پیش‌گزارش آزمایشگاه _ آز2 : کیبورد ورودی و ارتباطات سریال

روش های ورودی دادن به میکروکنترلر به انتخاب کاربر :

1. میکروسوییچ ها:

سوییچی که داری چهار پایه است و به صورت ضربدری (بالاراست به پایین چپ و برعکس) متصل می شود. میکروسوییچ ها ساده ترین راه گرفتن ورودی هستند.

برای متصل کردن هر سوییچ به دو پایه از آن و دو پورت نیاز است که یک سر آن به پایه مثبت و یک سر آن به ورودی متصل می شود و زمان فشار دادن دکمه ورودی مثبت و رها کردن آن منفی شدن ورودی را نشان می دهد (وجود یک مقاوت روی پایه ورودی باری pull down باعث می شود تا عدد رندومی تولید نشود و زمانی که دکمه فشار داده نمی شود 0 مطلق در خروجی خوانده شود.)

در صورت استفاده از برد mega2560 و ورودی pin(#, INPUT_PULLUP); نیاز است که یک پایه آن به gnd و پایه دیگر به ورودی میکرویی که روی مود inuut pullup قرار دارد متصل شود و در صورت عدم فشار دکمه high و زمانی که دکمه فشار داده می شود low باز می گرداند.

چون تعداد پورت های میکرو Atmega128 (که نسبتا کوچک نیز هست) کم است و برای ورودی گرفتن از همه ورودی ها با میکروسوییچ کافی نیست، این روش به صرفه و منطقی نیست.

1. کیپد میکروسوییچ:

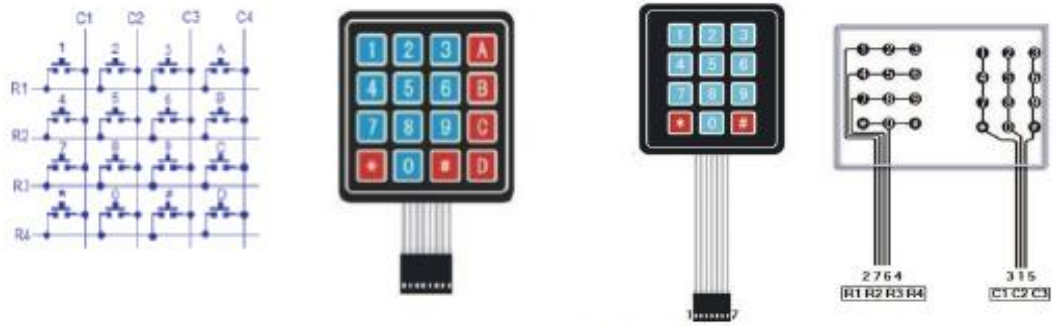
یک ماتریس از میکروسوییچ ها که به صورت دو به دو متقاطع پیاده شده اند به طوری که اگر هر ستون به هر ردیف متصل شده باشد به این معنی ست که کلید فشار داده شده است. در این روش با اسکن کردن می توان یافت که کدام کلید ها با یکدیگر فشار داده شده اند. کیپد های معمولی 4x4 و دارای 8 خروجی هستند. خروجی های کیپد ممکن است ترتیب و ترکیب های متفاوتی داشته باشند اما به طور نرمال ردیف ها در چپ کیپد و از بالا به پایین شروع شده و ستون ها نیز در بالا از چپ به راست شماره گذاری می شوند.

ورودی دادن در این روش به این صورت است که در صورت فشار دادن یک سوییچ یکی از سیم های سمت چپ خارج شده از کیپد به یکی از سیم های سمت راست متصل می کند.

در عملیات اسکن کردن 4 خروجی اول از چپ که مربوط به ردیفها هستند را به خروجی های میکرو متصل میکنندو همه را high (+v5)قرار می دهند و 4 خروجی دوم که مربوط به ستون ها هستند را به ورودی های درمود input pullupمیکرو وصل می کنند(یعنی در حالت دیفالت در سطح منطقی 1و درحالتی که دکمه فشار داده شد 0 شود و همه اینها بر عهده کتابخانه keypad است).

در روش اسکن کردن عملیات به این صورت است که در یک حقله بینهایت هر بار یکی از کلید های ردیف را به ترتیب lowکرده و تمام ستون ها را بررسی می کند. اگر کلیدی از همان ستون نیز low شده باشد یعنی سوییچ مربوط به آن ستون و ردیف فشار داده شده است.

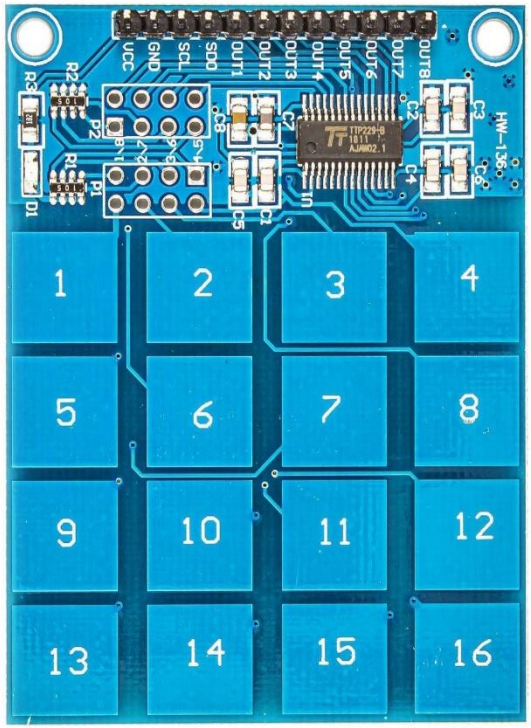
فشار دادن سوییچ ها باعث نوسانی می شود که سیستم آن را چندبار فشار دادن دکمه تعبیر می کند. برای جلو گیری از این مشکل یک روش آن کتابخانه کیپد است که با روش های نرم افزاری یک تایم آپی برای هر دکمه تعریف کرده به طوری که تغییر استیت در یک مدت زمان خاص را فشار دادن کلید در نظر بگیرد و روش دیگر لچ ها و خواندن پین ها در لبه ساعت و یک لحظه است و نوسانات قبل بعد را در نظر نمی گیرد.



تصاویری از ساختار درونی و بیرونی دو کیپد

1. کیپد خازنی:

مانند کیپدهای خازنی هستند و سطح آن فقط جدول بدون دکمه ای از اعداد و ورودی ها است.
روش کار کرد آن ها به این صورت است که باری را در خازن های مدار بالای کیپد شارژ می کند و در صورت تپ کردن و لمس کردن صفحه کلید آن کلید دشارژ شده و مدار بالا فشار دادن دکمه را تشخیص می دهد.
مدار بالای این کیپد ها مانند کیپد های سوییچی 4در4 هشت خروجی است.



کتابخانه های keypad.h :

Boolean keyStateChanged():

جدید در نسخه 2.0: اجازه می دهد بدانیم چه زمانی کلید از یک حالت به حالت دیگر تغییر کرده است. به عنوان مثال ، به جای فقط آزمایش یک کلید معتبر ، می توانید زمان فشار دادن یک کلید را امتحان کنید. این تابع وجود دارد تا if های get state را حذف کند و با set debouncetime که فاصله زمانی خاصی را بین هر دوبار فشردن کلید ایجاد می کند تغییر state میدهد.

Char getKeys():

برای آنکه بفهمیم آیا همزمان چند کلید با هم فشار داده شده اند یا خیر. وخروجی آن یک بولین است. می توان با توابع bool isPressed(char keychar) بررسی کرد که کلید مورد نظر ما فشار داده شده است یا خیر.

KeyState getState():

حالت فعلی هر یک از کلیدها را برمی گرداند. چهار حالت دارد : بیکار ، تحت فشار ، آزاد شده و یک مدت زمانی نگهداری شده(با تابع setholdtime زمان داده می شود که چقدر نگه داشته شود) به طور خودکار در یک حقله هر بار کلید ها بررسی می شوند و وضعیتشان ثبت می شود.

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

زبان استفاده شده در برد آردایینو c++ است و شی گرا است.
Keypad یک کانستراکتور است که در وردی اش یک کی مپ بدهیم که از کلید هایی که می خواهد به کاراکتر مورد نظر مپ شود و ستون ها و ردیف ها را مشخص کند. کار آن ساخت یک آبجکنت کی پد است.

Char getKey():

کلید فشرده شده را در صورت وجود برمی گرداند. این عملکرد غیر مسدود کننده (non_blocking) است . (یعنی اگر کلیدی فشار داده نشود کاراکتر خالی(false) باز می گرداند و باید حتما با یک شرط ولید بودنش چک شود.)

Char waitForKey():

این عملکرد برای همیشه منتظر می ماند تا اینکه کسی یک کلید را فشار دهد. (برعکس get key). یک حلقه است که آنقدر می چرخد تا event اتفاق بیفتد.
هشدار: تا زمانی که کلید فشرده نشود ، همه کدهای دیگر را مسدود می کند. به معنای عدم چشمک زدن LED ها و عدم به روزرسانی صفحه LCD است و هیچ چیز به جز روال وقفه(intrrupt) وجود ندارد.

در حالت عادی high است و اگر از بیرون 0 شود میکرو ریست می شود
5 volt
0 volt
ولتاژ مرجع
ARD1
ARDUINO MEGA 2560
Power arduino
www.TheEngineeringProjects.com
Arduino Mega 2560
ATMEGA2560
16AU 1126
Reset BTN
ANALOG IN
آنالوگ پورت
دیجیتال پورت
DIGITAL
COMMUNICATION
PWM
مجموعه پین های سریال
NO
NO
NO
GND
VCC
RESET
AREF
پین های خاصی که می توانند دیجیتال هم استفاده شوند
انتقال داده ها

نحوه کاربرد های سریال در آردااینو

مهم ترین کاربرد سریال ها برای اتصال USB و یا ارتباط و انتقال اطلاعات میان دو یا چند میکرو یا برد آردااینو با هم است.
یک روش آن است که پین های سریال برد های مختلف را بهم وصل کرده و در پورت ها داده ها را بخوانیم یا دریافت کنیم. پورت سریال دارای یک سری پروتکل و استاندارد است.
برای ارتباط دو طرفه سریال ما به حداقل به 3 سیم نیاز داریم :

1. gnd(ولتاژ مرجع را میان دو طرف تعیین می کند. در واقع ولتاژ 0 برای برد، USB و کامپیوتر باید یکسان باشد)
2. RX برای دریافت اطلاعات است.
3. TX برای انتقال داده ها است.

برای ارتباط میان 2 برد شرط لازم آن است کهRX برد اول به TX برد دوم و برعکس متصل شود. (گوینده به شنونده متصل شود)
پین هایی که ماربرد سریال ندارد هم می توانند به عنوان سریال استفاده شوند زیرا سخت افزار ویژه ای لازم ندارد و برد آردااینو خود روی پین ها انتقالات را ساپورت می کند ولی در زمان انتقال اطلاعات پردازنده متوجه آمدن داده نمی شود و ممکن است در حال انجام عملیات دیگری باشد برای همین دیتا در بافری ذخیره شده و میس نمی شود ولی پین های غیر انتقال اطلاعات این ویژگی را ندارد برای همین پینی که به طور خاص برای RX استفاده می شود باید inttrupt را ساپورت کند.

کتابخانه سریال و توابع آن :

Serial.println()

ورودی را چاپ می کند و انتها یک\n می زند و به خط بعدی می رود.
عدد را به صورت ascii کد می فرستد.

Syntax

```
Serial.println(val)
Serial.println(val, format)
```

Serial.read()

اولین بایت ورودی را بر می گرداند و اگر ورودی نیامده باشد -1 باز می گرداند.

Syntax

```
Serial.read()
```

Serial.readStringUntil()

تا زمانی که به ورودی مورد نظر نرسیده باشیم داده را می خواند.

Syntax

```
Serial.readStringUntil(terminator)
```

Serial.write()

عینا چیزی را که در ورودی آن نوشته شود می نویسد.
یعنی در صورت خواندن از سمت دیگر بدون هیچ تغییری ورودی را می توان خواند.

Syntax

```
Serial.write(val)
Serial.write(str)
Serial.write(buf, len)
```

Serial.begin()

پورت سریال را باز می کند و آماده است که اطلاعات را دریافت و انتقال دهد.

Syntax

```
Serial.begin(speed)
Serial.begin(speed, config)
```

Serial.end()

بر عکس begin است و TX , RX را آزاد می کند. (می بندد)

Syntax

```
Serial.end()
```

Serial.find()

مقدار ورودی تابع را در بافر جست و جو می کند.
خروجی آن یک بولین است.

Syntax

```
Serial.find(target)
Serial.find(target, length)
```

Serial.parseInt()

عدد می خواند و ارور ها و نوع خواندن را هندل می کند.

Syntax

```
Serial.parseInt()
Serial.parseInt(lookahead)
Serial.parseInt(lookahead, ignore)
```