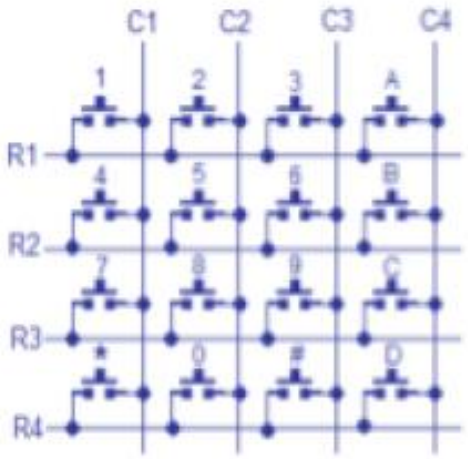
به نام خدا

محمدمهدی هجرتی	**۹۷۲۳۱۰۰**
آزمایشگاه ریزپردازنده - آزمایش ۲	**کیبورد ورودی و ارتباطات سریال**
استاد صالحی	**۱۹ مهر ۹۹**

آزمایش دوم مربوط به کار با کیبورد و ترمینال مجازی می باشد. که ابتدا به بررسی چند تعریف از موارد مورد استفاده در این آزمایش می پردازیم.

انواع keypad ماتریسی و چگونگی کارکرد آن ها

استفاده از keypadها یک راه برای سریع برای گرفتن ورودی از کاربر میباشد. و در ابعاد مختلف مثلا ۳در۴ یا ۴در۴ برای کاربردهای گوناگون وجود دارند. Keypadها در واقع ترکیبی از چند کلید می باشد که به صورت ماتریسی در کنار هم قرار گرفته اند. و به ازای هر سطر یا ستون یک سیم وجود دارد که به پین برد متصل میشود.

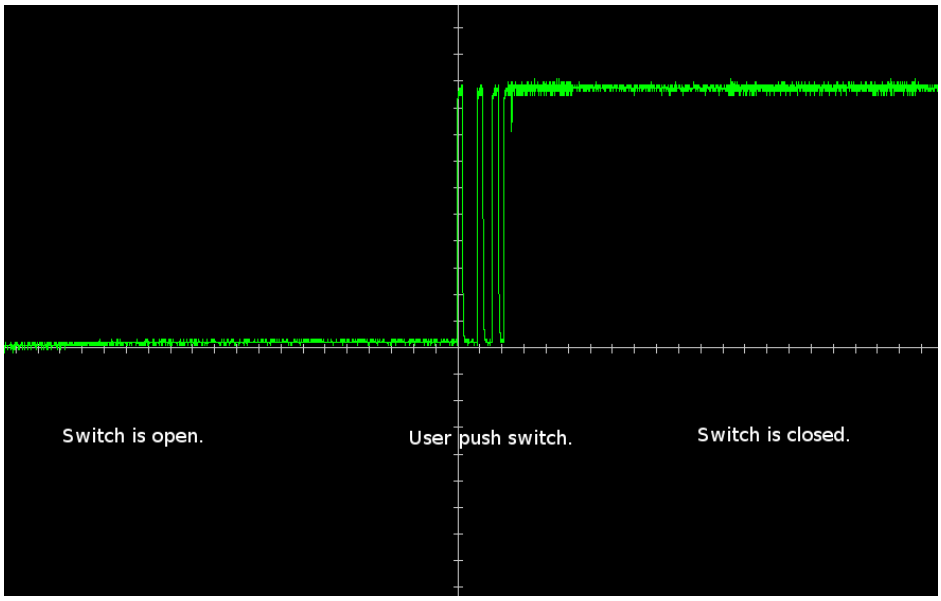


با فشردن هر کدام از این کلیدها ۲ ولتاژ دو سیم متصل میشود و میتوان فهیمد که کلید موجود در کدام سطر و ستون انتخاب شده است.

پدیده نوسان کلید و جلوگیری از آن

با توجه به این که کلید ها از صفحات فلزی که قرار است به هم نزدیک شوند تشکیل شده اند، هنگامی که یک کلید فشرده و این صفحات به هم نزدیک میشوند نوسانات بسیار کوچکی به وجود می آید، که

ممکن است کلید اشتباها این نوسانات را به عنوان فشردن و رها کردن کلید تاقی کند و چند بار قطع و وصلی ثبت کند.



یکی از راه های برطرف کردن این مشکل، بستن خازن به دو سر این کلید است. در این صورت نوسانات اولیه صرف پر شدن خازن شده و بعد از آن کلید فشرده شده در نظر گرفته میشود.

در کتابخانه ی Keypad.h این مشکل به صورت نرم افزاری نیز هندل شده است. درصورتی که کلید کمتر از یک زمان مشخصی در یک وضعیت جدید قرار گرفت، تغییر وضعیت ثبت نمیشود.

توابع مورد نیاز از کتابخانه Keypad.h

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

این تابع درواقع کانستراکتور کلاس keypad می باشد که یک object از این کلاس می سازد تا در ادامه از آن استفاده شود. پارامتر اول آن labelها یا همان مقادیری است که بر روی کیبرد استفاده میشود. پارامتر دوم و سوم سطر و ستون پین های متصل به آردوئینو برای راه اندازی کیپد است. و ۲ پارامتر آخر اندازه ی سطرها و ستون های آن می باشد.

Char getKey()

این تابع در هر لحظه بررسی میکند که آیا کاراکتری وارد شده است یا خیر. درصورتی که وارد نشده باشد، مقدار false برگردانده میشود.

Char getKeys()

برای بررسی تغییر تعداد زیادی از کلیدها از این تابع استفاده میشود. به این صورت که بررسی میکند که آیا کلید وارد شده ی ما جزو کلید هایی که تغییر کرده اند هست یا خیر.

char waitForKey()

این تابع عملکردی مشابه ()getkey دارد با این تفاوت که هر بار منتظر می ماند تا کاربر یک کاراکتر را وارد کند سپس برنامه ادامه پیدا میکند.

KeyState getState()

این تابع state کلید را برمیگرداند.state میتواند یکی از ۴ حالت idle, pressed, released, hold باشد.

boolean keyStateChanged()

با کمک این تابع میتوان از تغییر وضعیت هر یک از کلید ها اطلاع یافت. که آن را با استفاده از یک Boolean نشان میدهد.

نحوه و کاربردهای ارتباطات سریال در آردوینو

برای برقرای ارتباط بین چند میکرو یا ارتباط میکرو با یک usb و ... از ارتباطات سریال استفاده میشود. به این صورت که با اتصال پین های سریال به صورت ضربدری به هم این ارتباط برقرار میشود.

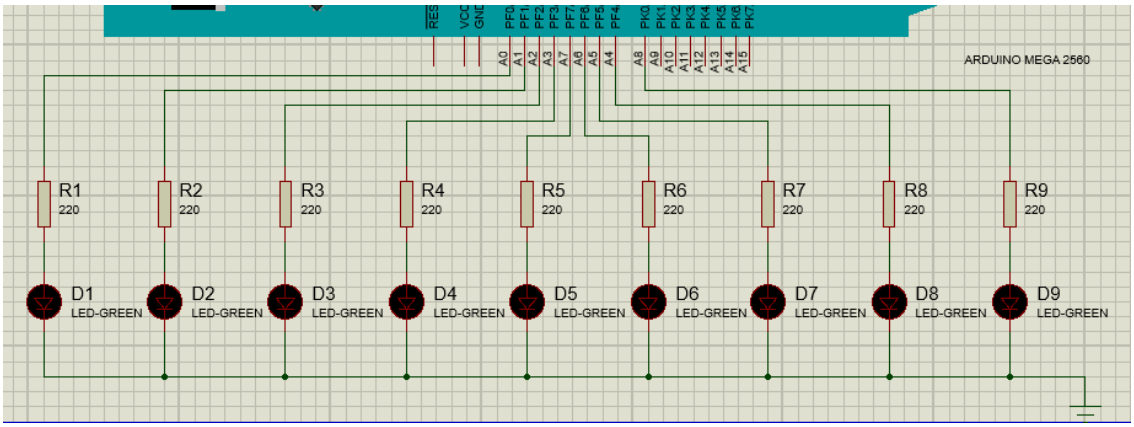
begin()
شروع ارتباط بوسیله ی سریال صفر (tx0, tx1) برد با سرعت مشخص
end()
اتمام ارتباط سریال و آزادسازی پین ها
find()
جست و جوی کاراکتر در سریال دریافت شده که خروجی آن boolean است.
parseInt()
خواندن اعداد (و حذف کردن متغیر خاص از میانه آن)
println()
چاپ یک رشته یا عدد یا ... در سریال و قرار دادن \n در انتهای آن
read()
خواندن اولین بیت دریافت شده
readStringUntil()
خواندن سریال تا زمانی که به یک رشته ی خاص وارد شده برسد.
write()

ارسال داده های باینری به سریال

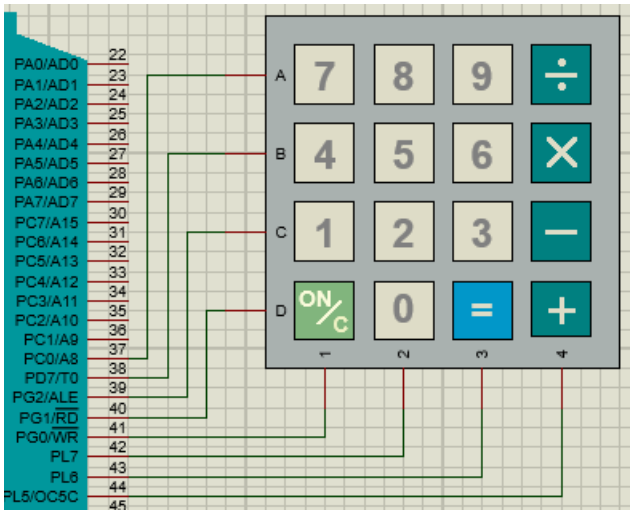
بخش اول

ابتدا باید پایه های A0 تا A8 آردوئینو را با سیم به LED ها متصل کنیم و برای اینکه LED ها دچار آسیب و سوختگی نشوند، مقاومت های ۲۲۰ اهمی را بین مسیر آن قرار دهیم.

برای تنظیم اینکه کدام LED به کدام یک از شماره های پین خروجی متصل شده است. چون ترتیب پین های A0 تا A8 متفاوت از شماره ی آن ها است.



در ادامه در سمت راست برد کیپد ۴ در ۴ را قرار میدهیم و پایه های آن را به برد متصل میکنیم.

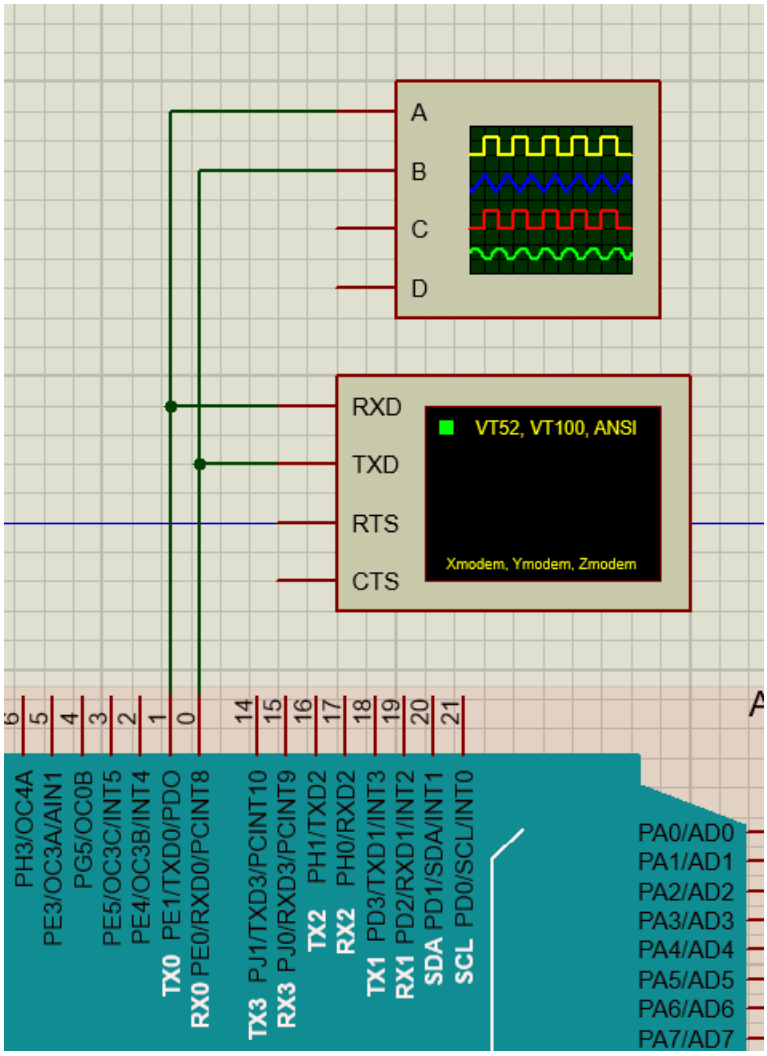


برای کد نیز مشابه آزمایش قبل LED ها را به عنوان خروجی تعریف کرده و این بار شرط روشن شدن آن را فشردن هر یک از کلید ها تنظیم میکنیم.

کلید صفر را نیز برای خاموش شدن تمام چراغ ها قرار میدهیم.

بخش دوم

برای تنظیم ترمینال مجازی و اوسیلوسکوپ لازم است تا پایه های RX و TX دستگاه را به صورت برعکس به پین های TX0 و RX0 برد متصل کنیم. تا ارتباط لازم برقرار شود.



در کد نیز بوسیله ی دستور begin ارتباط را شروع میکنیم. سپس با خواندن کلید های فشرده شده توسط کیپد، آن را در ترمینال نشان میدهیم.

بخش سوم

برای این بخش بدون نیاز به تغییر مدار شبیه سازی شده در proteus، کد جدیدی را مینویسیم تا بتواند با خواندن مقدار وارد شده در ترمینال چراغ متناظر با آن را روشن کند.

در بخش loop برنامه هر وقت عدد جدیدی وارد ترمینال شد، آن را خوانده و مشابه بخش قبل بررسی میکنیم که اگر مربوط به هریک از چراغ ها بود، همان روشن شود.