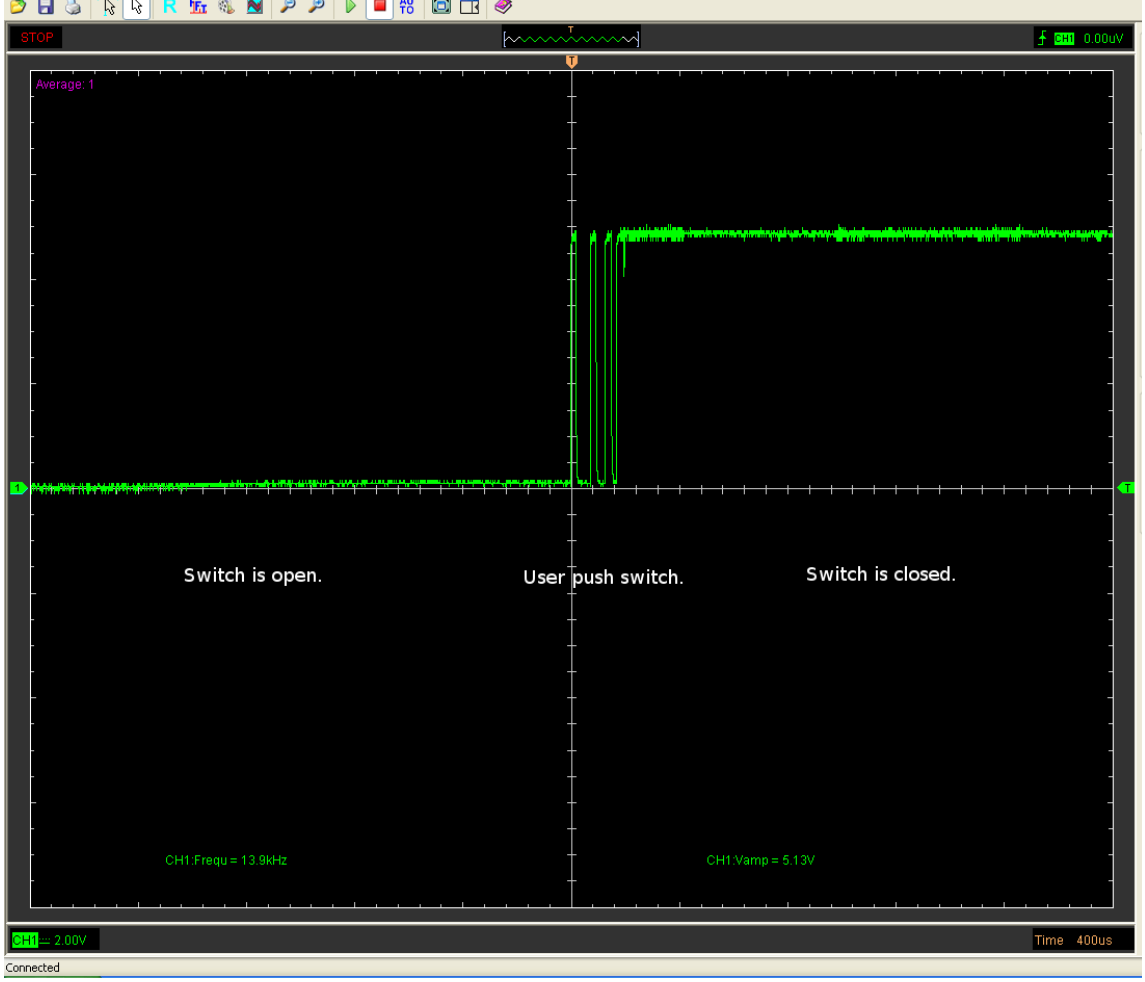
1- انواع keypad ماتریسی و چگونگی کارکرد آن ه ا:

دو نوع keypad دکمه ای و خازنی داریم. Keypad های ماتریسی دکمه ای بر اساس عملکرد میکرو سوئیچ ها کار می می کنند. در واقع میکروسوئیچ ساده ترین راه برای گرفتن ورودی در میکروکنترلر هاست.
در این نوع keypad ها به ازای هر سطر و ستون در صفحه کلید، یک سیم داریم تا با روشن شدن دو سیم در هر فشردن کلید، سطر و ستون مربوط به آن دکمه مشخص می شود و مشخص می شود که کدام دکمه در صفحه کلید فشرده شده است. در هنگام فشردن یک دکمه در این نوع از keypad ها، سیم های سطر و ستون مربوط به آن دکمه low می شود.
در keypad های خازنی بر اساس مکانیزم شارژ و دشارژ خازن عمل می کنیم و وقتی دست خود را بر روی دکمه مورد نظر میگذاریم، خازن مربوط به آن دشارژ می شود و پین هایی که دارد مشخص می کند کدام دکمه در keypad فشرده شده است.

2- پدیده ی نوسان (bounce) کلید چیست و چگونه می توان از بروز اشکالات ناشی از آن جلوگیری کرد؟

کلید در مدار، متشکل از دو صفحه فلزی رو به روی هم است که اگر از هم جدا باشن مدار قطع یا به اصطلاح کلید باز است و اگر آن دو صفحه را به هم بچسبانیم، به این معنی می باشد که کلید را بسته ایم و جریان را در مدار برقرار کرده ایم.
به دلیل اینکه در شرایط واقعی هوا عایق نیست و می تواند جریان الکتریکی را عبور دهد، هنگامی که ما در حال بستن کلید هستیم و دو صفحه فلزی را کم کم به همدیگر نزدیک می کنیم، قبل اینکه این دو صفحه با همدیگر کامل تماس داشته باشند، بارها و بارها از طریق هوا جریان الکتریکی بین آن دو برقرار و قطع می شود که به اصطلاح می گویند پدیده bounce اتفاق افتاده است که مانند این است که ما چند بار پشت سر هم کلید را قطع و وصل کنیم. bounce باعث نویز در مدار می شود و خطا در هنگام آزمایش و یا حتی آسیب به سخت افزار را می تواند منجر شود. برای حل این مشکل دو راه که یکی سخت افزاری و دیگری نرم افزاری است، پیشنهاد می شود.



راه سخت افزاری : می توان دو سر کلید یک خازن خالی وصل کرد تا هنگامی که کلید چند بار به وسیله هوا قطع و وصل می شود، باعث شارژ شدن خازن شود و نویزش گرفته شود و به بقیه مدار نرسد.

راه نرم افزاری: می توان از طریق کد برنامه، یک طول زمانی خاصی را مشخص کرد که دکمه های keypad اگر حداقل به اندازه این طول زمانی فشرده شدند، به این معنی است که دکمه فشرده شده و در زمان های کمتر از این نادیده گرفته شوند. (به دلیل اینکه bounce به سرعت رخ می دهد و کلید خیلی سریع قطع و وصل می شود با یک طول زمانی مناسب و بیشتر می توان از bounce جلوگیری کرد)

3- تعریف توابع کتابخانه keypad.h :

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

یک constructor است که با instance گرفتن از Keypad، صفحه کلیدی که وصل کردیم را با اطلاعاتش مثل سطر و ستون و کلید های موجود در آن ها، به کتابخانه keypad معرفی میکند.

char getKey() :

یک تابع non-blocking که کلید فشرده شده را بر می گرداند.

char waitForKey() :

یک تابع blocking است که بر اساس روال busy waiting کار میکند. هنگام رسیدن برنامه به این تابع، در یک لوپ بی نهایت می افتیم و برنامه آنقدر صبر می کند تا کلیدی فشرده شود تا بتوانیم از آن حلقه بیرون بیاییم.

KeyState getState() :

حالت هر کلید را مشخص می کند و بر میگرداند.
هر کلید می تواند چهار حالت IDLE, PRESSED, RELEASED and HOLD داشته باشد.

boolean keyStateChanged() :

با برگرداندن مقدار ۰ یا ۱ مشخص میکند که آیا یک کلید خاص حالتش تغییر کرده یا خیر. مثلا از حالت IDLE به PRESSED تغییر کند، این تابع مقدار ۱ را بر می گرداند.

4- نحوه و کاربرد ارتباط سریال در آردوینو :

ارتباط سریال یک راه ارتباطی بین بورد آردوینو و کامپیوتر یا دستگاه های دیگر، با کابل usb است. در ارتباط سریال دیتا ها به صورت پشت هم و پی در پی در قالب کد های ۰ و ۱ جابه جا می شوند.

هر بورد آردوینو حداقل یک پورت سریال (UART or USART) دارد و با استفاده از پین های ارسال و دریافت که در Mega 2560 پورت های 0 (RX) و 1 (TX) می باشد، ارتباط سریال انجام می شود. در صورت استفاده از پین های 0 و 1، باید از Serial، در صورت استفاده از پین های 18 و 19، باید از Serial1 و ... استفاده کرد.

5- تعریف توابع سریال :

begin() :

مقدار data rate را بر حسب بیت بر ثانیه برای انتقال داده سریال، به عنوان آرگومان ورودی می گیرد و مشخص میکند. همچنین شروع کننده ارتباط سریال است.

end() :

سریال ارتباط را به پایان می رساند و پین های RX و TX را برای استفاده عمومی به عنوان ورودی/خروجی آزاد می کند. برای شروع دوباره ارتباط سریال باید دوباره begin() فراخوانی شود.

find() :

از بافر به طور متناوب دیتا را می خواند تا آرگومانی را که به آن تحت عنوان target دادیم پیدا کند. اگر پیدا کند true و در غیر این صورت false برگردانده می شود.

parseInt() :

در ورودی سریال به دنبال یک مقدار integer میگردد و دارای timeout است. اگر این زمان تمام شود و ورودی integer دریافت نکند، 0 را بر میگرداند و به پایان می رسد.

println() :

data را در ترمینال به عنوان یک متن قابل فهم (human-readable ASCII text) چاپ می کند.

read() :

ورودی سریال را به صورت بایت می خواند.

readStringUntil() :

ورودی سریال را تا رسیدن به یک کاراکتر خاص، به صورت string می خواند. اگر به آن کاراکتر خاص برسد به پایان می رسد. همچنین می تواند به دلیل سر رسیدن timeout به پایان برسد.

write() :

اطلاعات باینری را بر روی پورت سریال می نویسد. این اطلاعات به صورت آرایه ای از بایت ها فرستاده می شود.