دانشکده فنی انقلاب اسلامی

بسمه تعالي

مایکرو دیزاینر الکترونیک

www.microdesigner.ir

ساخت تابلو روان صنعتي سه رنگ

استاد راهنما: آقای انصاری

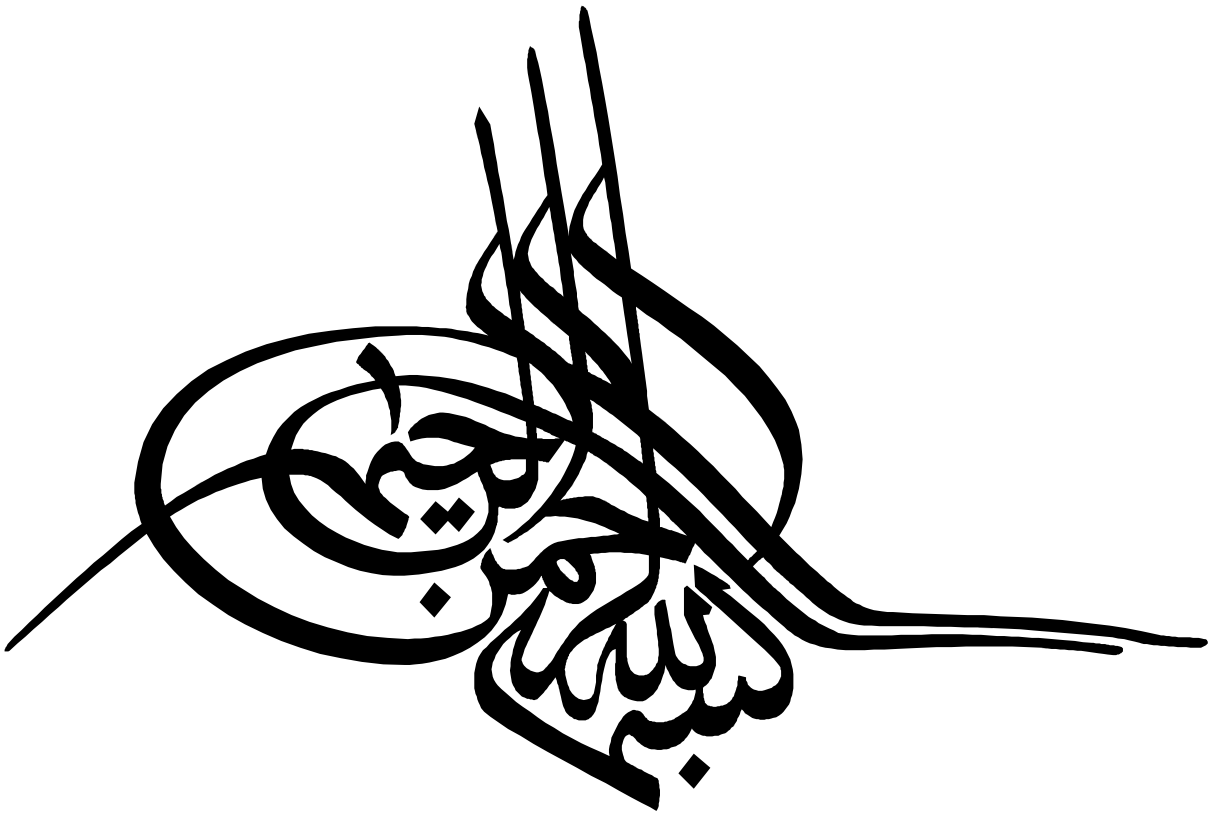
دانشجویان:

شهریار فرهادي

محمد کرامتي

نیمسال اول

سال تحصیلي 89-88


بسم الله الرحمن الرحيم

هيئت ژوري

پايان نامه طراحي و ساخت تابلو روان صنعتي سه رنگ به تاريخ 88/11/6 شماره

مورد پذيرش هيات محترم داوران با رتبه و نمره قرار گرفت.

استاد مشاور

داور داخلي

داور مدعو

مدير پروژه

رئيس انستيتو

فصل اول

آشنايي با تابلو روان

مقدمه :

همانطور كه مي دانيد امروزه بسياري از تابلوهاي تبليغاتي در خيابان ها نصب مي شوند و بر روي آنها نوشته هايي بصورت انگليسي و فارسي حركت مي كنند كه به انها افكت مي گويند. اگر به اين تابلوها دقت كنيد، مي بينيد كه تعداد زيادي LED در كنار هم قرار گرفته اند و اين LED ها توسط يک ميكروكنترلر كنترل مي شوند. امروزه انواع مختلفي تابلو روان در ابعاد و سايزهاي مختلف وجود دارند و هر چه تابلو بزرگتر باشد تعداد LEDهاي آن بيشتر است.

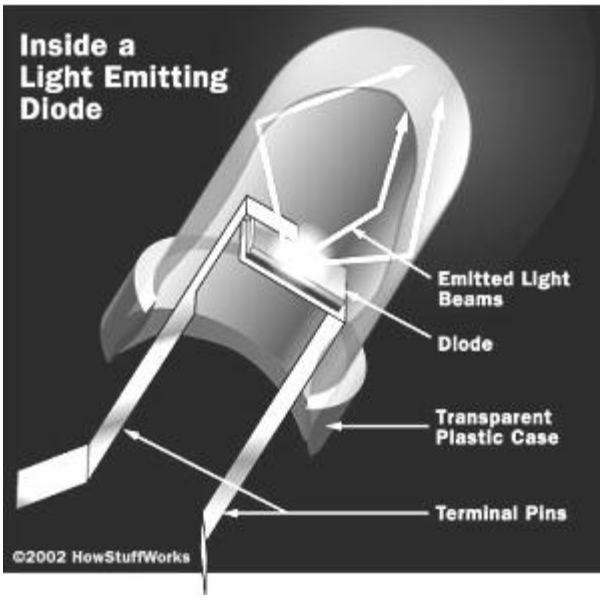
حال ممكن است اين سوال پيش بيايد كه چرا از LCD استفاده نمي شود؟ اگر به يک LCD دقت كرده باشيد شما مي توانيد از روبرو به آن نگاه كنيد و به صورت مايل تصوير آن واضح نيست و همچنين در روز تصويرش از دور ديده نمي شود به همين دليل تابلو روان ها با LED ساخته مي شوند زيرا هم در روز ديده مي شوند و هم از كنار قابل رويت هستند.

تابلوهاي فوق به دو صورت رنگي و تك رنگ موجود است كه از تابلوهاي تك رنگ به عنوان تابلو نويسنده و از تابلوهاي چند رنگ به عنوان تابلوهاي تبليغاتي در پاركها، اماكن عمومي و يا حتي تلويزيون‌هاي شهري استفاده مي‌كنند.

آشنايي با LED:

تمام كساني كه با الكترونيك سروكار دارند با LED كه از قطعات پايه‌اي الكترونيكي است آشنايي دارند و با آن كار كرده اند. به طور ساده LED يك نوع لامپ است كه در رنگ هاي متفاوت ساخته مي شود كه سرعت بالايي هم دارد LEDها انواع مختلفي نيز دارند.

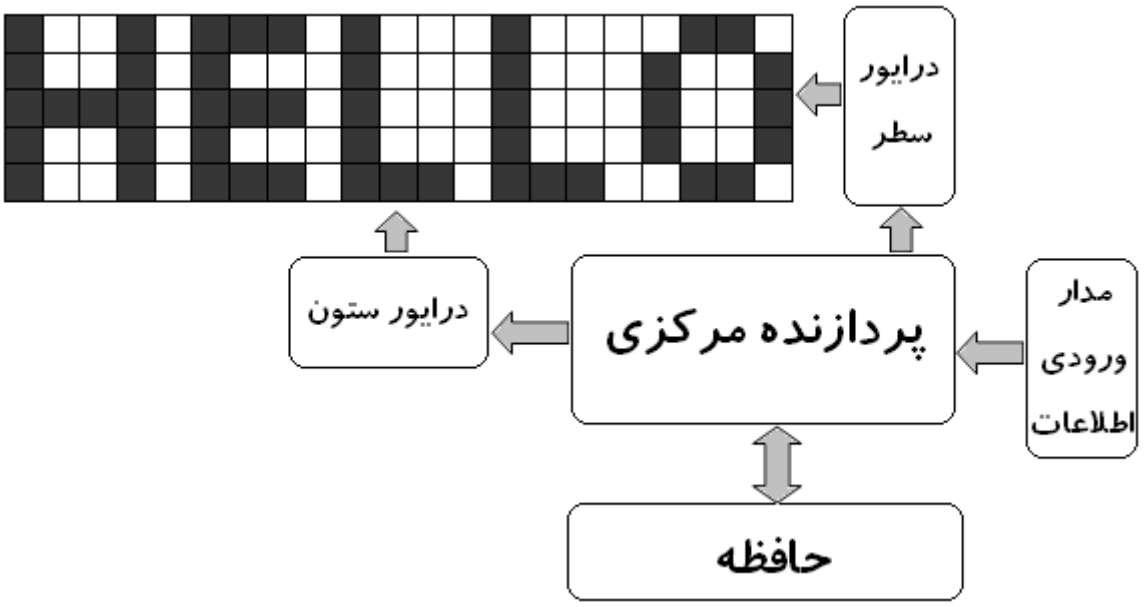
شكل زير ساختمان داخلي يك LED را نشان مي دهد:



بايد دقت داشته باشيد كه در ساخت تابلو روان از LEDهاي مرغوب استفاده كنيد زيرا LEDهاي معمولي ممكن است جريان نشتي داشته باشند كه باعث نويز انداختن در كار شده و يا ميزان نور آنها يكسان نباشد.

بلوك دياگرام تابلو روان:

اولين چيزي كه در مورد تابلو روان بايد دانست بلوك دياگرام آن است. شكل زير نماي كلي يک تابلوروان را نشان مي دهد:



توضيح قسمت هاي مختلف بلوک دياگرام :

LEDها :

همانطور که در شکل مي بينيد LEDها بايد به صورت ماتريسي بسته شوند. ماتريس يعني اينکه داراي سطر و ستون باشد. به عنوان مثال يک تابلوي 5×8 يعني شامل 8 سطر و 5 ستون است.

درايور ستون:

اين درايور جهت راه اندازي ستون ها مورد استفاده قرار مي گيرد.

درايور سطر:

اين درايور جهت راه اندازي سطرها مورد استفاده قرار مي گيرد.

پردازنده :

اين قسمت عمل پردازش را انجام مي دهد يعني به درايوهاي سطر و ستون فرمان مي دهد که کدام سطر و ستون ها روشن و کدام خاموش شوند.

تجهيزات ورودي اطلاعات:

اين بلوک نشان مي دهد که اين دستگاه شامل يک قسمت سخت افزاري است که کاربر اطلاعاتي را وارد کند و پردازنده با توجه به اطلاعاتي که کاربر وارد مي کند کار پردازش را انجام دهد.

نحوه ي عملکرد کلي يک تابلو روان :

در واقع يک تابلوي نمايشگر ديجيتالي، متن مورد نظر خود را از طريق تجهيزات ورودي همچون صفحه کليد و يا پورت سريال دريافت مي کند و اين اطلاعات را در اختيار پردازنده قرار مي دهد. سپس پردازنده پس از آناليز اطلاعات آن را در حافظه تابلو ذخيره نموده. علاوه بر آن حافظه موجود در تابلو مي تواند کدهاي برنامه را در خود نگهداري نمايد. از طرفي پردازنده با توجه به اطلاعات ذخيره شده، سيگنالهاي لازم را جهت نمايش توليد کرده و در اختيار درايورها قرار مي دهد. با توجه به اينکه نحوه چيدمان LEDها در نمايشگر بنا به دلايلي که بعدا توضيح داده خواهد شد به صورت ماتريسي مي باشد، لذا دو دسته درايور براي راه اندازي ماتريس نياز است که شامل درايورهاي سطر و درايورهاي ستون مي باشند. اين درايورها با توجه به فرامين دريافتي از سوي پردازنده، با روشن و خاموش نگاه داشتن LEDهاي موجود در ماتريس، باعث به نمايش در آمدن مطالب (اعم از متن و يا تصوير) بر روي ماتريس خواهند شد.

اثر فليکر :

اثر نور در چشم انسان براي مدت کوتاهي باقي مي‌ماند. اين خاصيت را اثر پس ماند نور (Flicker) مي‌نامند. برمبناي همين خاصيت است که در سينما و تلويزيون احساس پيوستگي تصوير بوجود مي‌آيد.

چنانچه تصاويري که از يک حرکت مثلا راه رفتن انسان عکس برداري شود و سپس با سرعت 16 بار در ثانيه به نمايش درآيد، چشم انسان منقطع بودن تصاوير را احساس نکرده و تصاوير را بطور پيوسته حس مي‌کند. بر مبناي اين خاصيت بود که صنعت سينما بوجود آمد.

ترتيب كار به اين صورت است كه توسط يک دوربين فيلمبرداري مخصوص كه قادر است در هر ثانيه 16 تصوير از يک صحنه عكس برداري نمايد، تصاوير تهيه شده سپس با همان سرعت به نمايش در مي‌آيند. البته به علت اينكه با 16 تصوير در ثانيه حركات نرم و طبيعي نداريم، فركانس مزبور بعدا به 24 تصوير در ثانيه افزايش داده شد. در اين فركانس براي بيش از 90 درصد حركات، پيوستگي طبيعي بوجود مي‌آيد. به همين علت به فركانس مزبور حد پيوستگي گفته مي شود. مشكل ديگر مسئله چشمک زدن تصوير است.

در فركانس 24 تصوير در ثانيه اگر چه مسئله پيوستگي تصاوير حل مي شود اما تصاوير چشمک مي‌زنند و اين بخاطر اين است كه چشم اگرچه در اين فركانس، تصاوير را پيوسته مي‌بيند و حركات را طبيعي احساس مي كند اما خاموش شدن صحنه كه در حين تعويض يک تصوير به تصوير بعدي بوجود مي‌آيد را بصورت چشمک زدن تصوير احساس مي كند. اين پديده بخصوص براي تصاويري با نور بيشتر محسوس‌تر است.

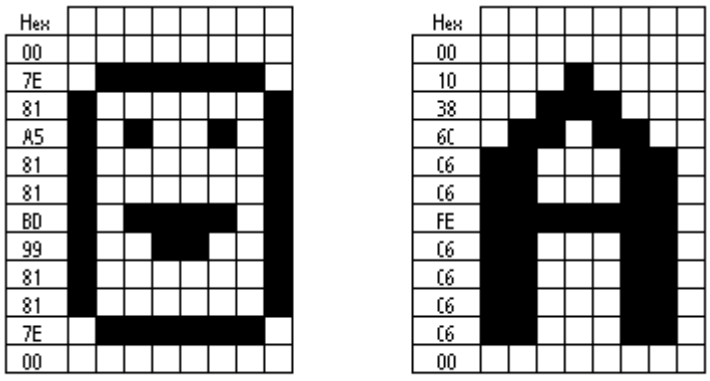
براي رفع اين مشكل بايد حداقل 48 تصوير در ثانيه به نمايش درآيد تا اثر چشمک زدن از بين رود. در سينما چون نمايش 48 تصوير در ثانيه، اشكالات عملي بوجود مي‌آورد، مسئله را به طريق ديگري حل نموده‌اند. به اين ترتيب كه سرعت حركت نوار فيلم از مقابل لامپ پروژكتور همان 24 تصوير در ثانيه است منتهي به كمک يک ديافراگم گردان به هنگام تعويض يک فريم به فريم بعدي و همچنين در زمان نمايش فريم و درست در وسط زمان مزبور نور لامپ پروژكتور به فيلم قطع مي‌شود. با اينكار هر فريم دو بار روشن و خاموش مي‌شود.

با اين تدبير كه هر تصوير دو بار روشن مي شود و سرعت حركت نوار 24 تصوير در هر ثانيه است، از نظر چشم 48 تصوير در ثانيه احساس مي شود و مشكل چشمک زدن از بين مي رود. در تابلوهاي روان هم مسائل پيوستگي تصاوير و

همچنين چشمک زدن، عوامل تعيين کننده سيستم جاروب و زمانهاي مربوطه هستند.

جدول گلايف:

براي نمايش هر تصوير ويا متني در تابلو روان ما نياز به اين داريم که ابتدا آنرا به نقاط تشکيل دهنده تقسيم کنيم. در مورد حروف نيز بدين شکل عمل ميکنيم و به ازاي هر حرف يک جدول درست ميکنيم، به مجموعه اين جداول که شامل تمامي حروف مي شود اصطلاحا جدول گلايف مي گويند. جهت روشن شدن مطلب به تصوير زير دقت نماييد.



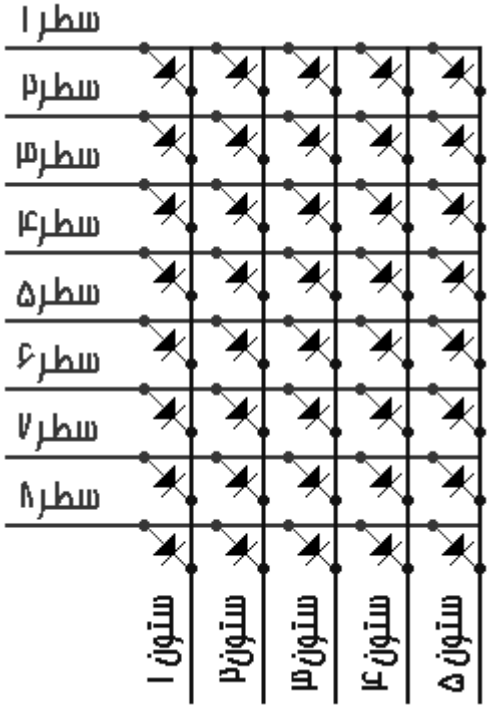
جدول گلايف نمونه براى كاراكتر A و صورتک خندان

همانطور که مشاهده مي کنيد در تصوير فوق نمونه اي از جدول گلايف را براي حرف A و صورتک خندان ترسيم شده و در ستون سمت چپ هر تصوير کد هگز(Hex) مربوط به هر سطر را درج شده است. که با فرض اين بوده که پهناي هر کاراکتر هشت پيکسل بوده و به ازاي هر پيکسل فعال بيت مرتبط با آن يک در نظر گرفته شده. در نتيجه در دو مثال فوق که ارتفاع هر کاراکتر 12 سطر است، براي ذخيره اطلاعات هر کاراکتر به 12 بايت نياز داريم. حال بسته به زبان برنامه نويسي که از آن استفاده ميشود نحوه ذخيره و بازيابي اين جدول متفاوت خواهد بود. البته نرم‌افزارهايي نيز جهت طراحي فونت نيز در اين زمينه وجود دارد.

جاروب ساده :

تشکيل تصوير بر روي پانل تابلو، نياز به روشن و خاموش نگه داشتن LEDهاي موجود بر روي تابلو متناسب با تصوير مورد نظر است. بنابراين نياز به کنترل تک تک LEDهاي

موجود در تابلو مي باشد. از طرفي هر LED داراي دو پايه است (با فرض تک رنگ بودن) و در صورتي که ما يک پانل LED با ماتريس 10x10 داشته باشيم، دويست پايه و يا دويست سيم جهت کنترل داريم. مسلما استفاده از اين تعداد سيم مقرون به صرفه نيست و باعث پيچيدگي مدار خواهد شد. جهت بر طرف کردن مشکل فوق مي‌توان پايه هاي يکسان در LEDها را به صورت سطري و ستوني به يکديگر متصل نمود. به تصوير زير دقت کنيد :



همانطور که در تصوير مشاهده نموديد، در اين آرايش آند تمامي LEDهاي موجود در يک سطر يکسان به هم متصل شدند، همچنين کاتد LEDهاي موجود در يک ستون نيز به هم اتصال داده شده‌اند.

حال ببينيم نحوه عملکرد اين روش چگونه است. در اين حالت جهت روشن کردن هر LED کافيست که سطري که آن LED در آنجا قرار دارد را به سطح ولتاژ مثبت اتصال داده و سپس ستون مربوط به همان LED را به زمين مدار وصل کنيم.

با اين روش ما توانستيم از تعداد سيمهاي مورد نياز جهت کنترل LEDها بکاهيم ولي در مقابل امکان کنترل همزمان تمامي سطرها را از دست داديم و در هر لحظه فقط و فقط مي توان LEDهاي موجود در يک سطر و يا يک ستون را کنترل نمود.

البته جهت نمايش نيازي هم به تمامي LED ها نبوده و مي‌توان توسط جاروب نمودن سطرها و يا ستون‌ها نيز به نمايش تصوير در تابلو روان پرداخت.

به هر حال در صورت عدم استفاده از روش فوق مدار را پيچيده خواهد کرد، مثلا براي کنترل LEDها موجود در تصوير روبرو حداقل بايد از طريق 41 سيم ماتريس را کنترل کرد. در حالي که با استفاده از روش ماتريسي فقط به 13 سيم نياز است. فقط در اين حالت برنامه شما کمي پيچيده خواهد شد.

روش جاروب ساده به دو صورت به کار برده مي شود :

- جاروب سطرها

- جاروب ستون‌ها

در جاروب سطرها LEDهاي موجود در سطر اول را روشن مي شود، سپس LEDهاي سطر دوم و . . . تا به سطر آخر برسد. دوباره همين كار را دوباره انجام مي شود .

در جاروب ستون‌ها LEDهاي موجود در ستون اول را روشن مي شود ، سپس LEDهاي ستون دوم و . . . تا به ستون آخر برسد. دوباره همين كار را دوباره انجام مي شود .

به يكبار جاروب كامل (خواه سطرها و خواه ستون‌ها) تازه سازي (Refresh) مي گويند.

توجه داشته باشين كه جاروب كردن علاوه بر كاهش سيم بندي سبب كم شدن پيچيدگي آن مي شود و باعث خواهد شد كه در هر لحظه تعداد كمتري از LEDهاي تابلو را روشن شود، در نتيجه ميزان مصرف جريان الكتريكي تابلو به ميزان قابل توجهي كاهش پيدا خواهد نمود.

همانطور كه گفته شد، جهت نمايش مناسب تصاوير متحرک بايد حداقل 24 تصوير در ثانيه نمايش داده شود. حال فرض كنيد مي خواهيم يک تابلو با 32 سطر طراحي كنيم و از جاروب سطري هم استفاده مي‌كنيم در اين حالت زمان نمايش هر فريم تصوير برابر با 41.6 ميلي ثانيه خواهد بود و در هر فريم 32 سطر جهت جاروب داريم پس زمان روشن بودن هر سطر برابر با 1.3 ميلي ثانيه خواهد بود.

اما پس از طراحي مدار و ساخت آن در پايان متوجه مي شويد كه نور LEDها بسيار كم تر از حالت معمولي است و حسابي متعجب خواهيد شد كه چرا با وجود استفاده از LEDهاي مرغوب نور تابلو روان تا اين حد كم است!

نكته اينجاست كه شما هر LED را فقط به مدت 1.3 ميلي ثانيه روشن نگاه مي داريد و سپس به مدت 31 برابر اين مدت خاموش نگاه مي داريد (به خاطر جاروب 31 سطر بعدي) يعني 1.3 ميلي ثانيه روشن و 40.3 ميلي ثانيه خاموش است. ودر واقع اثر نور LED در چشم به ميزان قابل توجه‌اي كاهش مي‌يابد.

جهت کم کردن این اثر و افزایش نور تابلو روان چند کار را مي‌توان انجام داد :

1- افزایش ولتاژ اعمالي به LEDها که معمولا این کار خطر سوختن LEDها در اثر هنگ کردن تابلو افزایش داده و همچنین از عمر مفید آن نیز خواهد کاست.

2- تقسیم تابلو به سگمنت هاي جداگانه مثلا براي مثال فوق تقسیم تابلو به 4 سگمنت 8 سطري. این روش مناسبي است ولي به پیچدگي نرم‌افزار و سخت‌افزار خواهد افزود و در تابلو هاي کوچک توصیه نمي‌شود.

3- جاروب یک در میان، این روش راه حل مناسبي در بر طرف نمودن مشکل فوق است. در عین حال که به پیچیدگي مدار منجر نخواهد شد و همچنین با تغییر ساده‌اي در الگوریتم برنامه مي توان از آن بهره برد.

ترتیب کار به این صورت است که هر فریم کامل را که در مثال فوق برابر با 32 سطر است به دو نیم فریم تقسیم مي شود . فریم اول شامل سطرهاي فرد (1،3،5،7،9،11،13،15،17،19،21،23،25،27،29،31) و فریم دوم شامل سطرهاي زوج (2،4،6،8،10،12،14،16،18،20،22،24،26،28،30،32) است. حال در هر جاروب فقط یکي از دو نیم فریم زوج ویا فرد جاروب مي شود . بدین ترتیب که یکبار نیم فریم فرد و بار بعد نیم فریم زوج و دوباره نیم فریم فرد و ... در این حالت چون در هر بار فقط 16 سطر جاروب میشوند لذا زمان روشن بودن هر LED بیشتر خواهد شد.

البته در این حالت در ثانیه فقط ما 12 تصویر (فریم کامل) نمایش داده‌ایم پس احتمالا بخاطر این کاهش مجددا مشکل لرزش تصویر را خواهیم داشت.

ولي نکته ظریفي هم این بین وجود دارد و آن هم این است که تعداد نیم فریم هاي نمایش داده شده در یک ثانیه همان 24 عدد است. در نتیجه لرزش تصویر منتفي خواهد بود.

تابلو روان سه رنگ :

اطلاعات فوق جهت ساخت یک تابلو روان تک رنگ ساده است ولي در تابلو روان سه رنگ به علت اینکه در هر پیکسل از سه رنگ استفاده مي شود مدار کمي پیچیده تر است. به این صورت که یک آي سي پردازشگر مرکزي به عنوان آي سي Master در نظر مي‌گیریم این برد Master به سه برد Slave (به تعداد رنگ ها) اطلاعات ارسال مي کند و این بردهاي Slave هستند که به LEDها متصل شده اند. بنابراین نحوه ي کار مدار به شکل زیر تغییر مي کند:

رایانه اطلاعات (کدهاي) تصویر را از طریق پورت سریال به میکرو Master ارسال مي کند و میکرو نیز اطلاعات دریافتي را در حافظه MMC ذخیره مي کند. هنگام نمایش اطلاعات آي سي Master اطلاعات ذخیره شده را به بردهاي Slave ارسال مي کند و برد هاي Slave نیز که از طریق درایورهاي جریان در سطر و ستون به LEDها متصل شده اند کدهاي دریافتي را به طور پیاپي به سطرها اعمال مي کنند و یک آي سي دکودر نیز به ترتیب هر ستون را فعال مي کند تا اطلاعات به صورت جاروب نمایش داده شوند.

طبق اطلاعات فوق ما سخت افزار تابلو روان سه رنگ را به 4 بخش تقسیم کرده و جداگانه به توضیح کامل هر قسمت مي پردازیم:

الف- برد Master

ب- بردهاي Slave

ج- برد تغذیه

د- برد نمایشگر(LED Dot-Matrix)

فصل دوم

طراحي برد Slave

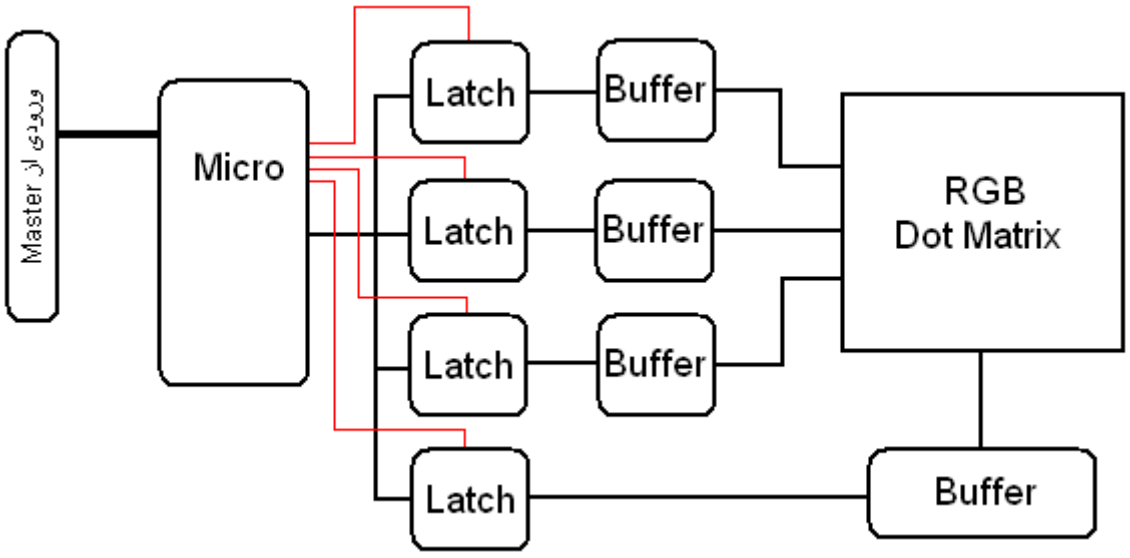
برد Slave در تابلو روان وظیفه ی اصلی نمایش اطلاعات را بر عهده دارد. به این دلیل که دیتا از Master برای نمایش به Slave فرستاده می شود و عمل اسکن و پردازش دیتا و در نهایت نمایش آن در این قسمت مدار انجام می شود.

در یک بررسی کلی نحوه‌ی نمایش داده های ورودی بر روی تابلو روان به صورت زیر است:

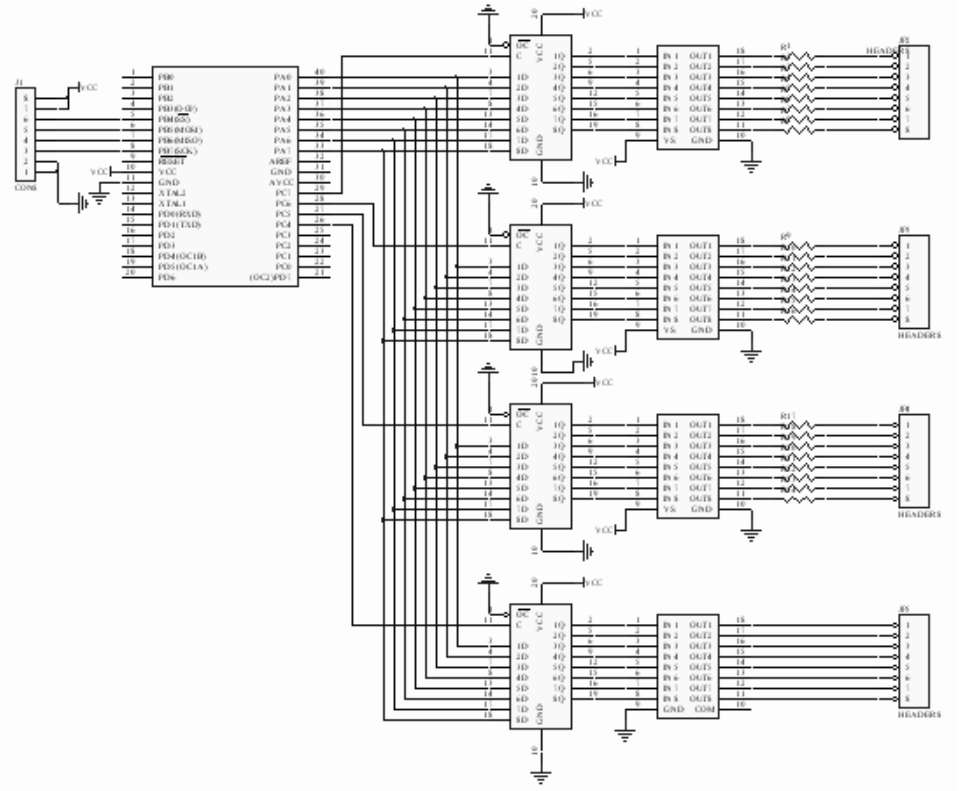
قسمت Master از طریق پروتکل SPI داده‌ها را به Slave ارسال می کند و پس از ارسال یک بسته کامل داده، Slave از طریق یک پورت به تناسب رنگ‌ها این اطلاعات را در برد LED نمایش می دهد که کنترل روشن شدن هر دسته از LEDهای قرمز سبز و آبی با آی‌سی های قفل کننده (Latch) انجام می شود. به دلیل اینکه برد LED جریان زیادی استفاده می کند برای جلوگیری از آسیب دیدن آی سی میکرو قبل از ارسال اطلاعات به تابلو، از بافر های جریان استفاده می کنیم.

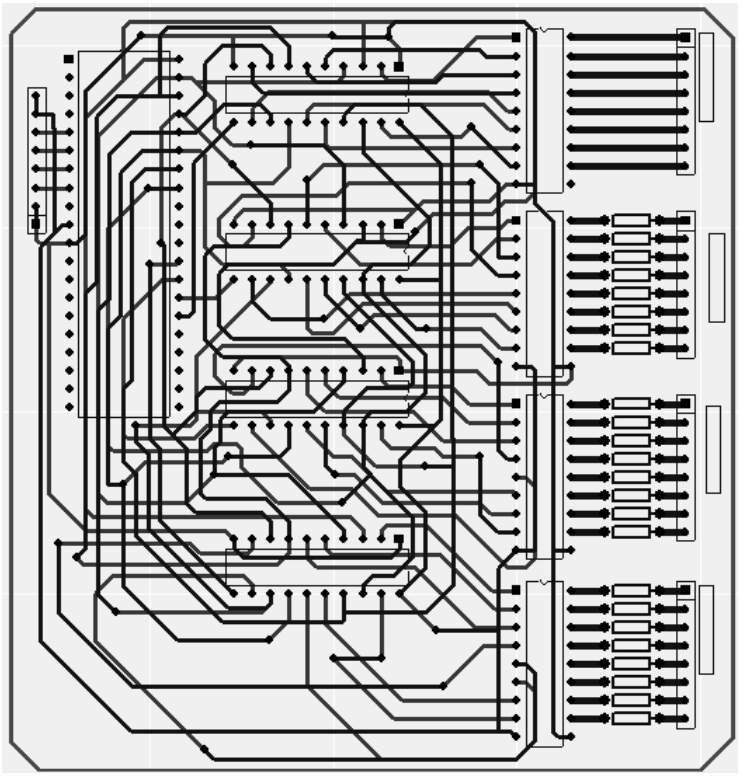
بلوك دیاگرام برد Slave:

شکل زیر بلوک دیگرام برد Slave را نشان می دهد:



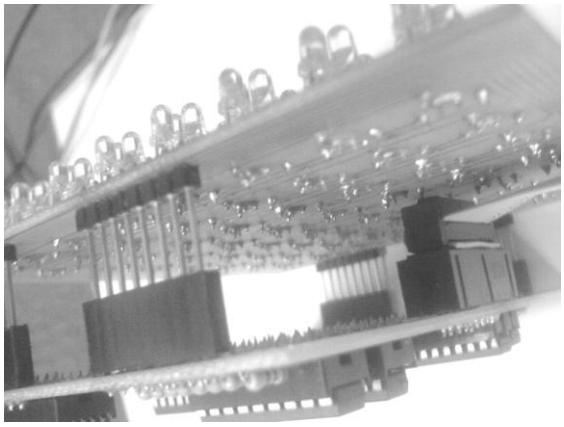
طراحی برد PCB:





ساخت عملی برد Slave:

برای ساخت تابلو روان 16 در 16 می توان با قرار دادن چهار تابلو روان 8 در 8 در کنار هم این کار را انجام داد که برای تست تابلو و جلوگیری از به وجود آمدن هزینه ی اضافی در صورت ایراد برد مدار چاپی، در ابتدا کار را با یک برد 8 در 8 انجام می دهیم. چون می خواهیم در انتها 4 برد 8 در 8 را در کنار هم قرار دهیم باید برد LED را جدا از برد Slave طراحی کنیم که در این حالت نیز تعدد سیم کشی ها و تعدد بردها کار را زشت و نامرتب می کند بنابراین می توان هر برد LED و Slave را به صورت پشت به هم (back to back) به هم کوپل کنیم و یک برد یکسره داشته باشیم.

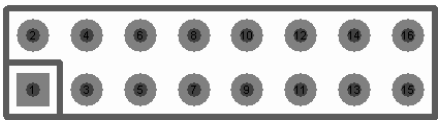


بدیهی است که در این حالت برد Slave باید از برد RGB کوچکتر باشد. همچنین می توان به جای استفاده کردن از سیم رابط بین برد Slave و RGB در طراحی ، پین های ارتباطی را روبروی هم قرار داد و از پین های نر و مادگی استفاده کرد تا وقتی دو فیبر روی هم قرار می گیرند به هم جفت شوند.

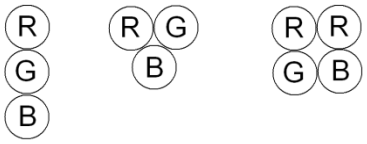
تذکر: درهنگام تست و راه اندازي تابلو LED باید به این نکته توجه داشت که سوختن يك LED در يك پيكسل به علت اتصال كوتاه شدن باعث خاموش شدن LEDهاي همرنگ در همان ستون مي شود و يا ممكن است LED هاي غير مرتبط نيز روشن شوند. در این شرایط با سوختن يك LED باید آن را حتما تعویض نمود تا مشکلات به وجود آمده فرد را گمراه نکند.

نکات عملی در طراحی PCB:

- ✓ کانکتور ورودی Slave برای استحکام بیشتر بهتر است 8 تایی دوبل باشد یعنی دو ورودی برای Vcc ، دو ورودی برای GND و 4 ورودی باقیمانده برای سیم های پروتکل SPI استفاده می شود.



- ✓ به منظور اینکه اطلاعات روی تابلو LED قابل تشخیص باشند از LED های ریز (3 mm) استفاده می کنیم.
- ✓ نحوه ی چیدن LEDها در کنار هم می تواند به صورت های زیر باشد:



آشنايي با فيبرهاي دورو و متاليزه:

فيبرهاي مدار چاپي به صورت هاي يك لايه، دو لايه و چندلايه ساخته مي شوند. در فيبرهاي يك لايه فقط يك طرف برد مس دارد و در دولايه ها هر دو طرف فيبر لايه مس دارد كه قابليت سيم كشي دو طرفه را فراهم مي‌كند.

همچنين اگر در فيبرهاي دورو، در هر Pad(سوراخ) و لايه بالايي و لايه پايييني به همديگر متصل باشند فيبر متاليزه است. اگر در قسمتي از طراحي نياز داشته باشيم كه سيم كشي به لايه ديگر برود از Via استفاده مي‌كنيم.

ساخت فيبرهاي متاليزه به ازاي هر سانتي متر مربع، 450 ﷼ و براي فيبرهاي يك رو تقريبا 1/3 فيبرهاي متاليزه هزينه دارد.

فيبر پروژه به دليل پيچيده بودن به صورت دورو طراحي و ساخته شده است. بنابراين در براي ساخت و لحيم كاري آن بايد به چند نكته اساسي توجه داشت:

- ✓ بايد هنگام طراحي به اندازه سوراخ ها دقت شود چون در هر سوراخ فيبر، سرب براي اتصال لايه بالايي به

پايـيـنـي وجـود دارد و در صـورت بـزرگـتـر كـردن آن بـا مـتـه سرب از بـيـن رفـتـه و اتـصـال قـطـع مـي شـود. پـس بـراي قـطـعـاتـي مـثـل Box و يـا پـيـن هـدر بـايـد از ابـتـدا سـوراخ هـا را بـزرگـتـر انـتـخـاب كـنـيـم.

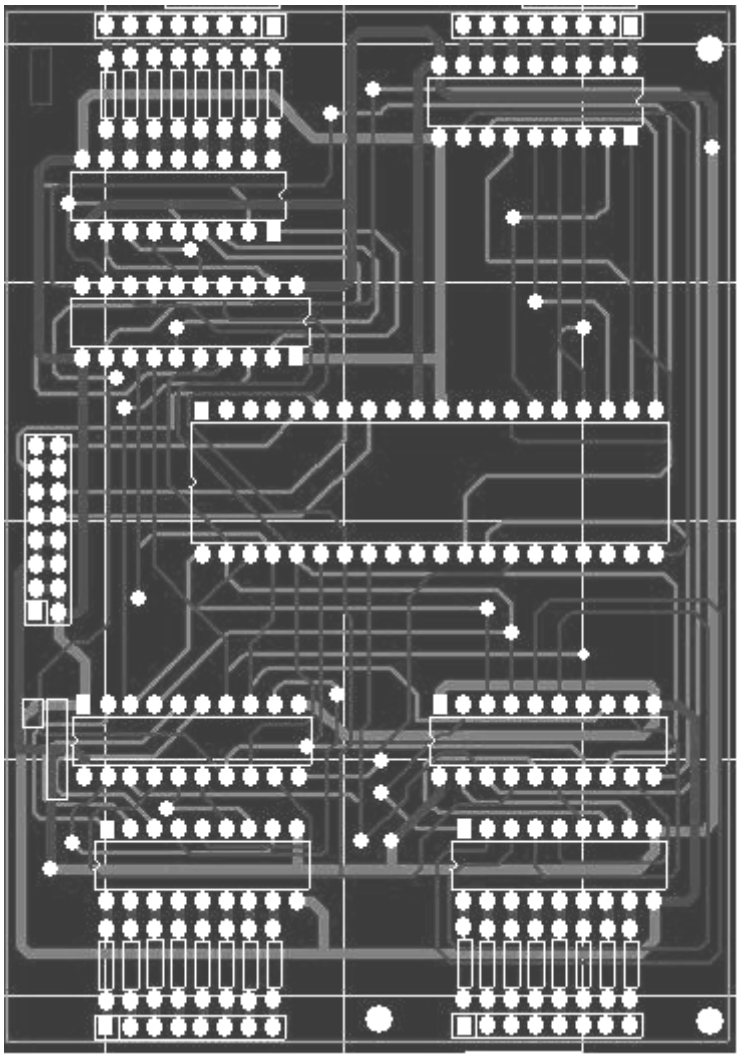
- ✓ زمـان لـحـيـم كـاري پـس از كـسـب اطـمـيـنـان قـطـعـه را لـحـيـم كـنـيـد چـون بـه دلـيـل وجـود سرب در Pad قـلـع بـه داخل سـوراخ نـفـوذ كـرده و قـلـع كـشي آن بـا قـلـع كـش هـاي مـعـمـولـي، سخت و بـراي قـطـعـاتـي بـا پـايـه زيـاد مـانـنـد پـيـن هـدر تـقـريـبـا غـيـر مـمـكـن است.
- ✓ هـمـچـنـيـن چـون آيسي‌هـا قـطـعـاتـي حـسـاس هـسـتـنـد و احـتـمـال سـوخـتـن آن‌هـا زيـاد است تـوصـيـه مـي شـود بـراي ايـن گـونـه قـطـعـات از سـوكـت اسـتـفـاده شـود. بـراي آيسي‌هـاي بـزگـتـر و گـران قـيـمـت مـانـنـد مـيـكـرو بـهـتـر است از پـايـه آيسي نـظـامـي اسـتـفـاده كـنـيـد.

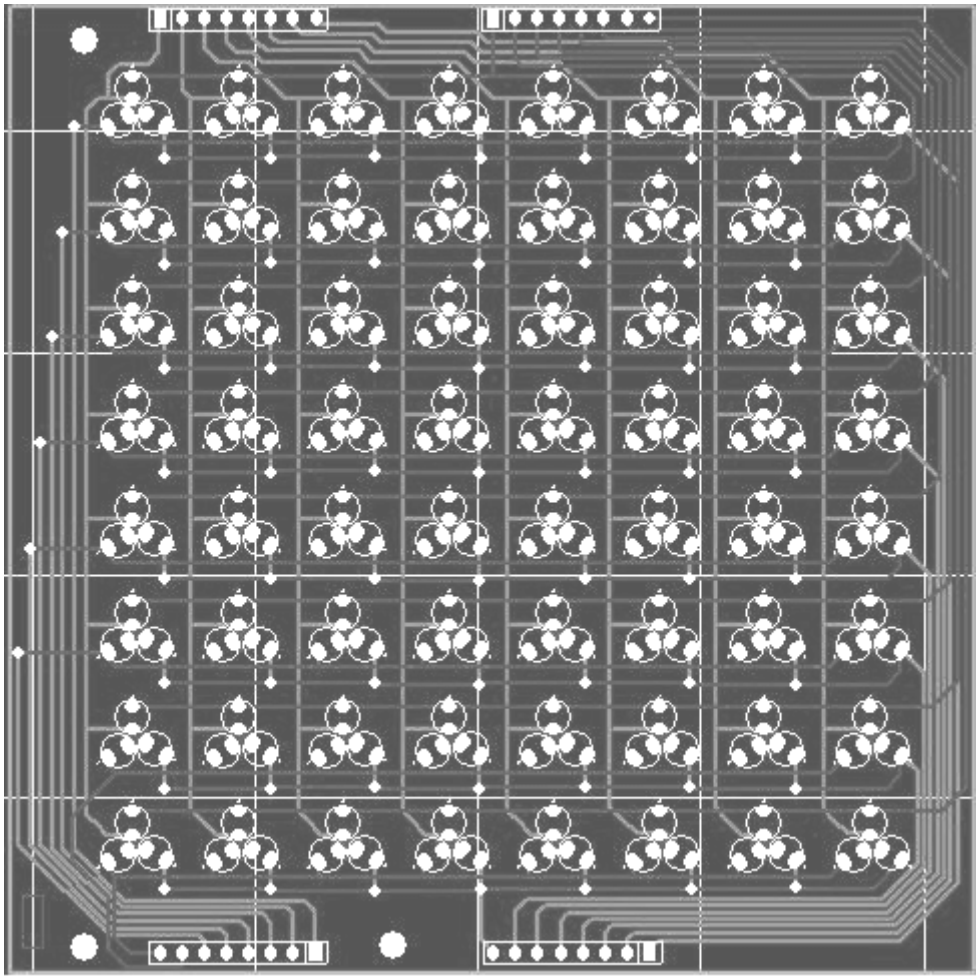
فـصل سـوم

تـوضيـح مـداري بـرد Slave و بـرنـامـه نـويـسي آن

در فصل قبل ما مدار Slave را به صورت دلخواه طراحی کردیم اما به دلیل اینکه هر برد Slave باید یک تابلو 8 در 8 را راه اندازی کند بنابراین باید 4 برد 8 در 8 را در کنار هم قرار داد تا یک برد 16 در 16 حاصل شود. اگر بخواهیم برد را به صورت صنعتی طراحی کنیم یک راهکار برای طراحی خوب به این صورت است که بردهای پردازنده و نمایشگر(LED) به هم کوپل شوند و از آن یک سیم رشته ای با اتصال IDC برای برد Master برده شود. با این کار اگر قطعه ای در یک بخش (Slot) معیوب شود به راحتی با تعویض کل آن Slot ، تابلو تعمیر می شود.

در شکل های زیر نحوه ی طراحی یک بخش نشان داده شده است . همچنین سه pad که در PCB تعبیه شده برای پیچ کردن دو برد روی هم است که دقیقا منطق همدیگر می باشند.



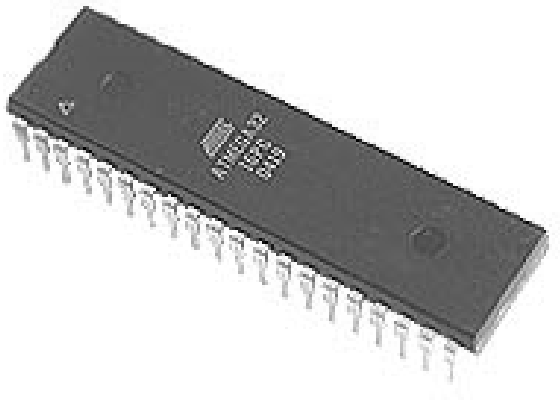


نحوه عملکرد و نکاتی در مورد قطعات استفاده شده در Slave:

در مدار Slave از سه آي سي بافر (UDN) و سه آي سي Latch و يك آي سي بافر (ULN) و يك آي سي AVR (Mega32) استفاده شده است.

: Micro

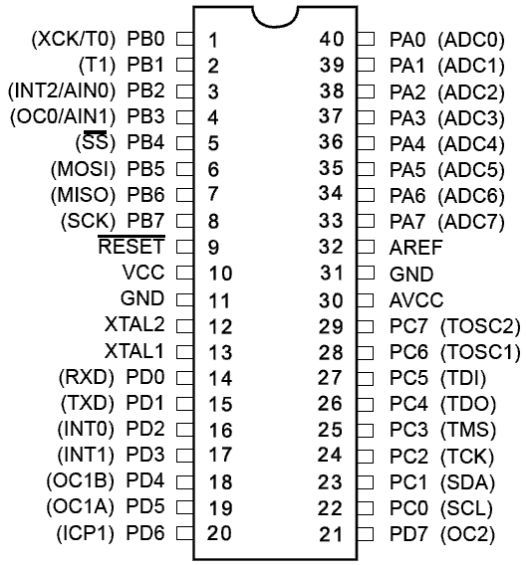
آی سی میکرو ATMEGA32 انتخاب شده است. مشخصات و قابلیت ها این آی سی در زیر آمده است:



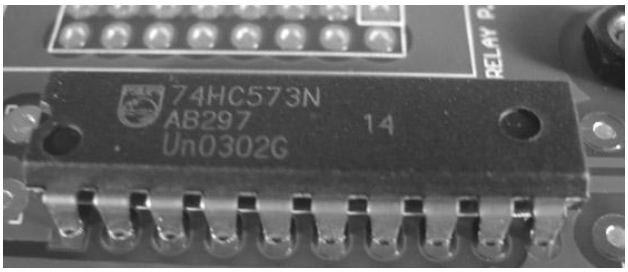
- ✓ کارایي بالا و توان مصرفي کم
- ✓ 32 رجیستر (ثبات) 8 بیتي
- ✓ سرعت با سقف 16 میلیون دستور در ثانیه در فرکانس 16 Mhz
- ✓ 32 کیلو بایت حافظه FLASH داخلي قابل برنامه ریزي با قابلیت ده هزار بار نوشتن و پاک کردن
- ✓ 2 کیلو بایت حافظه داخلي SRAM
- ✓ 1024 بایت حافظه EEPROM داخلي قابل برنامه ریزي با قابلیت صد هزار بار نوشتن و خواندن
- ✓ قابلیت ارتباط JTAG
- ✓ دو تایمر/شمارنده هشت بیتي
- ✓ یک تایمر/شمارنده شانزده بیتي
- ✓ چهار کانال PWM
- ✓ هشت کانال مبدل A/D ده بیتي
- ✓ یک مقایسه کننده آنالوگ داخلي
- ✓ WATCHDOG قابل برنامه ریزي با اسیلاتور داخلي
- ✓ ارتباط سریال براي برنامه ریزي: ISP
- ✓ USART سریال قابل برنامه ریزي
- ✓ داراي شش حالت SLEEP
- ✓ منابع وقفه داخلي و خارجي
- ✓ اسیلاتور داخلي RC
- ✓ کار با ولتاژ 4.5 تا 5.5
- ✓ فرکانس کاري 0 تا 16 مگاهرتز

✓ 32 خط داده ورودي و خروجي قابل برنامه ريزي

پايه هاي ميكروكنترلرATMega32 :

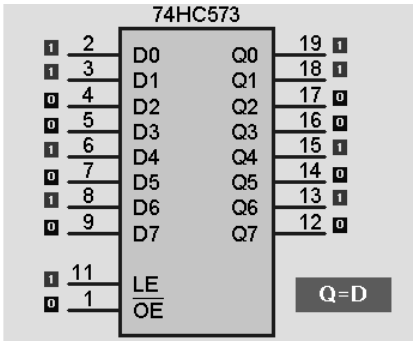


: Latch

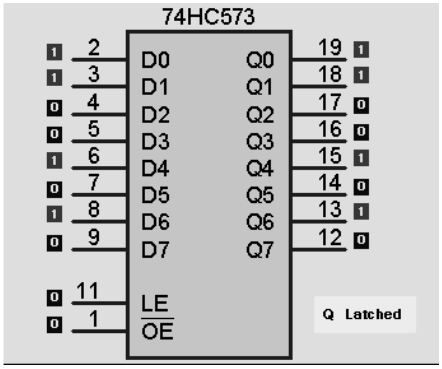


آي سي 74573 از خانواده TTLها است.اين آي سي 20 پايه دارد كه پايه 10 زمين (GND) و پايه 20 تغذيه (VCC) است. پايه هاي 2 تا 9 ورودي هاي D0 تا D7 و پايه هاي 19 تا 12 خروجي هاي Q0 تا Q7 مي باشند.از طرفي اين آي سي داراي دو پايه كنترلي به نام هاي LE كه همان فعال ساز لچ و OE كه فعال ساز خروجي هستند مي باشد.

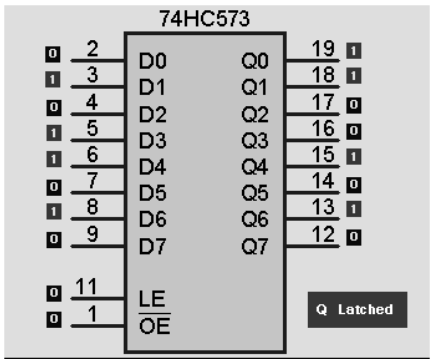
در اين آي سي تا زماني كه ورودي LE را در سطح يك نگه داريم مقادير ورودي هاي D0 تا D7 در خروجي ظاهر مي شوند ، و هر تغييري در ورودي باعث تغيير در خروجي هاي Q0 تا Q7 خواهد شد. در اين وضعيت اصطلاحاً مي گويند آي سي شفاف است يعني شما در خروجي، ورودیهاي آيسي را مي‌بينيد. در تصوير زير اين حالت نمايش داده شده است:



حال اگر پايه LE را دوباره به سطح صفر برگردانيم ديگر اطلاعات خروجي از ورودي تبعيت نمي‌كنند و آخرين وضعيت خود را حفظ مي‌كنند. به اصطلاح در اين حالت مي گويند كه آي‌سي اطلاعات ورودي را لچ كرده است. در تصوير زير اين وضعيت نشان داده شده است:



در تصوير زير حالتي نمايش داده شده است كه با اينكه اطلاعات ورودي تغيير كرده‌اند، ولي خروجي‌ها همان آخرين مقدار خود را حفظ كرده‌ند.

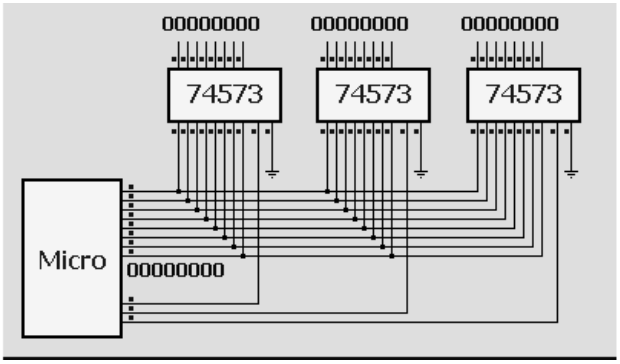


حالا كه با روند كلي كار اين آي‌سي آشنا شديد مي‌توانيم يك جمع بندي از آن داشته باشيم:

از اين ويژگي مي توان جهت گسترش پايه‌هاي ميكروكنترولر استفاده كرد به اين ترتيب كه ورودي چند آي‌سي لچ را به

هم متصل كرده و به يكي از پورتهاي ميكرو مي‌دهيم. سپس هر يك از پايه هاي LE لچ ها را به يك پين ميكرو اتصال مي دهيم و پايه OE را جهت فعال بودن خروجي ها به صفر وصل مي‌كنيم. حالا ميكرو اطلاعات هر لچ را در پورت متصل به ورودي لچ ها قرار مي‌دهد ولي فقط لچي اطلاعات را در خروجي خود ثبت مي كند كه پايه LE آن از طريق ميكرو يكبار به سطح يك رفته و مجدداً صفر شود. و اطلاعات ذخيره شده در ساير لچ ها بدون تغيير باقي مي‌ماند.

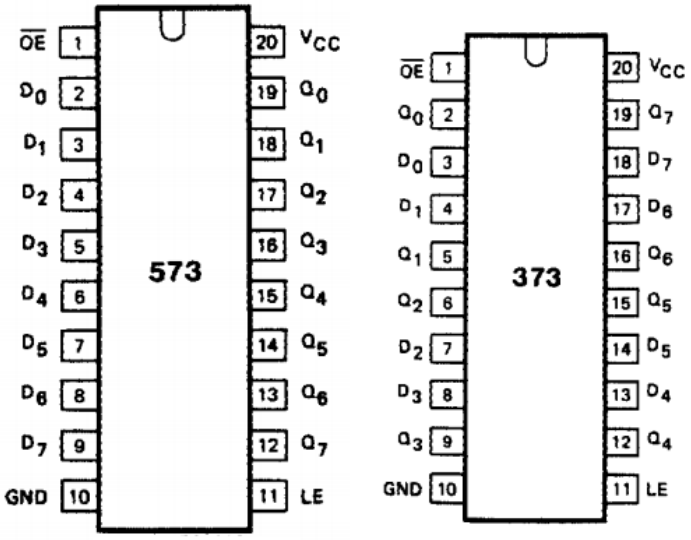
پس عملاً با فقط يك پورت ميكرو ميتوان چندين خروجي 8 بيتي ايجاد نمود.



همانطور كه در تصوير ديده مي‌شود ما با استفاده از 3 لچ مداري ساختيم كه ظرفيت ميكرو را به سه برابر افزايش داده است. اين مدار محدوديتي از بابت افزايش ظرفيت ندارد در صورتي كه تعداد لچ هاي مورد نياز شما از 8 عدد بيشتر است مي توانيد كار كنترل پايه هاي LE را توسط يك دكودر انجام داد. بعنوان مثال با 74154 مي توانيد از طريق 4 پين ميكرو 16 لچ را كنترل كرد.

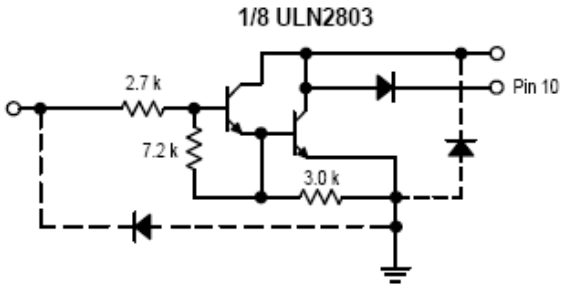
آي سي 74573 يا 74373

عملكرد اين دو آي سي شبيه هم مي باشد فقط يكي از مزاياي 74573 اين است كه ورودي ها در يك طرف و خروجي ها در طرف ديگر قرار دارند و در طراحي ساده تر است كه در طراحي و خريد قطعات بايد كاملا دقت شود. OE فعال ساز خروجي است و هميشه به GND وصل است. پايه LE فعال ساز لچ است كه با لبه بالا رونده فعال مي شود.



: Buffer

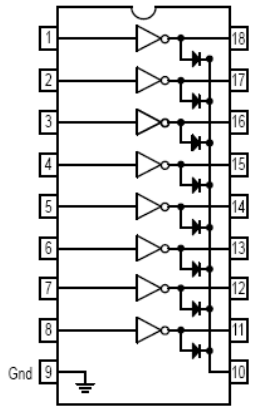
ULN2803 يك IC با 18 پايه است كه درون اين IC تعداد 8 ترانزيستور دارلينگتون NPN قرار دارد كه در ورودي بيس هر كدام هم مقاومتي معادل 2.7 كيلو اهم قرار داده شده كه باعث مي شود با اعمال يك ولتاژ 5 ولتي بتوان هر ترانزيستور را روشن نمود.



چناچه در تصوير بالا هم مشاهده كرديد علاوه بر ترانزيستورهاي دارلينگتون ديودهايي نيز جهت محافظت در مقابل ولتاژ معكوس كه معمولاً در بارهاي سلفي بوجود مي‌آيد نيز در اين IC در نظر گرفته شده است. از طرفي تمامي خروجي‌ها از طريق يك ديود به پايه شماره 10 وصل شده‌اند. كه بعنوان پايه مشترك خروجي استفاده مي‌شود.

مي‌توان از اين IC جهت درايو كردن خروجي‌هاي ميكروكنترولر در مدارات استفاده كرد.

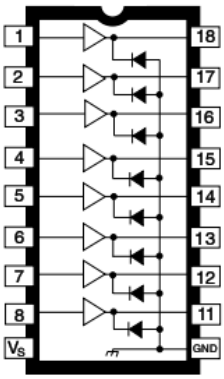
PIN CONNECTIONS



استفاده از این IC ها هم مقرون به صرفه بوده و هم از تعداد قطعات روي برد کم کرده و در نتیجه باعث ساده تر شدن برد مدار چاپي مي شود.

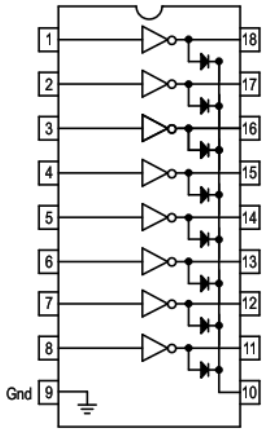
UDN2981-3

UDN2981A thru UDN2984A



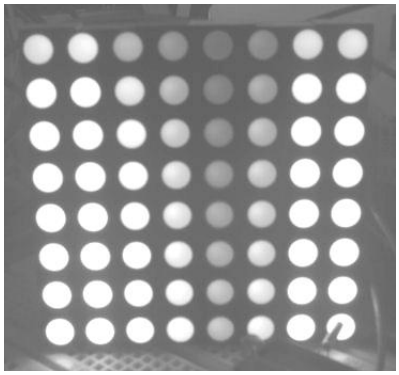
از این آي سي هاي بافر براي درايو کردن سطرها استفاده مي شود که براي هر رنگ از يك آي سي استفاده مي کنيم. يا مي توان گفت اين آي سي ها، آي سي هاي جريان دهنده هستند که پايه هاي 1 تا 8 ورودي و 11 تا 18 خروجي هستند و براي جريان دهي به آند LEDها به کار مي روند.

ULN2803



در اين آي سي خروجي NOT ورودي است و از آن براي درايو كردن ستون‌ها استفاده مي شود يعني براي جريان كشي از كاتد LED كه به صورت مشترك وصل شده اند. پايه هاي 1 تا 8 ورودي و 11 تا 18 خروجي هستند. پايه شماره 10 مي تواند آزاد باشد يا به Vcc وصل شود كه بهتر است به Vcc وصل باشد.

توضيحاتي در مورد برد LED:



برای ساخت یک تابلو روان سه رنگ (که در حقیقت با ادغام این رنگ ها، 64 رنگ به وجود می آید) باید از سه رنگ اصلی یعنی RGB (قرمز، سبز، آبی) استفاده کرد. در صنعت LED هایی به نام RGB LED وجود دارد که دارای 4 پایه هستند، یک پایه مشترک و سه پایه دیگر برای سه رنگ مختلف است که می توان با کنترل درصد روشنایی آنها، رنگ ها را تغییر داد. اما به دلیل اینکه در عمل این LED ها دارای ایرادات فراوان است و همچنین قیمت آنها بالاست از این LED ها

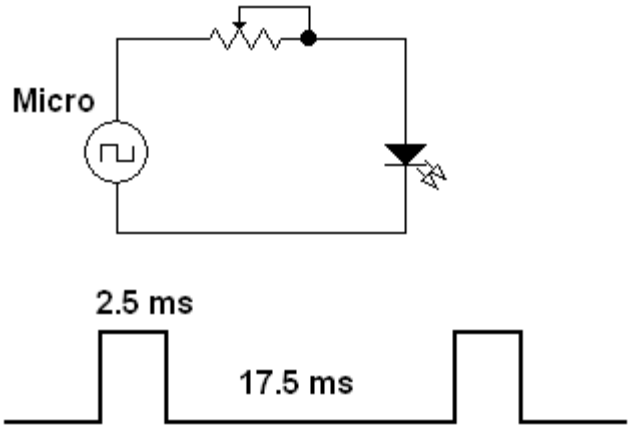
استفاده نمی کنیم. بنابراین برای ساخت رنگ های مختلف مجبوریم سه LED تک رنگ برای هر پیکسل به کار بریم.

تذکر: همانطور که مي دانيم اگر خروجي ميكرو (+5v) را مستقيما به LED وصل كنيم، در زمان روشن بودن آن جريان زيادي از LED عبور مي‌كند كه باعث سوختن آن مي‌شود. براي محدود كردن جريان از مقاومت هاي سري شده با LEDها استفاده مي‌كنيم، يعني براي هر سطر يك مقاومت.

چون ولتاژ كاري هر رنگ LED متفاوت است براي يكسان كردن رنگ، مقدار مقاومت ها را متفاوت انتخاب مي كنيم.

نحوه‌ي بدست آوردن مقاومت ها:

با توجه به اینکه در هر برد Slave 8 ستون داریم و در هر لحظه فقط یك ستون روشن است، نسبت روشن بودن به خاموش بودن LEDها 1/7 است. يعني در مدت زمان اسكن يك صفحه كه 20 ميلي‌ثانيه انتخاب شده، هر LED 2.5 ميلي‌ثانيه روشن و 17.5 ميلي‌ثانيه خاموش است، پس بايد يك پالس با عرض 2.5 ميلي‌ثانيه و دوره تناوب 20 ميلي‌ثانيه ايجاد سپس يك پتانسيومتر با هر LED سري كنيم. با اعمال پالس به پتانسيومترها و تغيير آن‌ها نور هر سه رنگ را در يك اندازه تنظيم مي‌كنيم. در نهايت با اهم متر مقدار هر پتانسيومتر تنظيم شده را اندازه گيري كرده و نزديك ترين مقاومت ثابت را به جاي آن قرار مي دهيم.



توضیح نحوه برنامه نویسی برای ساخت رنگ:

در تابلو های رنگی معمولا از روش PWM برای کنترل رنگ ها استفاده می کنند که برای این کار می توان از PWM تایمرهای میکرو و یا از آی‌سی‌های PWM استفاده کرد.

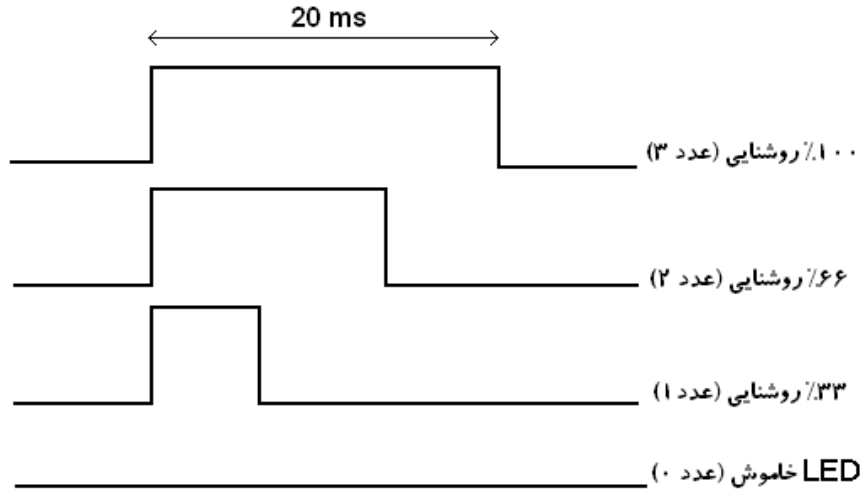
روش فوق به دلایلی در بازار ایران قابل استفاده نیست چرا که به دلیل جوابگو نبودن تایمرهای میکرو باید از آی سی های PWM استفاده کرد که این آی سی ها هم در بازار ایران بسیار کمیاب و گران قیمت هستند. بنابراین از تکنیکی استفاده می کنیم تا هم PWM ساخته شود و هم نیاز به سخت افزار پیچیده ای نباشد.

یک تابلوی 8 در 8 را در نظر بگیرید. این تابلو 64 پیکسل دارد که هر پیکسل آن شامل سه رنگ است.

هر پیکسل باید 4 سطح ولتاژ مختلف داشته باشد يعني با صفر ، 33% ، 66% و 100% نور روشن شود که با این حساب 64 رنگ متفاوت مي توان ساخت. بنابراين كدهاي رنگ كه قرار است ما در تابلو نمايش دهيم به صورت اعداد دو بيتي مي باشند كه به ترتيب براي درصد نورهاي فوق از صفر تا 3 هستند.

نحوه كار به اين صورت است كه ما زمان اسكن يك عكس را به 4 قسمت، تقسيم مي كنيم. پس برنامه را به صورتي مي نويسيم كه به ازاي هر كد رنگ، LED به همان مدت روشن باقي بماند. به عنوان مثال اگر كد رنگ ما 2 است LED در مدت 3/4 زمان روشن و در 1/4 زمان خاموش است.

تصوير زير اين مطلب را بهتر توضيح مي دهد:



براي نوشتن برنامه فوق كافي است مراحل زير را انجام دهيم:

هر 20 ميلي ثانيه يك بار از اطلاعات اصلي كپي گرفته و در يك متغير جديد مي ريزيم. هر 5 ميلي ثانيه نيز يك بار اطلاعات متغير جديد را بررسي مي كنيم اگر عدد بزرگتر يا مساوي يك بود LED را روشن و اگر صفر بود LED خاموش مي شود. همچنين بعد از نمايش از اطلاعات كپي شده يكبار عدد يك را كم مي كنيم تا براي 5 ميلي ثانيه بعدي آماده شود. با اين كار زمان روشن ماندن LED كنترل مي شود.

برنامه زير اين عمل را انجام مي دهد:

اطلاعات رنگ براي نمايش

```
#include <mega32.h>

unsigned char
R[64]={0,0,0,0,1,1,1,1,0,0,0,0,1,1,1,1,0,0,0,0,1,1,1,1,0,0,0,0,1,1,1,1,2,2,2,2,3,3,
3,3,2,2,2,2,3,3,3,3,2,2,2,2,3,3,3,3,2,2,2,2,3,3,3,3};

unsigned char
G[64]={0,1,2,3,0,1,2,3,0,1,2,3,0,1,2,3,0,1,2,3,0,1,2,3,0,1,2,3,0,1,2,3,0,1,2,3,0,1,
2,3,0,1,2,3,0,1,2,3,0,1,2,3,0,1,2,3,0,1,2,3,0,1,2,3};

unsigned char
B[64]={0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,2,2,2,2,2,2,2,3,3,3,3,3,3,3,3,0,0,0,0,0,0,
0,0,1,1,1,1,1,1,1,1,2,2,2,2,2,2,2,2,3,3,3,3,3,3,3,3};
```

```
unsigned char RC[64],GC[64],BC[64];
unsigned char shift=0x01,k,a,b,c,d=0x01,i,j;


void main(void)
{
// Input/Output Ports initialization
// Port A initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTA=0x00;
DDRA=0xff;
// Port B initialization
// Func7=In Func6=Out Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=T State6=0 State5=T State4=T State3=T State2=T State1=T State0=T
PORTB=0x00;
DDRB=0x00;
// Port C initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTC=0x00;
DDRC=0xff;
// Port D initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
```

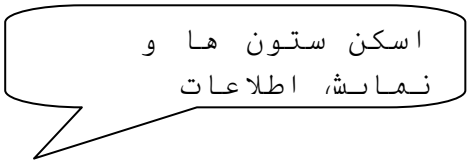
```
PORTD=0x00;
DDRD=0xff;

while (1)
    {
    // Place your code here
     for(i=0;i<64;i++)
     {
       RC[i]=R[i];
           GC[i]=G[i];
           BC[i]=B[i];
     }
       for(j=0;j<4;j++)
       {
         a=0;b=0;c=0;
           for(i=k;i<k+8;i++)
           {
                if(RC[i]>=1)
                {
                     a=a|shift;
                     RC[i]=RC[i]-1;
                }
                if(GC[i]>=1)
                {
                     b=b|shift;
                     GC[i]=GC[i]-1;
                }
```

کپي گرفتن از

حلقه 4 تايي براي 4 بار کم کردن عدد "1" از مقدار

کم کردن از کپي به صورت 8 بسته 8 تايي و ساختن متغيرهاي 8

```
            if(BC[i]>=1)
            {
                c=c|shift;
                BC[i]=BC[i]-1;
            }
            shift=shift<<1;
            if(shift==0)
                shift=0x01;
        }
          k=k+8;
          if(k>56)
                k=0;
    PORTD=d;
            PORTA=a;
            PORTC.6=1;
            PORTC.6=0;
            PORTA=b;
            PORTC.5=1;
            PORTC.5=0;
            PORTA=c;
            PORTC.7=1;
            PORTC.7=0;
    d=d<<1;
    if(d==0)
          d=0x01;
    }
};
```



}

فـصل چـهـا رم

طراحـي بـرد Master

قطعات مورد استفاده در برد Master:

ATMEGA 32

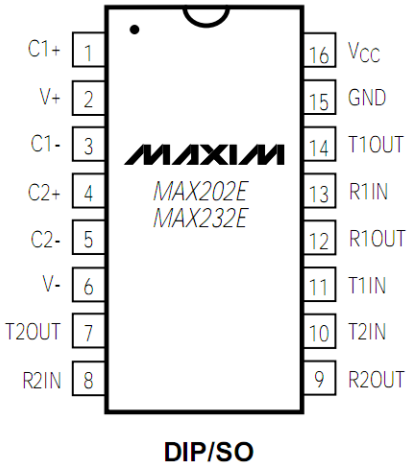
به دلیل اینکه این آی سی قبلا در برد Slave استفاده شده است توضیح تکراری در مورد این آیسی نمی‌دهیم.

MAX 232

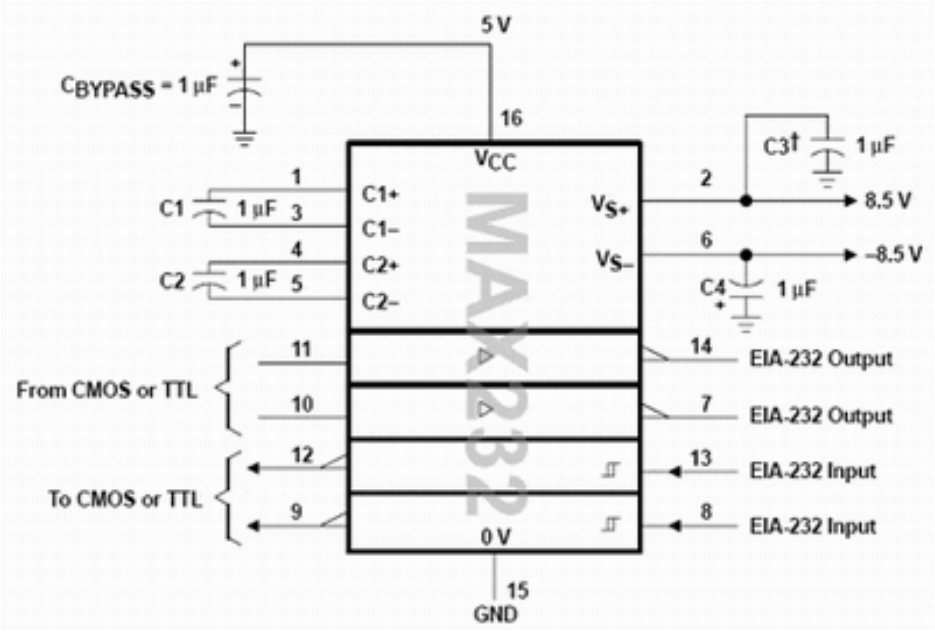
براي ارتباط سريال بين دو سيستم ديجيتالي بايد استانداردهاي اين سيستم ها در پذيرش مقادير صفر و يک منطقي يکسان باشند. در اين پروژه براي نمايش اطلاعات جديد بر روي تابلو روان ما نياز به اتصال پردازشگر تابلو روان(mega 32) به رايانه داريم. اما مشكلي كه در اين بين وجود دارد اين است كه سطوح ولتاژ آنها يكي نيست، يعني براي رايانه سطح "صفر" از 3+ تا 25+ ولت و سطح "يک" منطقي 3- تا 25- ولت (استاندارد RS232) مي باشد در حالي كه اين مقادير براي آي سي ميكرو صفر و 5 ولت (استاندارد TTL) است.

براي تبديل ولتاژ RS232 و TTL به يكديگر بايد از مبدل هاي ولتاژ استفاده كرد كه يكي از اين‌ها آيسي MAX232 و يا HIN232 مي باشد. كه MAX232 يک تراشه ي 16 پايه است كه شامل 2 فرستنده و 2 گيرنده است.

نماي كلي اين آي سي همراه با نام پايه هاي آن را در زير مي بينيد:



در ادامه نيز يک مدار نمونه را براي کار با اين آي سي مشاهده مي کنيد.



آي سي MAX233 :

اين آي سي نيز شبيه آي سي MAX232 مي باشد با اين تفاوت که تمامي خازن هاي مورد نياز براي مدار، داخل MAX233 گنجانده شده اند و ديگر نيازي به نصب خازن خارجي با اين آي سي نيست. البته با يد به اين نکته هم اشاره کرد که قيمت اين آي سي در مقايسه با MAX232 حدود 10 برابر بيشتر است.

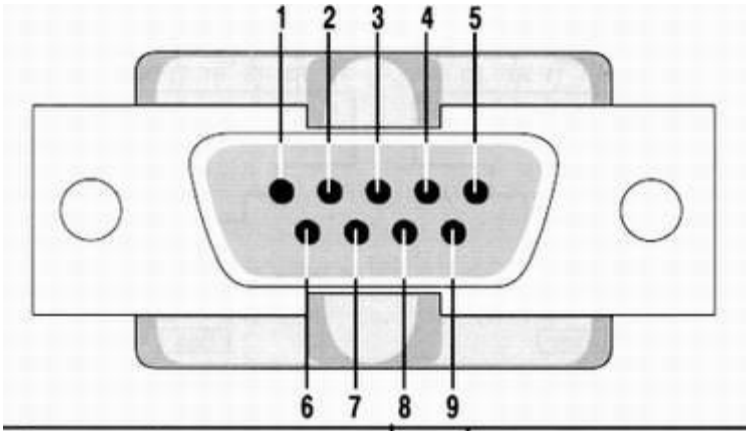
نکته: ارتباط سريال بين دو ميکرو در مسافت هاي طولاني و يا بشدت نويزي جوابگو نيست و براي اين کار مي توان از MAX485 استفاده کرد.

تذکر: در اتصال پورت سريال به ميکرو بايد پايه هاي پورت به صورت زير به همديگر وصل شوند:

1 → 4 → 6

7 → 8

پايه‌ي 5 GND، پايه 2 RXD و پايه 3 TXD مي باشد.



Pin	Signal	Pin	Signal
1	Data Carrier Detect	6	Data Set Ready
2	Received Data	7	Request to Send
3	Transmitted Data	8	Clear to Send
4	Data Terminal Ready	9	Ring Indicator
5	Signal Ground		

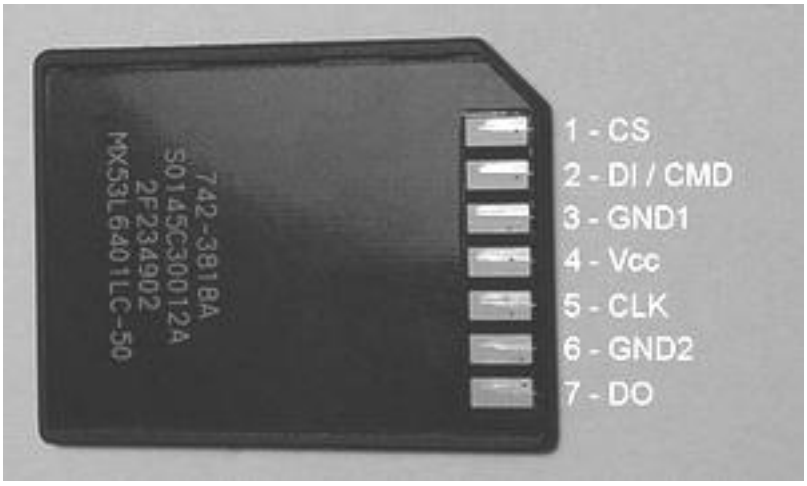
حافظه ي MMC (Multi Media Card)

هميشه يكي از مشكلاتي كه در بيشتر پروژه هاي الكترونيكي وجود دارد كم بودن حافظه است مثلا براي ساخت تابلو روان و ذخيره اطلاعات آماده نمايش، نياز به حافظه با حجم بالا و نوعي از حافظه مي باشد كه با قطع تغذيه اطلاعات آن پاك نشود. آي سي هاي حافظه زيادي در بازار وجود دارند كه داراي ظرفيت زيادي هستند ولي MMC به دليل حجم زياد و سرعت بالا و در دسترس بودن و نيز ارزان بودن از همه آي سي هاي موجود مقرون به صرفه تر مي باشد. و چون به صورت

کارت مي باشد داراي مزيت هايي نسبت به آي سي هاي حافظه است.

اين حافظه ها به صورت سريال و با پروتکل SPI ارتباط برقرار مي کنند که در ميکروکنترلرهاي AVR اين قابليت وجود دارد.

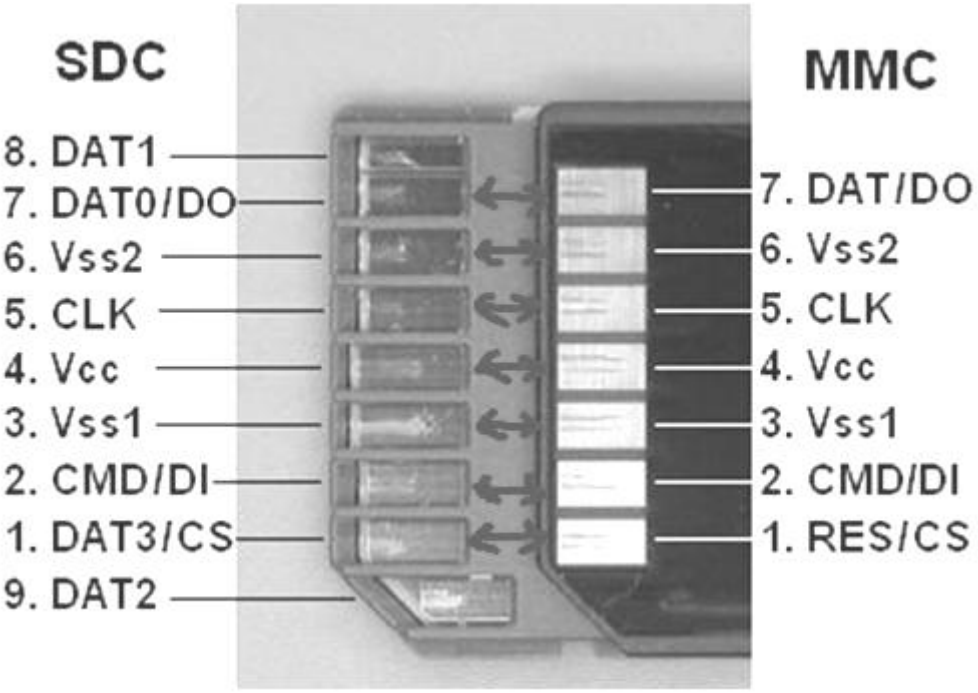
پايه هاي MMC :



همانطور که در شکل مي بينيد حافظه ي mmc 7 پايه دارد که نام هر پايه در زير آمده است و با استفاده از اين اطلاعات مي توان mmc را با پروتکل SPI به ميکرو وصل کرد:

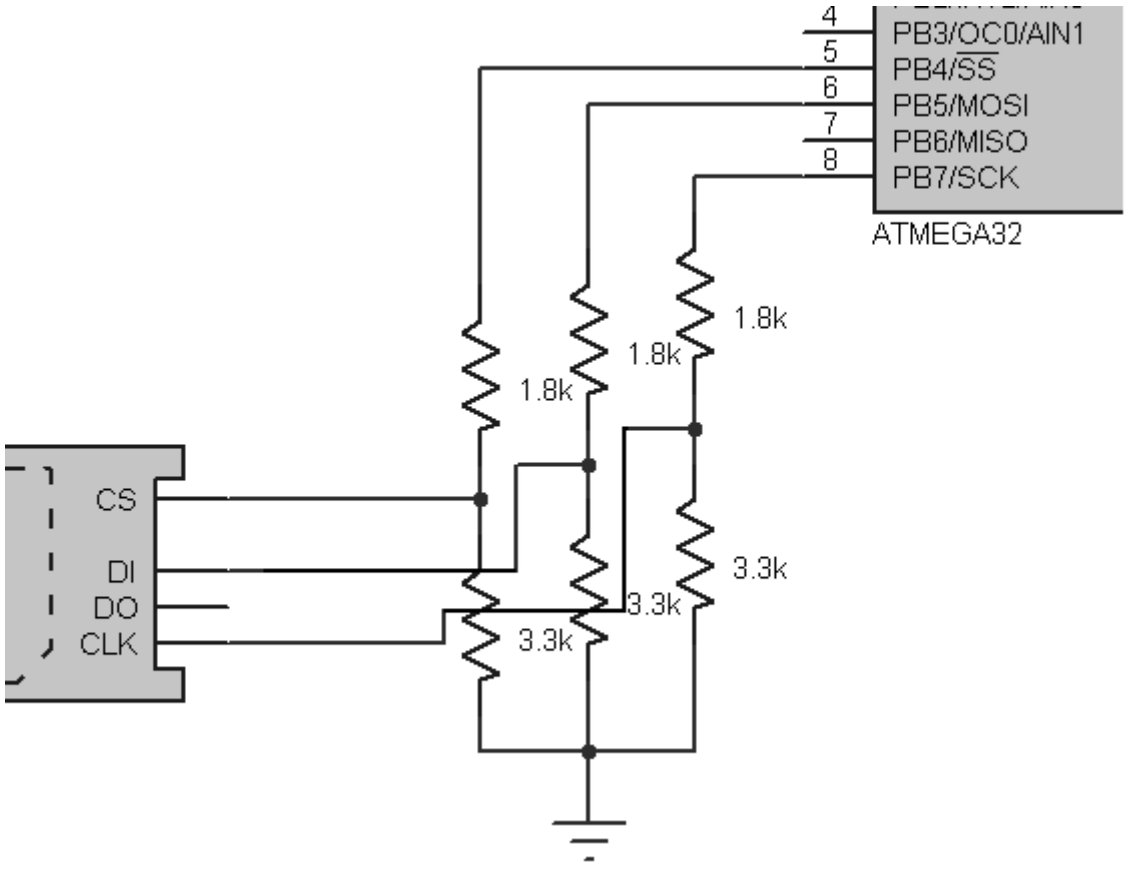
شماره پايه	عملکرد پايه
1	CS (Chip Select)
2	DI (Data In)
3	GND
4	VCC (3.3 v)
5	CLK (Clock)
6	GND
7	DO (Data Out)

همچنين در شکل زير اين حافظه با کارت هاي SD مقايسه شده که دو پين اضافه در SD به نام هاي DAT1,DAT2 مربوط به پروتکل سرعت بالاي SD است .



نحوه اتصال MMC به میکرو:

چون تغذیه MMC ، 3.3v است برای اتصال به میکرو که خروجی آن 5v است از یک شبکه تقسیم مقاومتی به شکل زیر استفاده می کنیم:

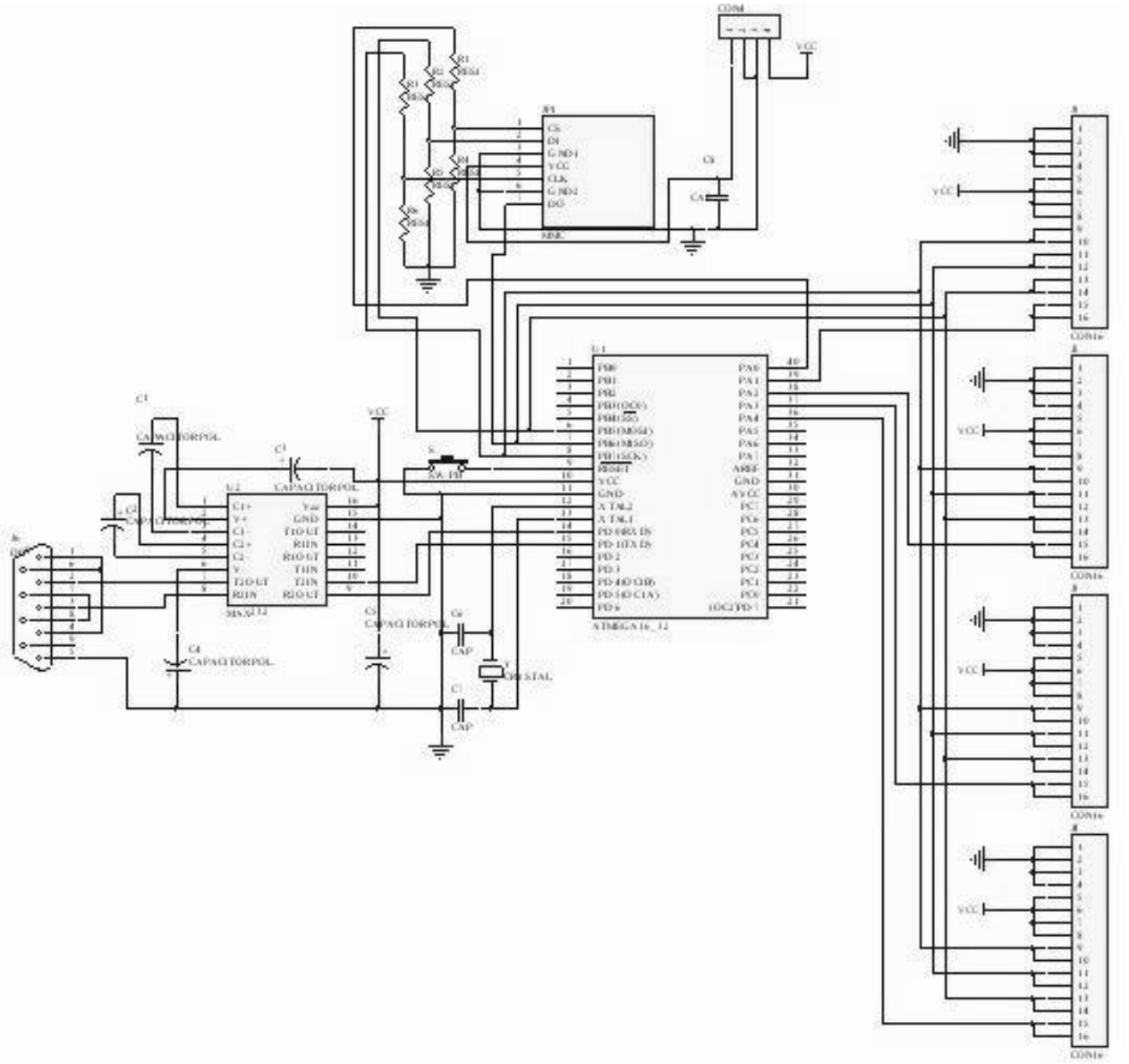


طراحی برد Master:

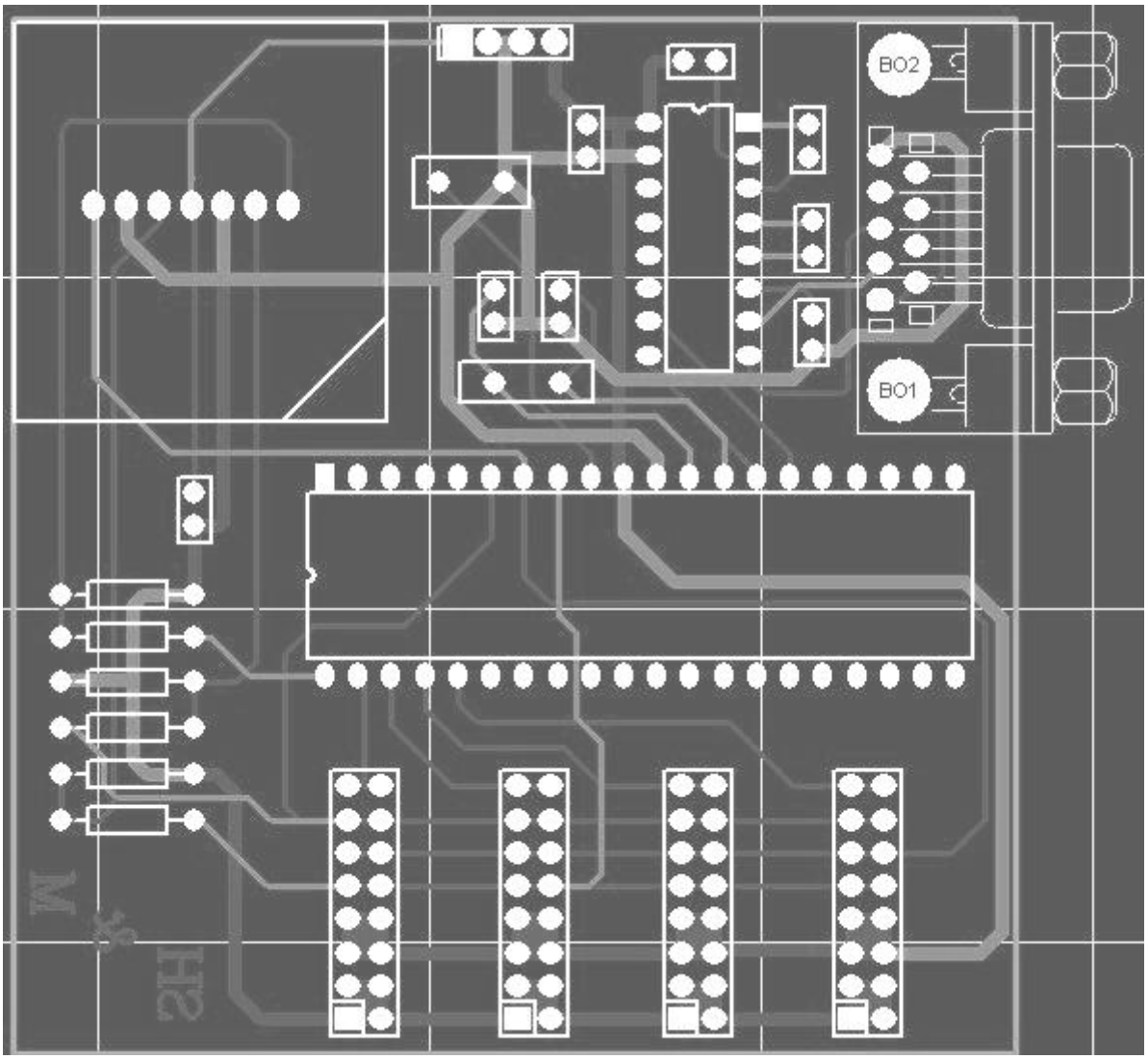
در برد Master به 4 عدد BOX برای 4 Slave نیاز است که باید در برد قرار داده شود همچنین برای اتصال به کامپیوتر یک پورت سریال باید به آن وصل شود و مهمتر از همه تغذیه مدار است که برای برد Master به تغذیه 3.3v و 5v و برای برد Slave به ولتاژ 5v نیاز است.

تغذیه مورد نیاز در این پروژه از یک منبع تغذیه کامپیوتر که ولتاژ DC صاف با جریان بالا را ارائه می دهد گرفته شده است.

طراحی شماتیک Master :

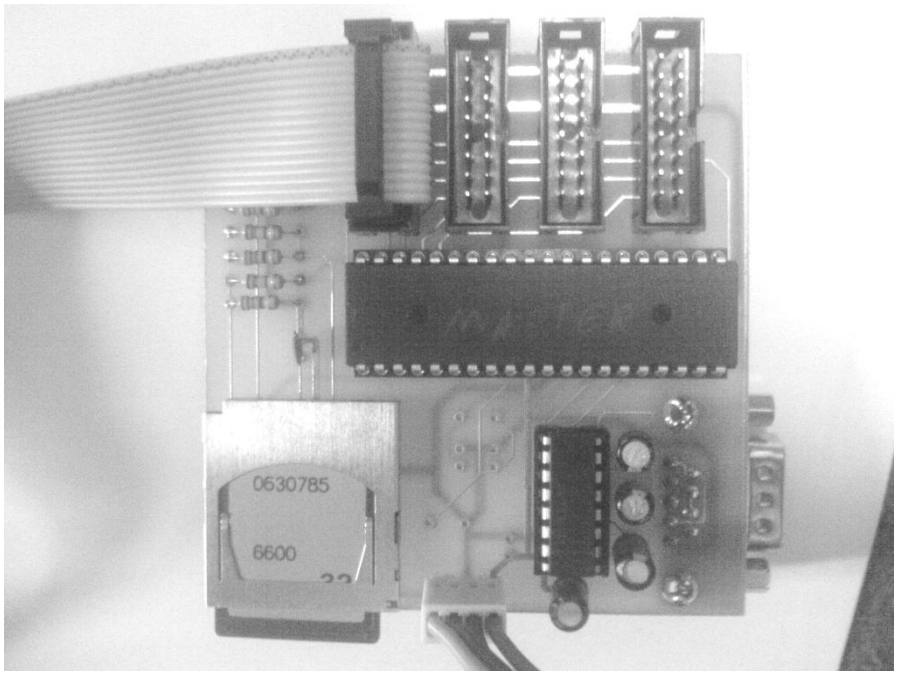


طراحی برد مدار چاپی Master:

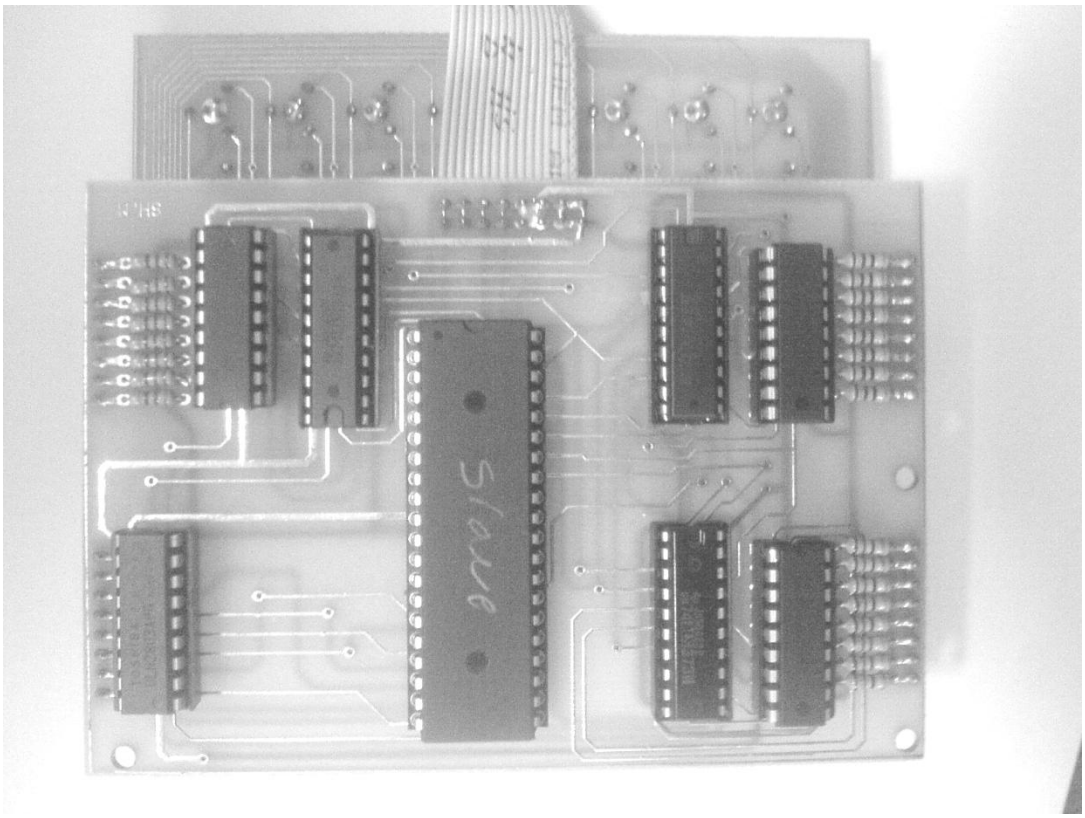


نمایی از سخت افزار آماده شده:

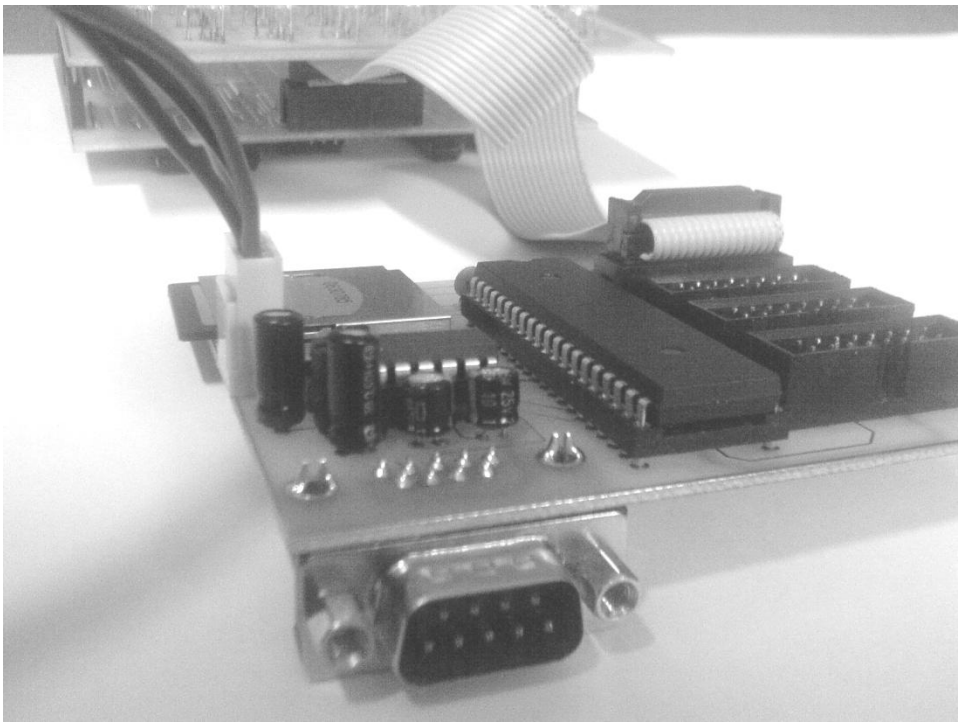
برد Master:



برد Slave:



ارتـبـاط Master و Slave :



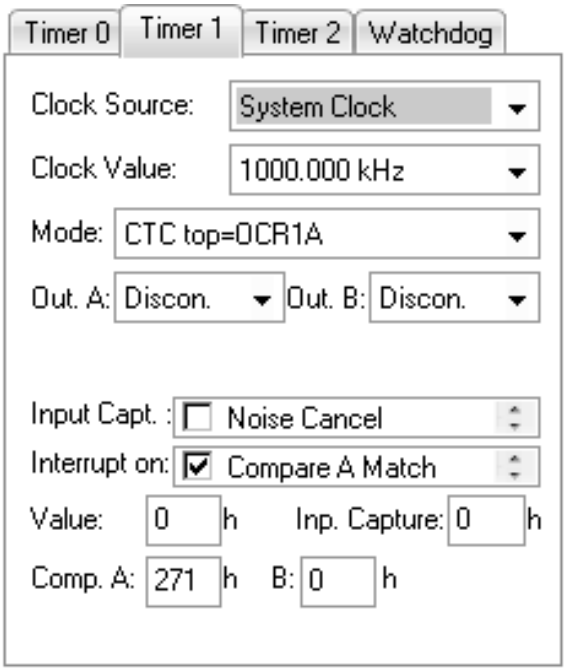
فصل پنجم

عملي كردن برنامه تابلوروان و نحوه محاسبات آن

براي داشتن تصوير ثابت هر تصوير بايد 50 بار در ثانيه نمايش داده شود بنابراين هر 20 ميليثانيه يك بار بايد صفحه به طور كامل Scan شود. همچنين براي درگير نشدن دائم CPU، اين كار بايد در وقفه ها انجام شود تا حلقه While(1) خالي بماند.

محاسبات Scan:

با توجه به اين كه هر Slave هشت ستون دارد Scan هر ستون 2.5 ms طول مي كشد و چون در مدت زمان روشن بودن هر ستون، كد رنگ بايد چهار بار كم شود بنابراين هر 625us يك بار از كد رنگ كم مي كنيم (2.5ms / 4=625 us). براي اينكه 625 us دقيق را به وجود آوريم از وقفه تايمر يك استفاده ميكنيم.



625(decimal) = 271(Hex)

- ✓ در تنظيمات زير فركانس CPU، 8 MHz است. براي استفاده از كريستال داخلي با فركانس هاي بيشتر از 1MHz بايد فيوز بيت ها را مطابق جدول زير تنظيم كنيم:

Internal Calibrated RC Oscillator Operating Modes

CKSEL3..0	Nominal Frequency (MHz)
0001	1.0
0010	2.0
0011	4.0
0100	8.0

برنامه کامل Slave :

(برنامه کامل آي‌سي‌ها در CD ضميمه شده با پايان نامه آورده شده است.)

```
/*****************************************************
This program was produced by the
CodeWizardAVR V1.24.6 Evaluation
Automatic Program Generator
© Copyright 1998-2005 Pavel Haiduc, HP InfoTech s.r.l.
http://www.hpinfotech.com
e-mail:office@hpinfotech.com

Project : Dot Matrix LED
Date    : 2010/01/04
Author  : Mohammad Keramati   &   Shahriyar Farhadi
Comments: copy is free
Chip type           : ATmega32
Program type        : Application
Clock frequency     : 8.000000 MHz
Memory model        : Small
External SRAM size  : 0
```

```
Data Stack size     : 512
*****************************************************/
#include <mega32.h>
unsigned char d=0x01,shift=0x01,i,R1,G1,B1,k,count,b=0xff,x;
unsigned char R[64],G[64],B[64],RC[64],GC[64],BC[64];

////////////////////////////////////////////////////////////////
void decrease (void)                //decrease each 625us
{
        R1=0;G1=0;B1=0;
        for(i=k;i<k+8;i++)
        {
            if(RC[i]>=1)
            {
                R1=R1|shift;
                RC[i]=RC[i]-1;
            }
            if(GC[i]>=1)
            {
                G1=G1|shift;
                GC[i]=GC[i]-1;
            }
            if(BC[i]>=1)
            {
```

```
                B1=B1|shift;
                BC[i]=BC[i]-1;
            }
            shift=shift<<1;
            if(shift==0)
                shift=0x01;
        }

            PORTC.6=0;
            PORTA=R1;
            PORTC.6=1;
            PORTC.6=0;
            PORTC.5=0;
            PORTA=G1;
            PORTC.5=1;
            PORTC.5=0;
            PORTC.7=0;
            PORTA=B1;
            PORTC.7=1;
            PORTC.7=0;
 }
////////////////////////////////////////////////////////////////////////
void copy(void)
{
```

```
        for(i=0;i<64;i++)
        {
            RC[i]=R[i];
            GC[i]=G[i];
            BC[i]=B[i];
        }
}

// Timer 1 output compare A interrupt service routine
interrupt [TIM1_COMPA] void timer1_compa_isr(void)
{
// Place your code here
    PORTD=d;
    decrease();
    count++;
    if(count == 32)  //copy each 20ms
    {
        count=0;
        copy();
    }
    b++;
    if(b%4 == 0)    //shift column each 2.5ms
    {
        d=d<<1;
```

```
            if(d==0)
                d=0x01;
            k+=8;
            if(k>56)
                k=0;
        }
}
// SPI functions
#include <spi.h>
void main(void)
{
// Input/Output Ports initialization
// Port A initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTA=0x00;
DDRA=0xff;

// Port B initialization
// Func7=In Func6=Out Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=T State6=0 State5=T State4=T State3=T State2=T State1=T State0=T
PORTB=0x00;
DDRB=0x40;
```

```
// Port C initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTC=0x00;
DDRC=0xff;

// Port D initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTD=0x00;
DDRD=0xff;

// Timer/Counter 0 initialization
// Clock source: System Clock
// Clock value: Timer 0 Stopped
// Mode: Normal top=FFh
// OC0 output: Disconnected
TCCR0=0x00;
TCNT0=0x00;
OCR0=0x00;

// Timer/Counter 1 initialization
```

```
// Clock source: System Clock
// Clock value: 1000.000 kHz
// Mode: CTC top=OCR1A
// OC1A output: Discon.
// OC1B output: Discon.
// Noise Canceler: Off
// Input Capture on Falling Edge
// Timer 1 Overflow Interrupt: Off
// Input Capture Interrupt: Off
// Compare A Match Interrupt: On
// Compare B Match Interrupt: Off
TCCR1A=0x00;
TCCR1B=0x0A;
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x02;
OCR1AL=0x71;
OCR1BH=0x00;
OCR1BL=0x00;

// Timer/Counter 2 initialization
// Clock source: System Clock
```

```
// Clock value: Timer 2 Stopped
// Mode: Normal top=FFh
// OC2 output: Disconnected
ASSR=0x00;
TCCR2=0x00;
TCNT2=0x00;
OCR2=0x00;

// External Interrupt(s) initialization
// INT0: Off
// INT1: Off
// INT2: Off
MCUCR=0x00;
MCUCSR=0x00;

// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=0x10;

// Analog Comparator initialization
// Analog Comparator: Off
// Analog Comparator Input Capture by Timer/Counter 1: Off
ACSR=0x80;
SFIOR=0x00;
```

```
// SPI initialization
// SPI Type: Slave
// SPI Clock Rate: 62.500 kHz
// SPI Clock Phase: Cycle Half
// SPI Clock Polarity: Low
// SPI Data Order: MSB First
SPCR=0x43;
SPSR=0x00;

// Global enable interrupts
#asm("sei")

while (1)
      {
       for(x=0;x<64;x++)
       {
              R[x]=spi( );
              G[x]=spi( );
              B[x]=spi( );
       }
      };
}
```

منابع :

1) www.ir-micro.com
2) www.eca.ir
3) www.kavirelectronic.ir
4) www.hlachini.com
5) forum.iranled.com
6) www.allegromicro.com
7) www.datasheetcatalog.com
8) www.atmel.com
9) www.alldatasheet.com
10) گزارشهاي پيشرفت پروژه