انواع اتصال microswitch ها به برد

میتوان microswitch ها را به دو صورت وصل کرد:

- یک پایه به زمین (0 ولت) وصل شده و پایه دیگری به pull up وصل باشد. در این صورت اگر کلید فشرده نشده باشد خروجی High است و در صورتی که فشرده شده باشد خروجی Low خواهد بود.
- برعکس این موضوع هم صادق است. یعنی اگر پایه دیگر به pull down وصل باشد، در صورت فشرده شدن کلید خروجی High است و در غیر این صورت Low.

انوع keypad های ماتریسی

(نحوه کارکرد) اگر keypad برای مثال 16 دکمه داشته باشد باید 16 سوییچ برای فهمیدن کلید فشرده شده نیاز است و بر روی برد های کوچک قابل پیاده سازی نیست. برای رفع این مشکل از keypad های ماتریسی استفاده میشود که تنها با 8 سیم (پین) میتوان کلید فشرده شده را تشخیص داد.

(نحوه اتصال) 4 سیم اول مربوط به ردیف ها را به ورودی microswitch وصل کرده به پیش فرض همه را High کرده (5+ ولت) و 4 سیم بعدی مربوط به ستون ها را به ورودی بورد Arduino در وضعیت INPUT_PULLUP وصل میکنیم که در حالت عادی (فشرده نشده) ولتاژ 5+ باشد. انواع آن:

- ### Keypad های عادی

این Keypad ها به ازای تعداد دکمه، پین های خروجی دارند و به همان تعداد به microswitch نیاز دارند که ساده ترین نوع Keypad هستند.

- ### Keypad های 4x4 معمولی

در صورت فشرده شدن هر کلید، یک سیم از 4 سیم چپ و یک سیم از 4 سیم راست فعال شده و میفهمیم چه کلیدی فشرده شده است. 4 پین چپ مربوط به پین های ردیفی و 4 پین راست مربوط به پین های ستونی هستند.

- ### Keypad های خازنی

روش کارکرد این Keypad ها به این صورت است که خازن های Keypad از پیش شارژ شده اند و با لمس کلید های Keypad، خازن ها دشارژ شده و میفهمیم کدام کلید فشرده شده است. 8 خروجی چپ این Keypad ها مانند Keypad های معمولی است.

طریقه scanning

میدانیم ستون ها مقدار 5+ یا 1 منطقی دارند. در این روش مدام چک میکنیم اگر کلیدی فشرده شده باشد و ردیف مربوطه Low باشد، ستون نظیر آن هم Low میشود. به عبارتی به طور مداوم یکی از ردیف ها (که پیش فرض High

هستند) را Low کرده و با ستون ها چک میکنیم؛ اگر ستون هم Low باشد یعنی کلید نظیر سطر و ستون فشرده شده است.

پدیده Bouncing و نحوه جلوگیری از آن

فشرده کردن یک کلید میتواند باعث ایجاد نوسان یا نویز شده که میتواند به عنوان فشرده شدن چندین بار آن دکمه ثبت شود. برای جلو گیری از آن دو راه وجود دارد:

- **استفاده خازن در دو سر کلید ها**

خازن انرژی نویز یا نوسان ایجاد شده را جذب کرده و شارژ میشود، در نتیجه مانع پدیده Bouncing میشود. اما در اینجا چون در Keypad خازنی در دو سر کلید ها وجود ندارد و اینکه تعداد کلید ها زیاد است این روش مطلوب نیست.

- **استفاده از روش های نرم افزاری کتابخانه** Keypad

به این صورت که اگر کلیدی فشرده شده وضعیت آن برای مدت خیلی کوتاهی نباید تغییر کند. با عبارتی هر کلید باید مدت کوتاهی نگه داشته شود تا ثبت شود.

- **استفاده از** Latch

یک Latch فقط یک لحظه آخر از ورودی داده شده (اینجا فشرده شدن کلید) را ثبت میکند و نوسانات قبل یا بعد آن را در نظر نمیگیرد.

توابع کتابخانه Keypad

- Keypad Constructor

در ورودی یک ماتریس از کاراکتر های مورد نیاز، لیست های پین های سطر و ستون، تعداد سطر و ستون است.

- Char getKey()

این تابع یک کاراکتر از کلید های از پیش تعریف شده در میگرداند. حائز اهمیت است که این تابع در صورتی که کلیدی فشرده نشده باشد مقدار پوچ بر میگرداند. (معادل 0)

- Char getKeys()

این تابع کلید هایی که همزمان فشرده شده باشند را برمیگرداند.

- Char waitForKey()

این تابع منتظر میماند تا کلیدی فشرده شود و آن را برمیگرداند. به عبارتی حلقه Loop را نگه میدارد و مانع اجرا شدن کار های دیگر میشود. البته مانع interrupt ها نمیشود !؟

- KeyState getState ()

این تابع حالتی که هر کدام از کلید های Keypad دارد را برمیگرداند. حالت های موجود:

IDLE, PRESSED, RELEASED, HOLD

- Boolean keyStateChanged()

خروجی این تابع باینری است و اعلام میکند که آیا وضعیت کلید خاصی تغییر کرده یا خیر. اگر دوبارgetState کنیم و کلید نگه داشته شده باشد نمیفهمیم وضعیت آن دقیقا چیست، با این تابع میتوان فهمید.

سریال، پین ها و توابع آن

در بورد Arduino پین های که با RX و یا TX علامت گذاری شده باشند پین های سریال هستند و کاربرد این پین ها اتصال با کامپیوتر و یا ارتباط بین چند برد با هم است. برای ارتباط سریال به حداقل 3 سیم احتیاج است. سیم GND برای یکسان سازی ولتاژ مرجع، دو سیم دیگر RX و TX هستند که به ترتیب برای دریافت و ارسال استفاده میشوند.

توابع آن:

- Begin()

این تابع پورت سریال را با سرعت تعیین شده باز میکند. (اتصال سریال را برقرار میکند)

- End()

این تابع برعکس تابع قبلی پورت سریال را میبندد و پین ها را آزاد میکند.

- Find()

این تابع در بافر ورودی رشته مقدار مورد نظر را جست و جو میکند و خروجی باینری میدهد.

- parseInt()

این تابع ورودی عدد میخواند. پارامتر lookahead مشخص میکند تابع بعد از چه کاراکتری عدد را بخواند.

- println()

این تابع یک مقدار را چاپ کرده و به خط بعدی میرود.

- read()

این تابع اولین بایت ورودی را برمیگرداند و آرگومانی نمیگیرد. (اگر ورودی نداشته باشیم 1– برمیگرداند)

- readStringUntil()

این تابع سریال را تا ورودی را تا زمانی که به کاراکتر مورد نظر برسد میخواند.

- write()

این تابع مقدار پاس داده شده را میفرستد که میتواند عدد، کاراکتر یا رشته باشد. با دستور print میتوان عینا مقدار را فراخواند.

پرسش

در UART فقط با دیتا نیاز است و ماژول آسنکرون کار میکند اما در USART علاوه بر دیتا به کلاک هم نیاز است و دیتا با سرعت ثابت منتقل میشود در صورتی که در UART این اتفاق نمی افتد. به همین علت USART سریع تر است.