دانشکده مهندسی پزشکی

مقاله

طراحی و ساخت دستگاه کنترل صوتی بیماران مبتلا به بیماری حنجره

طراح، ساخت و نگارش:

مهندس حامید احمدی (لاوین)

تابستان ۱۳۹۰

و خدايي كه در اين نزديكي

است ...

فهرست عناوین

از دوستان و تمام کسانی که بنده را در به انجام رساندن این پروژه یاری نمودند نهایت تشکر و قدر دانی را دارم؛ بالخصوص دوست عزیزم، مهندس جسیم امینی که در کُدنویسی بسیار مرا همراهی کرد.

حامید احمدي، شهریور ۹۰

چکیده

انسان موجودی اجتماعی است و گفتار ابزاری جهت ابراز ارتباط جمعی می باشد که فرد از طریق آن با جامعه ی پیرامون خود ارتباط برقرار می کند. این ابزار ارتباط جمعی مانند هر ابزار دیگری گاها با مشکلاتی مواجه می شود. به این مشکلات اختلالات گفتاری اطلاق می شود که نتیجه ی آن تکلم غیر عادی و ناهنجار است.

تکلم غیر عادی و ناهنجار، تکلمی است که با تکلم و گفتار عامه ی جامعه تفاوت فاحش دارد که این تفاوت باعث جلب توجه ی افراد جامعه ی پیرامون شخص شده و ارتباط شخص را با آنان مختل می کند. این جلب توجه باعث کاهش اعتماد به نفس گوینده می شود و لذا در گفتار احساس ناراحتی و خستگی می کند.

انواع اختلالات گفتار عبارتند از: ۱- اختلالات صوتی، ۲- اختلالات روانی، ۳- اختلالات تاخیری گفتار و زبان، ۴- اختلالات زبانی، ۵- اختلالات تولیدی، ۶- اختلالات تشدیدی.

در این پروژه هدف کنترل اختلالات صوتی جهت بهبود و کاهش اختلالات گفتاری در این حوزه است. اختلالات صوتی خود انواع مختلف دارد که در اینجا هدف، کنترل پر کاری، بلندی و زیر و بمی صداست.

گاها مشاهده شده است که گوینده به دلیل پر گویی و یا بلندی گفتار و گفتار پی در پی در یک بازه ی زمانی پیوسته دچار خستگی و ناراحتی در سیستم گفتاری خود می شود و یا گاها دیده شده است که شخص گوینده ناخواسته زیرتر و یا بمتر از حد نرمال خود سخن گفته و باعث جلب توجه ی دیگران گردیده است.

لذا ضرورت طراحی و ساخت دستگاهی پیدا گردیده که سه مورد فوق یعنی پرگویی، بلند گویی و زیر وبمی نامتناسب را تشخیص و کنترل نماید و به بیمار و پزشک گزارش دهد.

فصل اول :

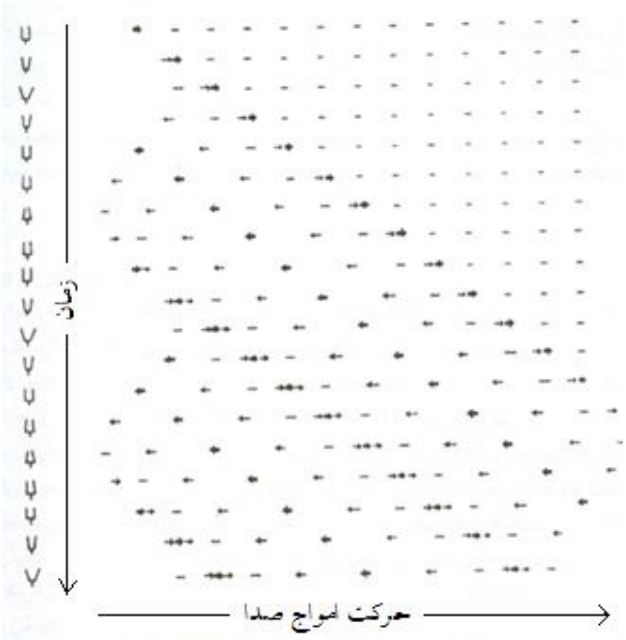
سیستم گفتار

سیستم گفتار:

سیستم گفتار را می توان از لحاظ ساختاری زیر مجموعه ای از سیستم تنفسی دانست چرا که تمامی اندام هایی که در تولید گفتار نقش دارند همان اندام های تنفسی هستند. تفاوت فاحش سیستم گفتار با سیستم تنفسی در ادراک گفتار در قشر مغزی است. غایت نهایی سیستم گفتار تولید صوتی است به گونه ای که منجر به ارتباط جمعی گردد.

۱-۱- صوت:

صوت یک موج مکانیکی طولی است که از ارتعاش مولکولهای هوا در راستای انتقال صوت تولید می شود. به عبارتی دیگر، موج صوتی با ایجاد رقت و تراکم مولکول های هوا ایجاد و انتقال می یابد. هر رقت و تراکم مولکولی در هوا موجب رقت و تراکم های دیگر می گردد، بدین معنی که هنگامی که یک لایه از مولکول های هوا به جلو رانده می شود، این لایه به نوبه ی خود لایه ی دیگری را به جلو می راند و خود به حال اول برمی گردد و با ادامه ی این روند صوت منتقل می شود.

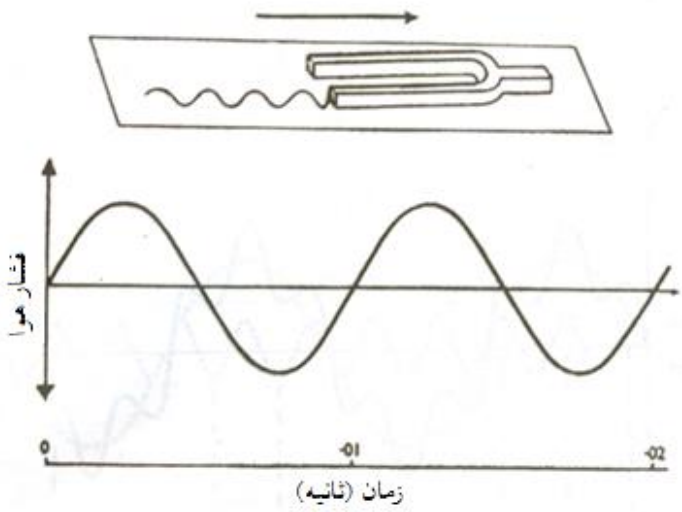


شکل ۱-۱ نمایش حرکت مولکولهای هوا

هر رقت و تراکم یک سیکل نام دارد . تعداد سیکل در ثانیه تواتر یا بسامد (فرکانس) نامیده می شود. هر قدر بسامد بیشتر باشد صدا به اصطلاح زیرتر است و هر قدر بسامد کمتر باشد صدا اصطلاحا بمتر است.

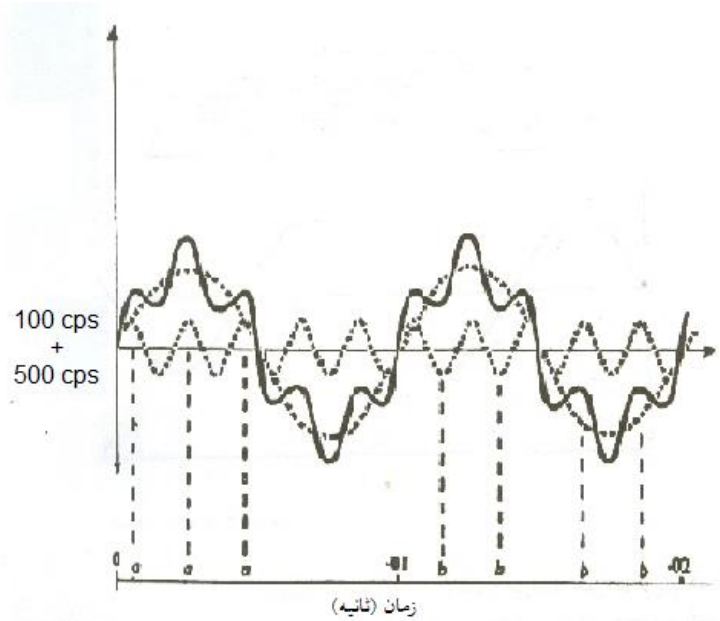
جابجایی یا ارتعاش مولکول های هوا در تمامی جهات صورت می گیرد و بسته به مقدار انرژی موجود، هر لایه از مولکول ها مسافتی را طی می کند. به سخن دیگر هر چه انرژی بیشتر باشد مسافتی را که موج می پیماید بیشتر است. طول مسافتی را که هر طبقه از مولکولهای هوا طی نموده و دوباره به جای اولیه خود برمی گردد، دامنه ی نوسان می نامند. هر چه این مسافت زیادتر باشد صدا بلندتر است. بلندی صدا را با زیر و بمی آن نباید اشتباه کرد، زیرا بلندی صدا مربوط به مقدار انرژی ولی زیر و بمی مربوط به تعداد سیکل ارتعاش

در ثانیه است. بنابراین صدایی ممکن است بم ولی بلند باشد و بلعکس صدایی دیگر ممکن است زیر ولی کوتاه باشد.

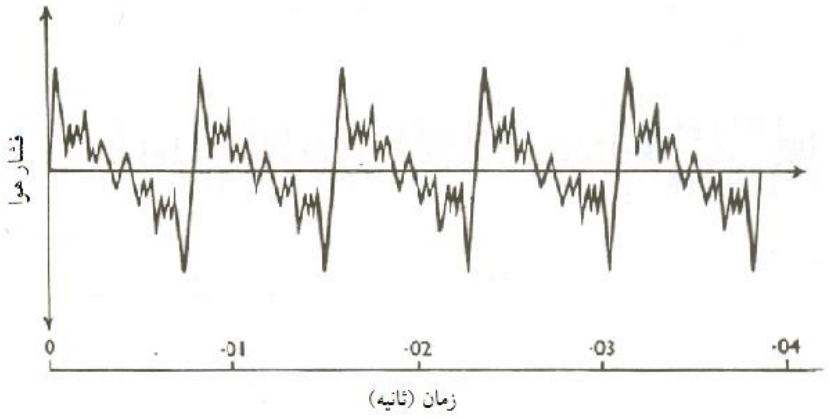


شکل ۱-۲ نمایش یک موج بسیط (سینوسی)

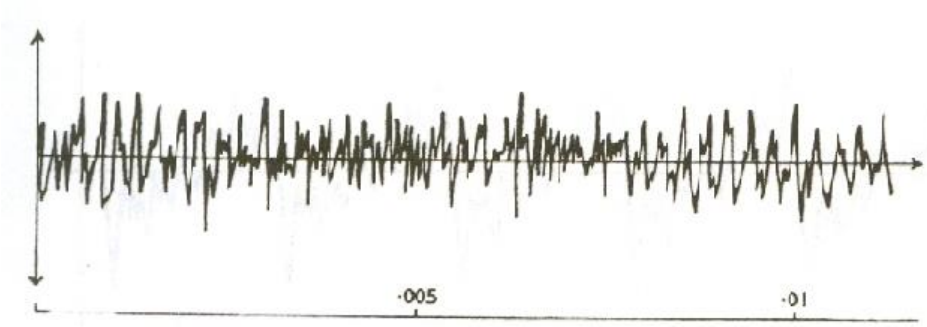
موج صوتی را از لحاظ چگونگی ساخت می توان به دو نوع عمده تقسیم کرد: بسیط و مرکب.موج بسیط یا سینوسی معمولا از یک موج منظم که طرح یک سیکل آن عینا تکرار می گردد تشکیل می شود، مانند صدای دیاپازون. موج مرکب یا پیچیده از ترکیب تعداد زیادی موج بسیط با بسامدهای گوناگون به وجود می آید.موج های مرکب خود بر دو نوع تقسیم می شود: منظم و نامنظم.



شکل ۱-۳ یک موج مرکب منظم



شکل ۱-٤ یک موج منظم بسیار پیچیده



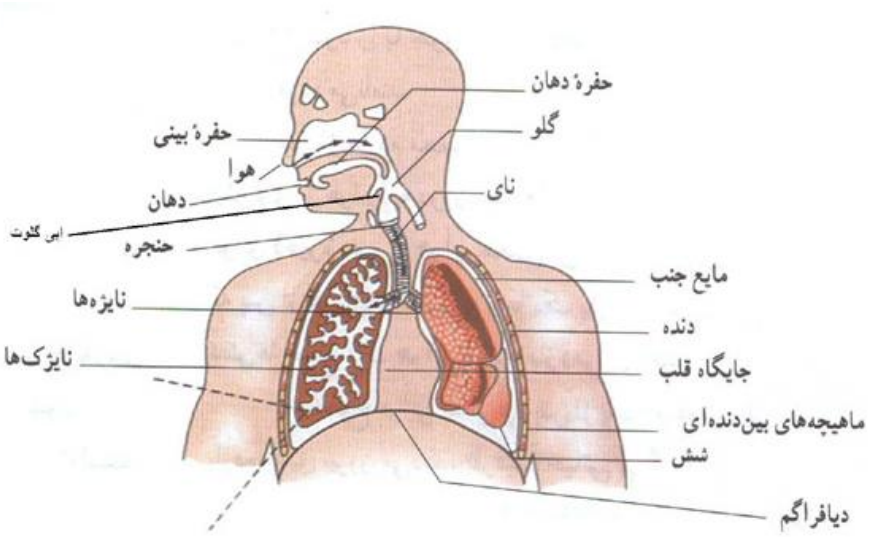
شکل ۱-۵ یک موج مرکب نامنظم (نوفه)

۱-۲- اندام های گفتار:

تولید صوت و تولید آواهای گفتار دو پدیده کاملا متفاوت هستند. تولید صوت در بین انسانها و حیوانات مشترک است و این در حالی می باشد که تولید آواهای گفتار تنها مختص انسانهاست، لذا به نظر می رسد که اصطلاح اندام های صوتی مناسبتر باشد ولی از آنجا که در تمامی متون آواشناسی، اصطلاح اندام های گفتار به کار رفته لذا در این جا ما نیز از همین اصطلاح استفاده می کنیم.

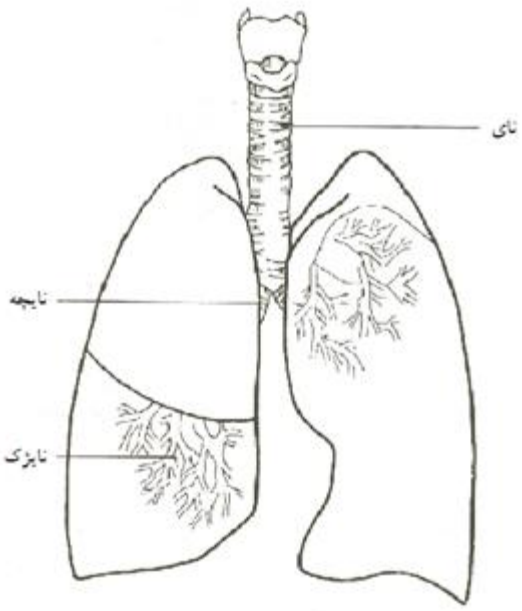
اندام هایی که درگیر تولید صوت و آواها هستند نقش مهم زیستی را دارند، لذا وظیفه تولید صوت و آواها برای این اندامها یک نقش ثانویه است نه نقش اولیه و اصلی.

اندام های گفتار عبارتند از: ششها، حجاب حاجز، نای، حنجره، تار آواها، گلوگاه، حفره بینی، دهان، کام، زبان، دندانها و لب ها.



شکل ۱-٦ اندام های گفتار

ششها: دو اندام اسفنجی هستند که در درون قفسه سینه جای گرفته و به وسیله نایچه به نای مربوط می شوند. جدار ششها از بافت های اسفنجی که دارای قابلیت ارتجاعی دارند تشکیل شده است به طوری که اگر شش فشرده شود می تواند دوباره به حالت اول برگردد. در واقع شش همچون یک بادکنک پر از هوا عمل می کند.



شکل ۱-۷ ششها

حجاب حاجز:

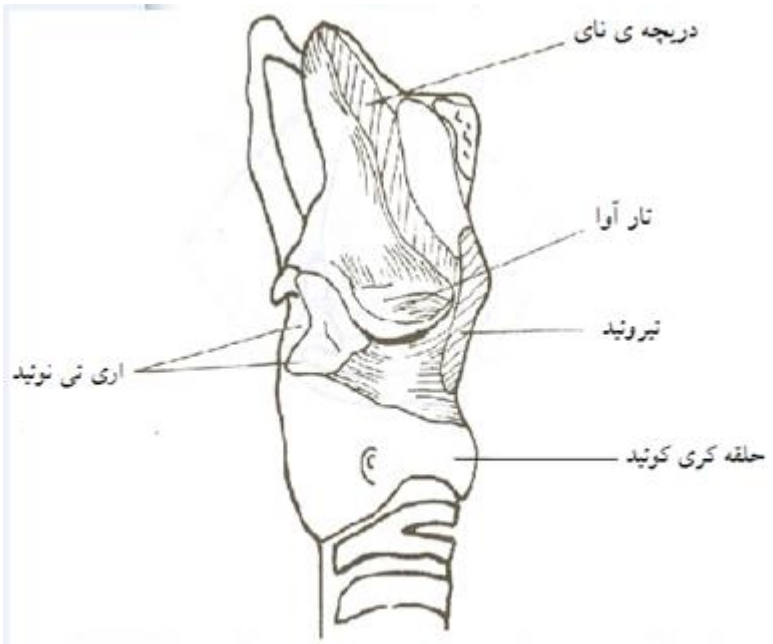
این اندام یک پرده ماهیچه ای گنبدی شکل است که قفسه سینه را از شکم جدا می سازد. در صورت انقباض حجاب حاجز (دیافراگم) قسمت بیرونی آن که به طرف قاعده ششهاست بالا رفته و حجم قفسه سینه را کاهش می دهد که این امر باعث فشرده شدن ششها شده لذا فشار داخل ششها از هوای بیرون بیشتر شده و هوا از ششها خارج می شود که به این فرآیند بازدم می گویند. با انبساط دیافراگم (پایین آمدن دیافراگم)، بر حجم قفسه سینه افزوده شده و به دلیل خاصیت اسفنجی باعث می شود که ششها دوباره به حالت اول برگردند. در این هنگام فشار هوای درون ششها کمتر از فشار هوای بیرون است و در نتیجه هوای خارج به درون ششها مکیده می شود که این عمل دم نام دارد.

نای:

لوله ای است که ششها را به گلو وصل می کند و از حلقه های ناقص غضروفی تشکیل یافته است. حلقه ها به وسیله ی بافت های ماهیچه ای به یک دیگر متصل گردیده اند. قطر نای بین ۲ تا ۲٫۵ سانتیمتر و طول آن از محل اتصال نایچه ها تا حنجره حدود ۱۱ سانتیمتر می باشد. نای را می توان همچون یک شلنگ فرض کرد.

حنجره:

آخرین حلقه فوقانی نای، یک حلقه کامل و از دیگر حلقه ها سخت تر و سنگین تر و طول آن بیشتر است که به آن کری کوئید که قاعده حنجره است گفته می شود. در واقع حنجره باکسی است که تارهای صوتی در آن قرار دارند. برآمدگی حنجره با لمس گلو کاملا مشخص و نمایان است. از دیدگاه زیست شناسی نقش اساسی حنجره فراهم کردن مکانیسمی است برای باز و بسته شدن مدخل ششها. به سخن دیگر این اندام به منزله شیر کنترل هوای ششهاست.

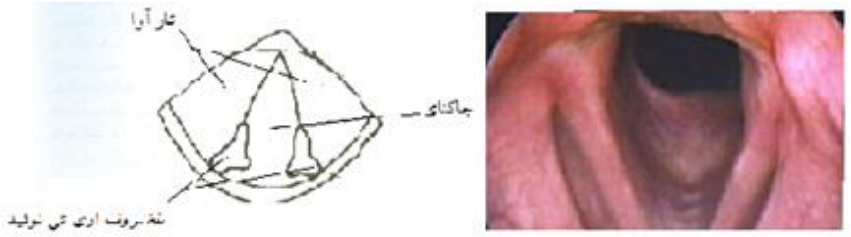


شکل ۱-۸ حنجره

تار آواها:

در داخل تیروئید و روی حلقه کری کوئید دو پرده ماهیچه ای بسیار نازک ولی پهن به شکل سرپوش قرار گرفته که تار آوا نام دارد.یک سر تار آواها به زاویه تیروئید و سر دیگر آنها هر یک به غضروف اری تی نوئید متصل می گردد. لبه خارجی تارهای صوتی به جدار کری کوئید چسبیده و لبه داخلی آنها که در مقابل یک دیگر قرار می گیرد آزاد است.

اری تی نوئید دو غضروف مثلث شکل هستند که به وسیله ماهیچه هایی که به آنها مربوط است می تواند حول قاعده خود حرکت دورانی داشته باشند. بدین ترتیب چرخش این دو غضروف و نزدیک شدن آنها به یکدیگر موجب می گردد که تار آواها (لبه داخلی آنها) به هم نزدیک و یا از هم دور شوند.



شکل ۱-۹ ساختمان داخلی حنجره

گلوگاه:

از حنجره به طرف بالا تا حفره های بینی گلوگاه نامیده می شود که محفظه ای است به شکل استوانه و می توان آن را به سه بخش تقسیم کرد:

۱- گلو: قسمتی که قاعده تحتانی آن حلقه کری کوئید و قاعده فوقانی آن غضروف نعلی شکل هیوئید است. جدار عقبی آن به دیواره ستون فقرات محدود است و در قسمت جلو آن دریچه نای یا اپی گلاتیس قرار دارد. طول تقریبی این قسمت حدود ۵ سانتیمتر است.

۲- حلق: از استخوان هیوئید تا زبان کوچک یا ملاز حلق نام دارد که طول این قسمت حدودا چهار سانتیمتر است. حلق در جلو به حفره دهان و یا به حفره بینی که در بالای حفره ی دهان قرار دارد ، باز می شود. دیواره جلویی حلق را ریشه زبان تشکیل می دهد.

۳- حلق فوقانی: از محل زبان کوچک به طرف بالا تا حفره بینی را حلق فوقانی می نامند. این بخش از گلوگاه حدود ۴ سانتیمتر طول دارد ولی با بالا رفتن و یا پایین آمدن ملاز، طول آن به طور قابل ملاحظه ای کم وزیاد می شود.

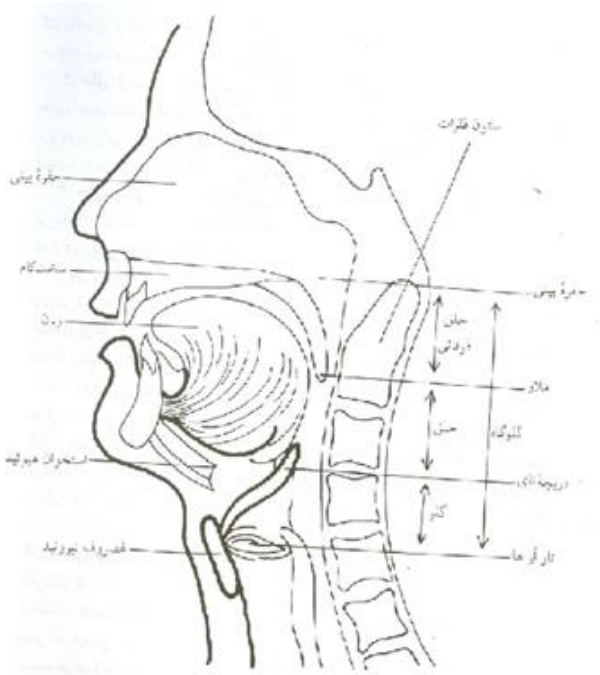
از نظر فیزیکی گلوگاه نقش بازخوان یا رزوناتور را برای صوت ایفا می کند. بدین ترتیب که ارتعاش تار آواها باعث ارتعاش هوای گلوگاه می گردد و این موضوع بر شدت صوت به طور چشم گیری افزوده و نیز بر کیفیت صوت تاثیر می گذارد. در واقع رابطه تار آواها نسبت به گلوگاه، همانند رابطه سیم های ویلون نسبت به بدنه ویلون است.

حفره بینی:

حفره های بینی در بالای گلوگاه قرار دارند که در قسمت جلو به سوراخ های بینی و در قسمت عقب به گلوگاه مربوط می شوند. نرم کام به منزله ی دریچه ی حفره های بینی است بدین معنی که با پایین آمدن آن گلوگاه به حفره های بینی متصل می گردد و هنگامی که نرمکام به بالا کشیده می شود راه عبور هوا از طریق حفره های بینی مسدود می گردد که این حفره ها نقش بازخوان را ایفا می کنند.

دهان:

حفره ی دهان از جلو به لب ها و از عقب به ملاز و از بالا به کام و از پایین به فک زیرین و در طرفین به جدار داخلی گونه ها محدود می شود. ابعاد این حفره به علت حرکات فک پایین، نرمکام، لبها، جدار گونه ها و زبان تا حد بسیار زیادی قابل تغییر است و همین حالت تغییر پذیری شکل و حجم دهان است که عامل تعیین کننده ی بسیاری از مشخصه های آوایی صداهای گفتار می باشد.



شکل ۱-۱۰ برش گلوگاه و دهان

کام:

کام یا سقف دهان از پشت دندانهای بالا (لثه ی بالا) تا زبان کوچک (ملاز) امتداد دارد. بخش پیشین آن به نام سخت کام، استخوانی و بدون حرکت است ولی بخش پسین آن نرمکام نامیده می شود که گوشتی و متحرک می باشد. هنگام تنفس معمولی نرمکام پایین می آید و بنابراین عبور هوا از طریق بینی صورت می گیرد. از لحاظ جایگاه تولید صداهای زبان، سقف دهان را می توان به چهار قسمت تقسیم کرد: لثه، سخت کام، نرمکام و ملاز (زبان کوچک که زاییده ایست نرم و گوشتی در انتهای نرمکام).

زبان:

زبان اندامی است نرم و گوشتی و دارای قابلیت انعطاف و تحرک فراوان. این اندام که مهمترین عضو گویایی به شمار می رود عامل اساسی، به طور مستقیم یا غیر مستقیم، در تولید نزدیک به تمام آواهای زبان است.

از دیدگاه محل تولید آواهای زبان و نیز در ارتباط با قسمت های مختلف کام، زبان را می توان به شش بخش تقسیم کرد: نوک زبان، تیغه ی زبان، جلوی زبان، مرکز زبان، عقب زبان و ریشه ی زبان.

دندانها:

دو ردیف دندانهای بالا و پایین از اندامهای گویایی محسوب می شوند. از آنجا که دندانها عامل سازنده ی برخی از صداهای زبان هستند، اختلال در نظم طبیعی آنها، به خصوص دندانهای پیشین (جای قرار گرفتن آنها، فاصله ی آنها از یک دیگر و تعداد آنها)، بر کیفیت صداهای تولید شده اثر نامطلوب می گذارد و حتی در پاره ای منجر به تولید صداهای به اصطلاح معیوب می گردد.

لبها:

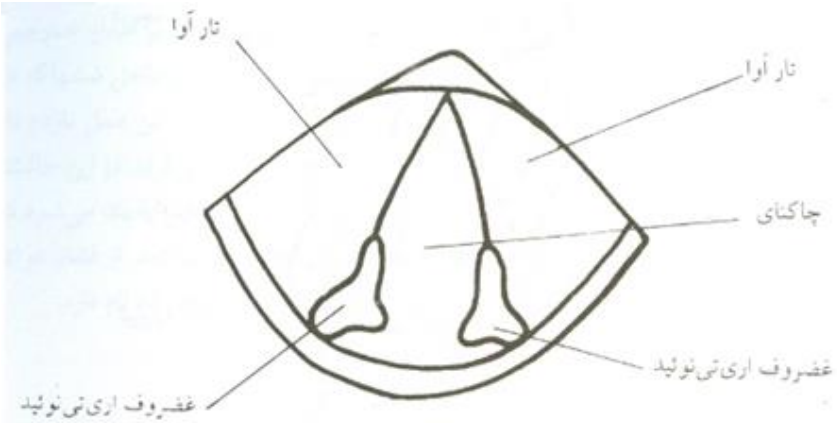
پس از زبان، لبها مهمترین اندام های گفتار به شمار می روند. ساختمان ماهیچه ای لبها آن را قادر می سازد که مانند زبان در تمام جهات حرکت نموده و نیز به شکلهای گوناگون، گرد، نیمه گرد، گسترده و غیره در آیند. لبها گذشته از آنکه خود عامل اصلی تولیدکننده ی برخی از آواها می باشند، به وسیله ی حرکات و نیز تغییر شکل خود، موجب تغییر حجم

حفره ی دهان شده و بالنتیجه سبب پیدایش کیفیتهای گوناگون در آواهای زبان، به ویژه واکه ها، می شوند.

علاوه بر اندامهای فوق، اندام های دیگری هستند که در تولید آواهای زبان نقش ثانویه بر عهده دارند. این اندامها عبارتند از: جدار داخلی گونه ها، غده های بزاقی، سینوسها و لثه ی پایین.

۱-۳- عملکرد (مکانیسم) سیستم گفتار:

عمل صحبت کردن و فرآیند ایجاد صوت نیاز به عبور جریان هوا دارد که این هوا توسط مخزنی به نام شش ها تامین می شود. پس ششها به مثابه ی یک مخزن هوا می باشند. برای خروج هوا از این مخازن (ششها) نیاز به ایجاد فشار در آنها می باشد که این فشار را دیافراگم و قفسه ی سینه تولید می کنند. با بالا آمدن دیافراگم و کاهش حجم قفسه ی سینه به ششها فشار وارد آمده و هوا از طریق مجاری تنفسی (نایچه، نای، گلو و...) به بیرون جریان پیدا می کند (عمل بازدم). در مسیر جریان هوای بازدم موانعی وجود دارند که کنترل این موانع ارادی است. در صورتی که شخص اراده به تولید صوت و گفتار نکند تنها عمل بازدم تنفسی صورت می گیرد، ولی اگر شخص این موانع را دخیل کند صوت تولید می شود.



شکل ۱-۱۱ طرز قرار گرفتن تار آواها در هنگام تنفس

اولین برخورد جریان هوای بازدمی با تارهای صوتی است که بسته به نحوه ی وضعیت قرار گیری تار آواها عبور هوا دستخوش تغییراتی واقع می شود. باز و بسته شدن سریع و متوالی تار آواها سبب بروز یک پدیده ی فیزیکی بسیار مهم به نام واک می گردد. هنگامی که تار آواها به هم نزدیک شده راه عبور هوا را به خارج مسدود می کنند، اگر این گرفتگی چنان نباشد که باز شدن آن مستلزم فشار زیاد باشد، طبیعی است که با اندک فشار هوای ششها که در زیر تارهای صوتی متمرکز گردیده تارهای مذکور کمی باز شده و نتیجتا مقداری هوا به بیرون می جهد. در این هنگام بر اثر کم شدن فشار هوای ششها، نیروی ماهیچه ای دوباره تار آواها را به هم کشیده و راه عبور هوا را می بندد. حال اگر این باز و بسته شدن ها به طور خیلی سریع و متوالی (بیش از ۱۶ بار در ثانیه) صورت پذیرد موج صوتی به وجود خواهد آمد. تولید بسیاری از آواهای زبان فارسی همراه با ارتعاش تار آواهاست.

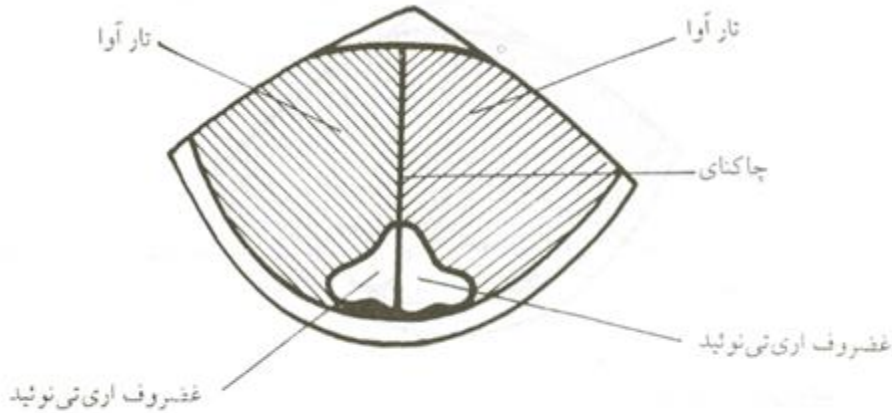
تار آواهای مرد ضخیم تر و طولانی تر از تار آواهای زن است و از این جهت بسامد صدای مرد کمتر از بسامد صدای زن است و یا به اصطلاح عوام صدای مرد کلفتتر از صدای زن است.

ساختمان ماهیچه ای تار آواها به نحوی است که می توان طول و ضخامت آنها را به طور ارادی تغییر داد. با چنین مکانیسمی طبیعتا کیفیت صوت از لحاظ زیر و بمی عوض می شود، به همین دلیل است که یک مرد قادر به نازک کردن صدای خود است. معمولا حد بسامد در صدای یک آدم متعارف بین ۷۰ تا ۱۰۰۰ سیکل در ثانیه است که البته بسامدهای پایین متعلق به صدای مرد و بسامدهای بالا مخصوص صدای زن است. از این پدیده ی فیزیکی یعنی زیر و بمی صدا، زبانها به نحوی وسیعی جهت ایفای نقش ارتباطی خود بهره می گیرند، مانند آهنگ صدا در زبان فارسی که یکی از عوامل مهم زبانی محسوب می شود.

درجه ی بازشدگی تار آواها و در نتیجه مقدار هوایی که به بیرون می جهد بستگی به مقدار فشاری دارد که بر تار آواها وارد می شود. این فشار پدید آورنده ی یک ویژگی دیگر صوتی است که آن را شدت صوت می نامیم. به سخن دیگر هر قدر فشار هوا هنگام عبور از چاکنای (فاصله ی میان دو تار آوا) بیشتر باشد دامنه ی نوسان موج بیشتر و نتیجتا صدا بلندتر است.

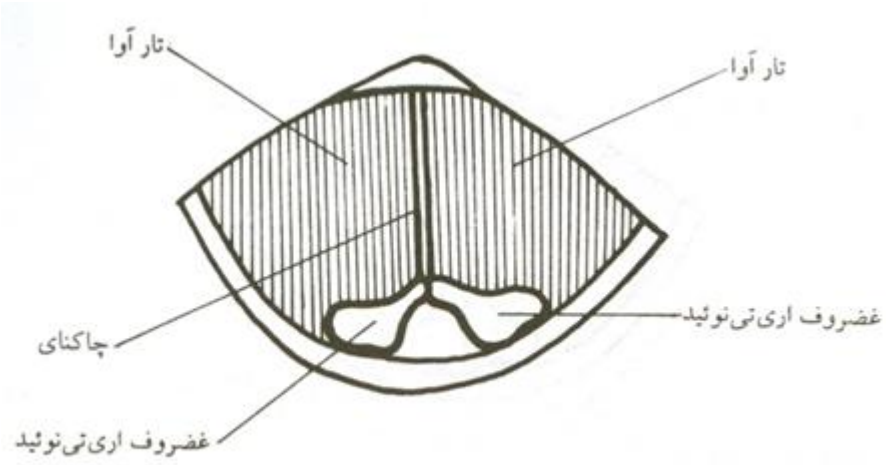
باز شدن تار آواها ممکن است به صورت انفجاری صورت پذیرد. بدین معنی که دو تار آوا در تمام طول خود به یکدیگر محکم چسبیده و راه عبور هوا را کامل می بندند. در این هنگام فشار شدید هوای ششها موجب می گردد که تار آواها به ناگاه از هم جدا شوند که در نتیجه

ی آن تمام هوای بند آمده با فشار زیاد به بیرون می جهد. در چنین حالتی است که صرفه یا عمل صاف کردن سینه به وقوع می پیوندد و نیز یکی از آواهای زبان فارسی که نشانه ی نوشتاری آن « ء یا ع » است بر اثر این مکانیسم به وجو می آید که البته شدت انفجار آن به مراتب کمتر از صرفه است.



شکل ۱-۱۲ طرز قرار گرفتن تار آواها هنگام تولید «ع» و «ء»

اگر لبه ی دو تار آوا طوری به هم نزدیک شوند که چاکنای به صورت یک شکاف باریک ترک مانند در آید، در این صورت فشار هوا هنگام عبور از این مجرای تنگ تولید سایش می نماید و این همان حالتی است که در هنگام نجوای آهسته (به اصطلاح صحبت در گوشی) وجود دارد. در موقع نجوای بلند، مقدار سایش بیشتر است زیرا اولا هوا با فشار بیشتری از چاکنای عبور می کند، ثانیا علاوه بر چاکنای، بین دو غضرف اری تی نوئید نیز فاصله ی کمی به وجود می آید که موجب رهایی مقدار بیشتری از هوا که توام با سایش است می گردد. همچنین تولید یکی از آواهای زبان فارسی که نشانه ی نوشتاری آن « ه یا ح » می باشد بر اثر این مکانسیم صورت می پذیرد.



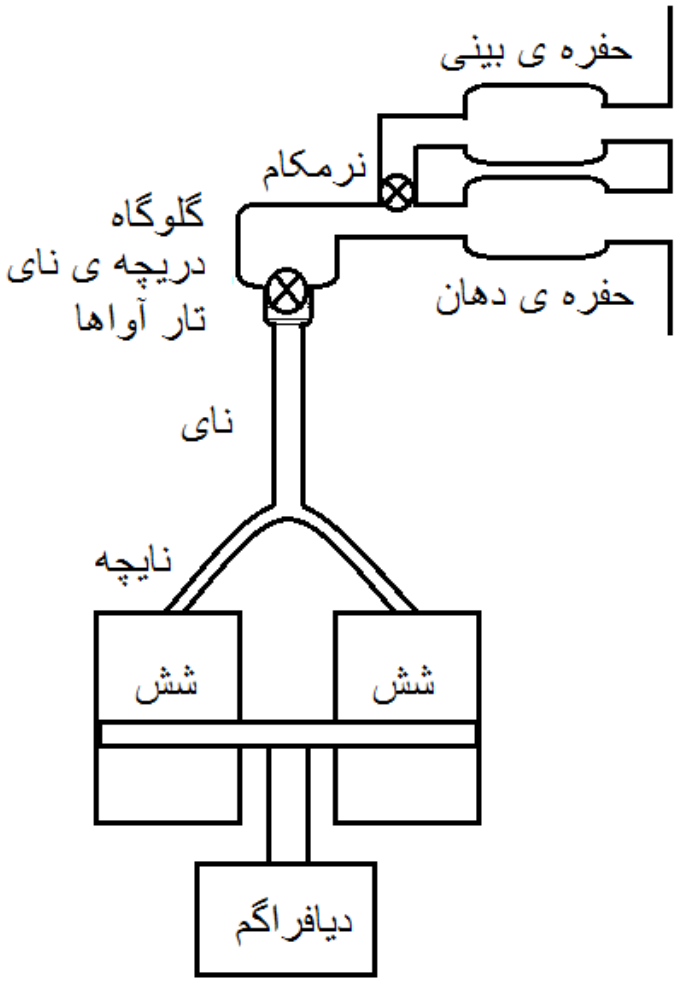
شکل ۱-۱۳ طرز قرار گرفتن تار آواها هنگام نجوا

بعد از نای جریان هوای بازدمی به گلوگاه می رسد. تار آواها به مثابه ی سیم های ویلون و گلو به مثابه ی بدنه ی ویلون است که گلو باعث تقویت صوت ایجادی تار آواها می گردد. از طرف دیگر ابعاد گلوگاه تحت تاثیر حرکات اندام های گفتاری همچون زبان، نرمکام، حنجره وغیره تغییر می کند و به تبع آن حجم هوایی که در حالات گوناگون به ارتعاش در می آید متفاوت خواهد شد و از طریق کیفیتهای گوناگون صوتی بروز می کنند.

پس از گلو صوت ایجادی (تا کنون) به دهان می رسد که با حرکت زبان بسیاری از آواهای فارسی تولید می شوند و همچنین کیفیت صوت بهبود می یابد. در نهایت از طریق دندانها و لبها گفتار تکمیل می شود.

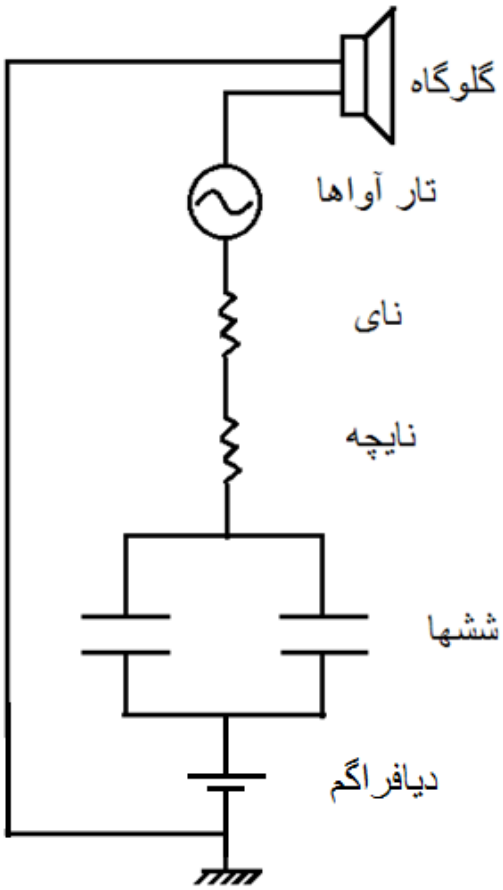
۱-۴- شبیه سازی سیستم گفتار:

همان طور که گفته شد، ششها به مثابه ی مخازن هوا (همچون بادکنک یا یک سیلندر و پیستون)، لوله ی نای و نایچه به مثابه ی شیلنگ، تارآوها به مثابه ی سیم ویلون، گلوگاه به مثابه ی بدنه ی ویلون، دریچه ی نای و نرم کام به مثابه ی یک شیر و حفره ی دهان و بینی به مثابه ی دو مجرا برای ارتباط هوا با دنیای بیرون می باشد. شکل زیر بیانگر این مساله می باشد:



شبیه سازی مکانیسم صوتی انسان

در شبیه سازی الکتریکی مکانیسم صوت، دیافراگم را با باتری، ششها را با خازن، نایچه و نای را با مقاومت، تار آواها را با نوسان ساز و گلوگاه را با بلندگو شبیه سازی می کنند. شکل زیر بیانگر این شبیه سازی است:



شبیه سازی الکتریکی مکانیسم صوتی انسان

در این شبیه سازی حجم معادل ولتاژ و فلو معادل جریان می باشد. حجم ششها را برابر فرض کردیم لذا دو شش به صورت دو خازن موازی هم فرض شده اند. از آنجا که فلوی هوا

به صورت سری از اجزای سیستم گفتار می گذرد لذا در شبیه سازی الکتریکی اجزای معادل به صورت سری به هم بسته شده اند. نیرویی که برای ایجاد فلو و کارکرد سیستم گفتار از طریق دیافراگم تولید می شود را با یک باتری و سری با خازن معادل می گیریم.

۱-۵- اختلالات گفتار:

اختلالات گفتار به شش دسته ی کلی تقسیم می شود که عبارتند از : ۱- اختلالات صوتی، ۲- اختلالات روانی، ۳- اختلالات تاخیری گفتار و زبان، ۴- اختلالات زبانی، ۵- اختلالات تولیدی، ۶- اختلالات تشدیدی.

۱- اختلالات صوتی:

زمانی اختلال صوتی وجود دارد که کیفیت، زیر و بمی، بلندی و انعطاف پذیری صوت فرد متفاوت با صوت افراد با جنس ، سن، گروه و فرهنگ مشابه ی وی باشد. وجود یا عدم وجود اختلال صوتی و میزان آن نسبت به قضاوت فرد شنونده متفاوت است. اختلالات صوتی را بر اساس سبب شناسی، ادراکی و یا بر اساس جنبش شناسی دسته بندی می کنند.

در طبقه بندی بر اساس سبب شناسی، اختلالات صوتی را به دو دسته ی عضوی: که شامل بد آوایی و بی آوایی به دلیل ضایعات بافتی یا بیماری های عصبی شناخت است و روانزاد: که شامل صوت ناهنجار در اثر سایکو نوروز، اختلالات شخصیتی یا الگوهای نادرست می باشد، تقسیم کرد.

در طقه بندی بر اساس ادراکی، اختلال در یک یا تعدادی از مولفه های صوت شامل زیر و بمی، بلندی و کیفیت وجود دارد.

در طبقه بندی بر اساس جنبش شناسی، اختلال صوتی به دو دسته ی پر کاری و کم کاری صوتی تقسیم می شود.

علایم اختلالات صوتی شامل موارد زیر است:

۱-گرفتگی صوت ۲- نفس آلودگی صدا ۳- خستگی صوتی ۴- کاهش دامنه ی آوا سازی ۵- بی صدا ۶- قطع زیر و بمی یا صدای زیر نا مناسب ۷- آواسازی توام با تلاش ۸- لرزش صدا.

۲- اختلالات روانی:

افرادی که لکنت زبان و پر گویی دارند جز این دسته هستند. لکنت زبان زمانی اتفاق می افتد که در روند طبیعی گفتار وقفه های غیر طبیعی و ناگهانی به واسطه ی تکرار صداها یا هجاها، کشیده گویی، میان پرانی، گیر، قفل و رفتارهای وابسته ایجاد شود.

پر گویی زمانی اتفاق می افتد که میزان تکلم دچار اشکال شود و گفتار فرد بیش از حد سریع باشد و فرد در گفتار از گفتن برخی از اصطلاحات صرفنظر کند.

۳- اختلالات تاخیری گفتار و زبان:

افرادی دارای این مشکل هستند که به دلایل مختلف از قبیل ناشنوایی، عقب ماندگی ذهنی و فلج مغزی دیرتر از حد طبیعی شروع به صحبت می کنند و گفتار و زبانشان در حد سنشان نمی باشد.

۴- اختلالات زبانی:

از جمله اختلالات زبانی ،آفازی است. آفازی در سه فرایند درک زبان، فرمول بندی زبان و یا هر دو اختلال به وجود می آورد.

آفازی در اثر آسیب به مناطق مختلف مغزی در نیمکره چپ به علل مختلف مانند تصادف، سکته ی مغزی، تروما و غیره به وجود می آید. آفازی بر اساس منطقه ای که آسیب دیده تقسیم بندی شده است و شامل انواعی است: ۱- آفازی ورنیکه ۲- آفازی بروکا ۳- آفازی انتقالی ۴- آفازی ترانس کورتیکال حسی ۵- آفازی گلوبال ۶- آفازی ترانس کورتیکال حرکتی

بر اساس نوع آفازی ممکن است که این علایم دیده شود: پار آفازیا، اختلال تکرار، اختلال در خواندن و نوشتن، اختلال در درک وبیان و اختلال در نامیدن.

۵- اختلالات تولیدی:

اختلالات تولیدی بیش از هر نوع عارضه گفتاری وقت و توجه ی آسیب شناسان گفتار را به خود مشغول می کند و تقریبا ۸۰ درصد موارد اختلالات گفتاری را تشکیل می دهد. این افراد در تولید و بیان همخوانها و واژه ها مشکل دارند و به صورت های زیر بیان می شوند:

۱-حذف: یعنی یکی از همخوانها را حذف می کنند. (مثلا خ در خط) ۲- خرابگویی: در این نوع خطا گوینده به جای واج هدف یک صدای غیر مشخص تولید می نماید. ۳- اضافی: در این حالت یک صدا یا هجای نابجا را وارد واژه یا هجای اصلی می نماید. (مثل قوه ری به جای قوری) ۴- جانشینی: در این گونه خطاها گوینده یک واج استاندارد به غلط جانشین واج هدف می نماید. (مثلا یوز به جای روز)

اختلالات تولیدی به علتهای زیر ممکن است به وجود آید:

۱- علل کارکردی: که بر اثر توانایی های حسی به وجود می آید.
۲- علل محیطی
۳- علل عضوی: مثل عقب ماندگی، ناشنوایی، اختلالات عصبی و عضلانی.

۶- اختلالات تشدیدی:

تشدید زیر مجموعه ی کیفیت صوت است. در اختلالات تشدید در اثر اختلال در عملکرد دریچه کامی-حلقی به علل مختلف تشدید به درستی انجام نمی شود. انواع اختلالات تشدید عبارتند از: ۱- خشیومی شدگی بیش از حد: تشدید بیش از حد هوا در

حفره ی بینی که ممکن است همراه با خروج هوا از بینی باشد یا نباشد؛ در این حالت ممکن است برای جلوگیری از خروج هوا در پرده های بینی حرکات انقباضی دیده شود. ۲- خیشومی کمتر از حد: تشدید صداهای خیشومی در حفره بینی یا صورت نمی گیرد یا بسیار کم است. ۳- مختلط: به علت اتصال حفره های بینی و انسداد در قسمت خلف یا قدام بینی هر دو حالت خیشوم شدگی وجود دارد.

فصل دوم

معرفی دستگاه کنترل صوتی بیماران مبتلا به بیماری حنجره

۲-۱- تعریف و نام گذاری دستگاه:

در این پروژه هدف طراحی و ساخت دستگاهی برای کنترل و درمان یکی از بیماریها و اختلالات مطرح شده در فصل اول است. همانطور که اشاره شد متخصصان گفتار درمانی اختلالات گفتار را به شش دسته ی کلی تقسیم کرده اند که در اینجا هدف کنترل بخشی از اختلالات صوتی می باشد. در واقع در این پروژه می خواهیم دستگاهی را طراحی کنیم که پر

گویی و صدای زیر نامناسب را تشخیص و به فرد بیمار اطلاع دهد تا فرد بیمار بتواند آن را کنترل نماید و پر حرفی و یا صدای زیر نامناسب خود را به اتمام برساند.

پر کاری صوتی یا به اصطلاح عامیانه پر حرفی مربوط به طبقه بندی بر اساس جنبش شناسی اختلالات صوتی و صدای زیر نامناسب مربوط به طبقه بندی بر اساس ادراکی اختلالات صوتی می باشد.

در دستگاهی که می خواهیم طراحی کنیم می توان بخش کنترل بلندی صدا که مربوط به طبقه بندی بر اساس ادراکی اختلالات صوتی می باشد را نیز به آن اضافه کرد و در صورتی که صدای فرد بیمار از یک حد بیشتر شد به بیمار اعلام کند تا فرد بلندی صدای خود را کم کند.

از آنجا که در این پروژه هدف کنترل بخشی از اختلالات صوتی بیماران گفتار است نام کنترل صوتی بیماران اختلال صوتی نامی شایسته تر از نام کنترل صوتی بیماران مبتلا به بیماری حنجره است ولی از آنجا که در ابتدای ثبت پروژه این نام اختیار شد نام دوم را برای این دستگاه اختیار می کنیم.

بنابراین دستگاه کنترل صوتی بیماران مبتلا به بیماری حنجره دستگاهی است که در آن بخشی از اختلالات صوتی بیمار همچون پر کاری صوتی، بلندی و زیر و بمی صوتی توسط دستگاه تشخیص داده شده و به بیمار ابلاغ می شود تا آن اختلالات را کنترل کند، پس در کل کنترل این اختلالات بر عهده ی بیمار است و این دستگاه فقط این اختلالات را تشخیص می دهد.

۲-۲- خصوصیات و ویژگی های دستگاه:

۱- تشخیص مدت صوت به طوری که اگر بیماربیشتر از یک حد آستانه به طور متوالی صحبت کرد دستگاه تشخیص داده و به بیمار هشدار دهد که بیشتر از آن به طور متوالی صحبت نکند. این حد آستانه را می توان به طور نرم افزاری (در برنامه ی میکرو) و یا در حالت پیشرفته تر کاربر از طریق صفحه کلید مشخص کرد. مثلا اگر آستانه را ۱۰ دقیقه تعیین کنیم، در صورتی که بیمار بیشتر از ۱۰ دقیقه پشت سر هم صحبت کند، دستگاه تشخیص داده و به بیمار هشدار دهد که صحبت خود را متوقف کند.

۲- ذخیره ی مدت زمان صحبت شده بیمار در یک دوره ی پزشکی (مثلا یک هفته) توسط دستگاه، دستگاه این قابلیت را دارد که برای یک دوره ی زمانی مقدار زمان صحبت شده را ذخیره کند و در پایان دوره پزشک از آن اطلاع یابد. در حالت ساده دستگاه باید دارای یک صفحه نمایش باشد که به پزشک مدت زمان صحبت شده و در حالت مطلوب تر روز و ساعت گفتگوها را در اختیار بگذارد و در حالت پیشرفته دارای قابلیت اتصال به رایانه را از طریق USBو یا کارت حافظه داشته و پزشک از طریق رایانه ی خود مطلع گردد.

۳- تشخیص فرکانس صوت به طوری که اگر فرکانس صوت بیمار بیشتر از یک حد آستانه شد دستگاه تشخیص دهد و آلارم بزند تا بیمار متوجه شود که غیر طبیعی صحبت می کند (بیمار با صدای زیر بیش از حد سخن می گوید). در حالت پیشرفته دستگاه روز و ساعت این حالات گفتار را ثبت کند و در پایان دوره در اختیار پزشک قرار دهد.

۴- دستگاه باید سبک و کوچک باشد تا بیمار راحت بتواند آن را در طی دوره ی پزشکی حمل کند.

۵- دستگاه باید قابلیت کار با باطری را داشته باشد به طوری که باطری بیش از ۴۸ ساعت شارژ بماند و به راحتی از طریق برق شهر شارژ گردد.

۶- دستگاه باید قابلیت اتصال به کمر بیمار را داشته باشد.

و در نهایت شخص بیمار را نرنجاند.

موارد ۱ و ۲ مربوط به پر کاری صوتی اختلالات صوتی و مورد ۳ مربوط به زیر و بمی نامناسب اختلالات صوتی می باشد.

۲-۳- فواید:

۱- بیمار به وسیله ی این دستگاه قادر خواهد بود که گفتار خود را کنترل نماید و به مرور زمان و تمرین گفتار خود را اصلاح کند به طوری که با گذشت زمان بدون دستگاه به راحتی بتواند گفتار خود را کنترل کند.

۲- این دستگاه اطلاعات پزشکی مناسبی را در اختیار پزشک و مراکز گفتار درمانی قرار می دهد به طوری که پزشک را در جهت معالجه بیمار کمک می کند.

۳- از این دستگاه می توان برای معالجه بیمار از طریق بیوفیدبک استفاده کرد.

۲-۴- معایب:

۱- با استفاده ی این دستگاه، افراد جامعه از بیماری فرد اطلاع می یابند و ممکن است که بیمار نخواهد دیگران از بیماری وی مطلع شوند.

۲- دستگاه بیمار را از نظر جسمی نیز می رنجاند، چرا که یک حلقه کشی باید بر گردن بیمار متصل شود، این حلقه ی کشی برای اتصال مناسب سنسور بر روی حنجره (جلوی گردن و بر روی سیب انسان) است.

۳- دستگاه صرفه ها و عطسه های پی در پی را اشتباها گفتار در نظر می گیرد و لذا مدت گفتار کاملا دقیق نخواهد بود که البته پزشک با در نظر گرفتن شرایط و بیماری بیمار از خطا می تواند بکاهد. بنابراین در کل می توان از این خطا چشم پوشی کرد.

فصل سوم

طراحی و ساخت دستگاه کنترل صوتی بیماران مبتلا به بیماری حنجره

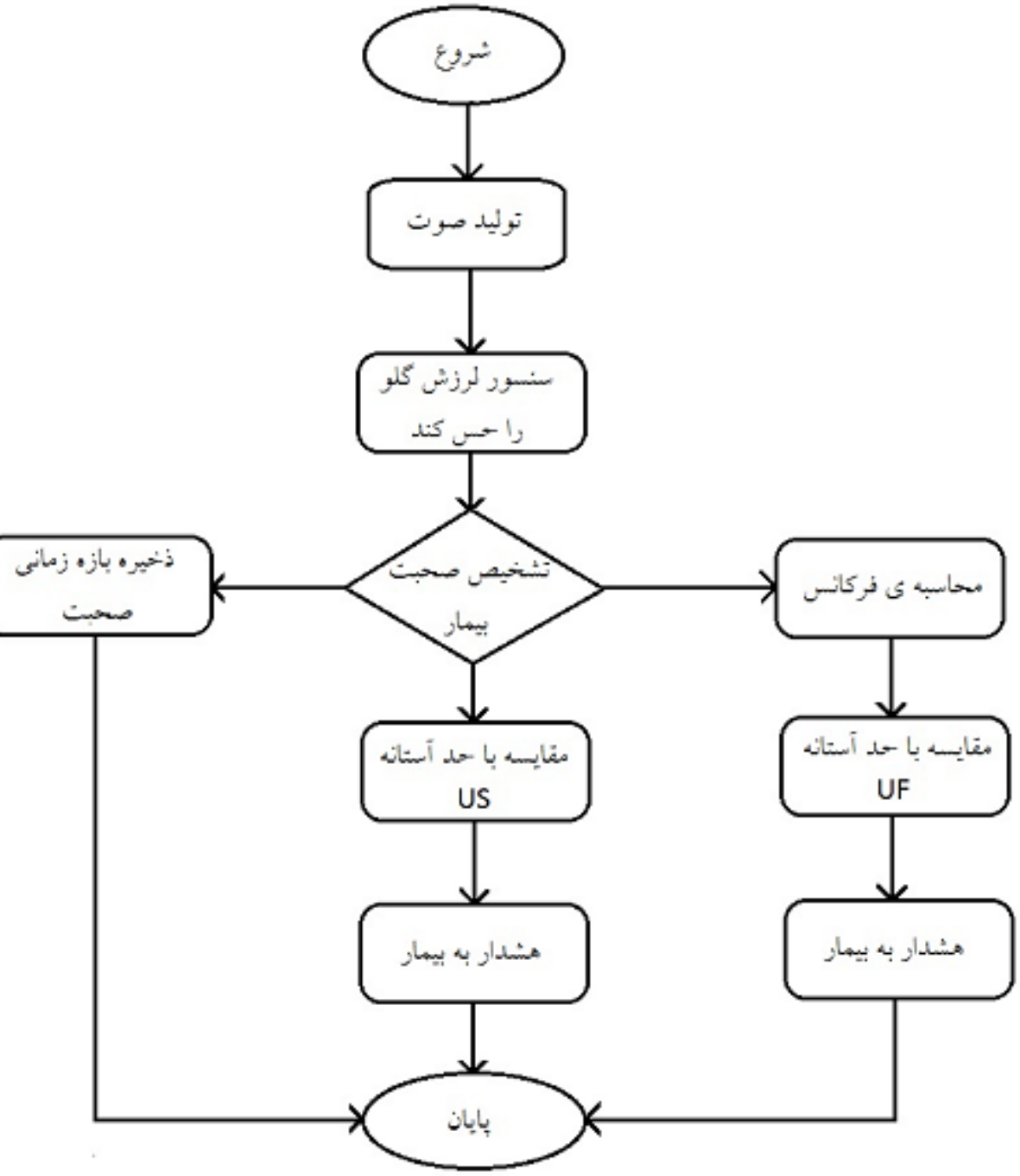
طراحی هر دستگاهی با توجه به نیاز آن می باشد، در حقیقت نیاز است که انسان را وادار به طراحی می کند و این پروژه نیز مستثنی نیست. کلینیکهای گفتار درمانی احساس نیاز به دستگاهی پیدا کردند که مدت زمان، شدت صوت و فرکانس صوت را در اختیار آنان قرار دهد تا با جمع آوری اطلاعات به دست آمده بتوانند در امر کمک رسانی به بیماران بیشتر

موثر واقع شوند. لذا هدف از طراحی و ساخت دستگاه کنترل صوتی برای بیماران مبتلا به بیماری حنجره امر کمک رسانی و اطلاع رسانی به کلینیکهای گفتار درمانی و بالخصوص بیماران گفتار درمانی است.

در طراحی ابتدا باید دید که هدف از طراحی چیست و خروجی دستگاهی که طراحی می شود چه می باشد، لذا با توجه به فصل قبل به طراحی الگوریتم دستگاه می پردازیم و بعد از شرح الگوریتم به بیان اجزا و بلوک دیاگرام و شرح آن اشاره می کنیم و در نهایت نمونه ی ساخته شده را بیان خواهیم کرد.

۳-۱- الگوریتم طراحی دستگاه:

برای پیاده سازی دستگاهی با ویژگی های فوق نیاز به الگوریتمی داریم که اهداف ما را تحقق بخشد، لذا الگوریتم زیر را برای طراحی این دستگاه در نظر گرفتیم:



۳-۲- شرح الگوریتم:

هنگامی که بیمار صحبت می کند، در اثر ارتعاش تار آواها ، در جدار خارجی حنجره (جلوی گردن) لرزش ایجاد می شود که این لرزش فرکانسی حدود بالاتر از ۷۰ هرتز (مردان) و پایین تر از ۲۵۰ هرتز (زنان) را داراست. بنابراین از طریق سنسوری از نوع شتاب سنج که محدوده ی کاری آن این بازه ی فرکانسی را شامل می شود می توان ارتعاش (لرزش) گلو را حس کرد و حتی آن را اندازه گیری کرد. برای تحقق ویژگی اول (بخش پرکاری صوتی) تنها حس کردن ارتعاش گلو کافی است، منتها برای تحقق ویژگی دوم (بخش زیر و بمی صوتی) باید مقدار فرکانس را نیز توسط سنسور اندازه گرفت تا با مقدار مرجع و تعیین شده از سوی پزشک (Upper Frequency) مقایسه گردد. در این پروژه از سنسور شتاب سنج ADXL203 استفاده شده است.

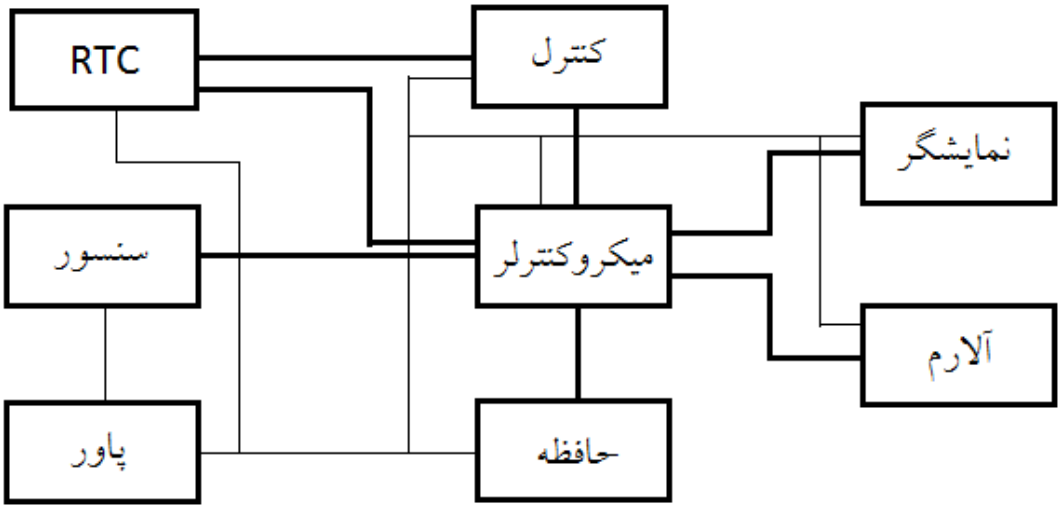
به محض آغاز صحبت بیمار، ارتعاش گلو باعث تغییر ولتاژ خروجی سنسور شده و لذا میکروکنترلر که پین ADC آن به خروجی سنسور متصل است این تغییر ولتاژ را حس می کند و تایمر شروع به شمارش خواهد کرد. در صورتی که مقدار شمارش شده توسط تایمر بیشتر از مدت زمان مرجع و تعیین شده توشط پزشک (Upper Speaking) باشد، دستگاه از طریق بازر آلارم خواهد داد. به محض توقف صحبت بیمار بازر خاموش خواهد شد. در ضمن مقدار شمارش شده ی تایمر در حافظه ذخیره خواهد گشت. برای بخش زیر و بمی صوتی، فرکانس توسط سنسور اندازه گیری شده و به محض آنکه مقدار فرکانس ارتعاش گلو به UF رسید، دستگاه از طریق بازر اخطار خواهد داد. البته باید توجه کرد که بوقی که از بازر پخش خواهد شد برای دو نوع آلارم متفاوت باشد تا کاربر از نوع آلارم با خبر شود که برای این کار می توان از دو نوع بازر متفاوت استفاده کرد و یا این که از طریق نرم افزاری یکی از بازرها را ممتد و دیگری را مقطع راه اندازی کرد. برای خاموش کردن آلارم در

این حالت می توان دستی (از طریق کلید) عمل کرد و یا به محض آنکه فرکانس از مقدار UF پایینتر آمد، بازر خاموش گردد.

۳-۳- اجزا و بلوک دیاگرام:

در این بخش ابتدا بلوک دیاگرام کلی دستگاه مطرح می شود و سپس به بیان کلی اجزا می پردازیم که در خلال شرح اجزای دستگاه قسمتهایی از اجزا را که در این پروژه استفاده خواهند شد نیز بیان می شوند.

بلوک دیاگرام کلی دستگاه:



بلوک دیاگرام دستگاه

ولتاژ خروجی آی سی حسگر (سنسور) بدون لرزش مقداری متفاوت با ولتاژ خروجی آی سی حسگر (سنسور) در حالت لرزش هست، لذا از طریق این اختلاف ولتاژ می توان لرزش و مقدار آن را محاسبه کرد. ولتاژ خروجی آی سی حسگر (سنسور) به پین ADC میکروکنترلر وصل است که میکرو از این طریق لرزش و مقدار آن را محاسبه می کند. قسمت مقایسه ی آنالوگ هم برای تشخیص مدت زمان صحبت شده از حد آستانه و هم برای تشخیص فرکانس از حد مطلوب در بلوک میکروکنترلر قرار دارد. از نمایشگر برای نمایش مدت زمان صحبت شده استفاده می شود. آلارم دارای دو مد ویبریشن و صدا دار (بازر) است که تنظیم آن توسط کاربر به کمک بخش کنترل صورت می گیرد. و همچنین کاربر از طریق بلوک کنترل می تواند آستانه ی مدت زمان و آستانه ی فرکانس مطلوب را تعیین کند. پاور انرژی الکتریکی لازم برای کلیه ی بخشهای دستگاه را تامین می کند.

۳-۴- شرح اجزای دستگاه:

در این بخش به بیان اجزای بلوک دیاگرام و بخشهای مهم این اجزا می پردازیم.

۳-۴-۱- آی سی حسگر:

ADXL103/ADXL203 یک IC مجتمع با دقت بالا، ولتاژ پایین و یک و دو محور شتاب است که از طریق سیگنال ولتاژ در خروجی شتاب را در یک محور و یا دو محور اندازه گیری می کند (ADXL103 شتاب را در یک محور و ADXL203 شتاب را در دو محور اندازه می گیرد).

رنج شتاب قابل اندازه گیری توسط این سنسور 1.7g- الی 1.7g+ است که می تواند هم شتاب دینامیک و هم شتاب استاتیک را اندازه بگیرد.

نویز زمین $110\mu g\sqrt{Hz}$ هست که برای کاربرد در پهناهای کم (زیر ۶۰ هرتز) به سیگنالهای زیر 1mg اجازه ی عبور داده می شود. خازنهای C_X و C_Y در پینهای X_{OUT} و Y_{OUT} به منظور انتخاب پهنای باند توسط کاربر در نظر گرفته شده اند. جدول زیر مقدار خازنهای C_X و C_Y را برای پهناهای باند مختلف برای فیلتر پایین گذر و برای فرکانس قطع 3db نشان می دهد:

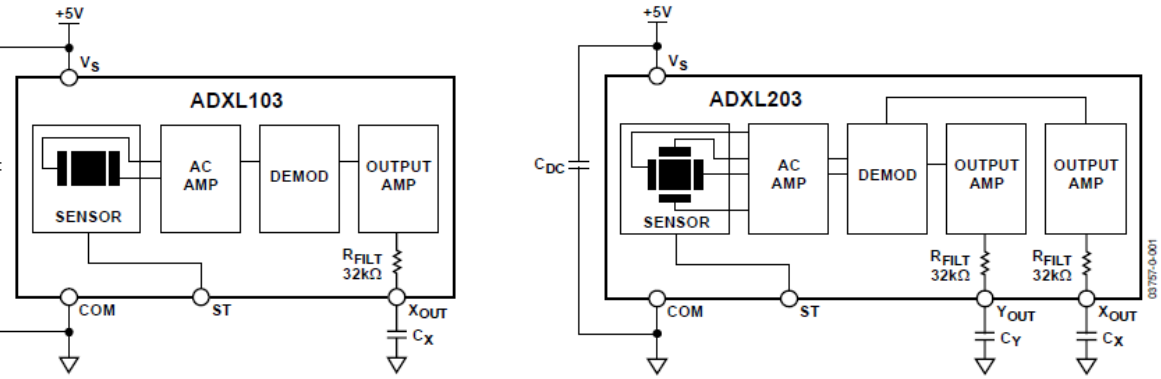
Filter Capacitor Selection, C_X and C_Y

Bandwidth (Hz)	Capacitor (µF)
1	4.7
10	0.47
50	0.10
100	0.05
200	0.027
500	0.01

انتخاب خازن C_Y و C_X برای پهناهای باند مشخص

در این پروژه پهنای باند را 500Hz می گیریم، لذا در خروجی آی سی خازنی با ظرفیت 10nf قرار می دهیم.

ساختار داخلی و بلوک دیاگرام ADXL103/ADXL203 به شکل زیر است:



بلوک دیاگرام ADXL103/ADXL203

هر دو آی سی حسگر ADXL103/ADXL203 شامل یک سنسور با سطح میکروماشین پلی سیلیکونی و مدار هدایت کننده ی سیگنال برای اندازه گیری شتاب هستند. سنسور دارای یک قرص سیلیکونی است که در اثر حرکت بر اثر اعمال نیروی شتاب، مقاومت آن تغییر می کند، انحراف صفحه سیلیکونی توسط یک خازن اختلاف اندازه گیری می شود که

این خازن شامل صفحه های ثابت و صفحه های متحرک (حساس به جرم) است. خروجی سنسور یک موج مربعی است که این موج مربعی در اثر انحراف در صفحه متحرک خازن ایجاد می شود. خروجی سنسور به یک تقویت کننده ی ac داده می شود. بعد از تقویت سیگنال مربعی سنسور، موج مربعی به یک دمودولاتور داده می شود که دمودولاتور متناسب با فرکانس موج مربعی یک ولتاژ در خروجی خود تولید می کند و در نهایت این خروجی به تقویت کننده جهت تقویت موج خروجی که یک سیگنال ولتاژ پیوسته است داده می شود. در پین خروجی کاربر می تواند با قرار دادن یک خازن پهنای باند را محدود کند.

پهنای باند کاری این آی سی از 0.5Hz تا 2.5KHz برای کاربردهای مختلف است که در این پروژه پهنای باند کاری 70Hz الی 350Hz می باشد، لذا این آی سی حسگر مناسب این پروژه است.

ADXL103/ADXL203 در ابعاد 5mm*5mm*2mm در دسترس هستند که به لحاظ اندازه نیز متناسب پروژه می باشد.

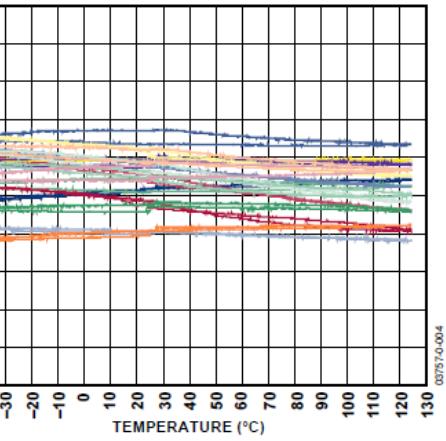
مشخصات آی سی حسگر ADXL103/ADXL203 در جدول زیر تشریح شده است.

SPECIFICATIONS

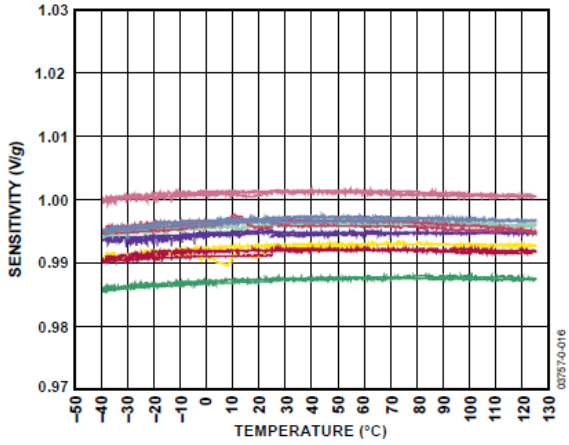
Table 1. $T_A = -40°C$ to $+125°C$, $V_S = 5$ V, $C_X = C_Y = 0.1$ µF, Acceleration = 0 *g*, unless otherwise noted.

Parameter	Conditions	Min	Typ	Max	Unit
SENSOR INPUT	Each Axis				
Measurement Range[1]		±1.7			*g*
Nonlinearity	% of Full Scale		±0.5	±2.5	%
Package Alignment Error			±1		Degrees
Alignment Error (ADXL203)	X Sensor to Y Sensor		±0.1		Degrees
Cross Axis Sensitivity			±2	±5	%
SENSITIVITY (Ratiometric)[2]	Each Axis				
Sensitivity at X_{OUT}, Y_{OUT}	$V_S = 5$ V	940	1000	1060	mV/*g*
Sensitivity Change due to Temperature[3]	$V_S = 5$ V		±0.3		%
ZERO *g* BIAS LEVEL (Ratiometric)	Each Axis				
0 *g* Voltage at X_{OUT}, Y_{OUT}	$V_S = 5$ V	2.4	2.5	2.6	V
Initial 0 *g* Output Deviation from Ideal	$V_S = 5$ V, 25°C		±25		*mg*
0 *g* Offset vs. Temperature			±0.1		mg/°C
NOISE PERFORMANCE					
Output Noise	< 4 kHz, $V_S = 5$ V, 25°C		1	6	mV rms
Noise Density	@25°C		110		µg/√Hz rms
FREQUENCY RESPONSE[4]					
C_X, C_Y Range[5]		0.002		10	µF
R_{FILT} Tolerance		24	32	40	kΩ
Sensor Resonant Frequency			5.5		kHz
SELF TEST[6]					
Logic Input Low				1	V
Logic Input High		4			V
ST Input Resistance to Ground		30	50		kΩ
Output Change at X_{OUT}, Y_{OUT}	Self Test 0 to 1	400	750	1100	mV
OUTPUT AMPLIFIER					
Output Swing Low	No Load		0.3		V
Output Swing High	No Load		4.5		V
POWER SUPPLY					
Operating Voltage Range		3		6	V
Quiescent Supply Current			0.7	1.1	mA
Turn-On Time[7]			20		ms

با توجه به منحنی های زیر مشخص است که این آی سی حسگر نسبت به دما تقریبا دارای رفتار خطی از لحاظ حساسیت و ولتاژ می باشد که این خود نشانه ی متناسب بودن برای پروژه است.

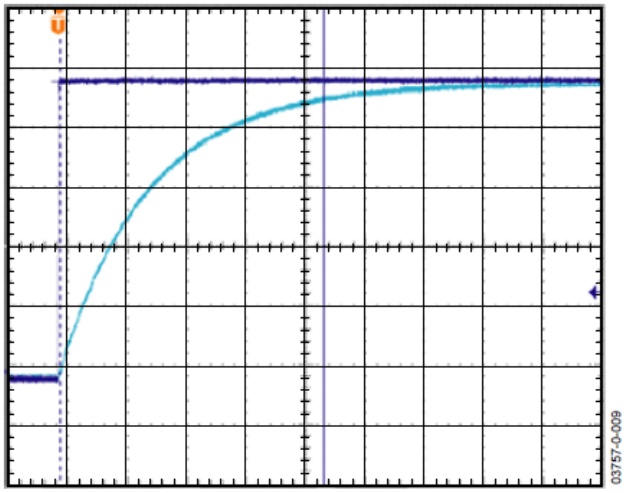


ero g Bias vs. Temperature – Parts Soldered to PCB



Sensitivity vs. Temperature – Parts Soldered to PCB

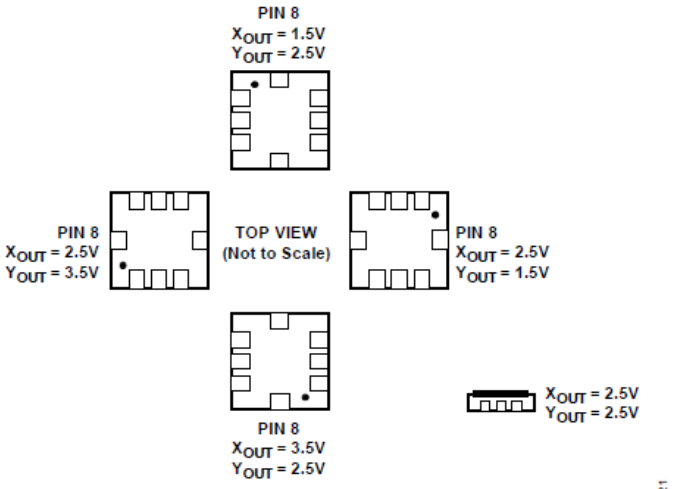
مدت زمان روشن شدن حسگر حدود ۲۰ میلی ثانیه است که از این مدت برای این پروژه به راحتی می توان صرفه نظر کرد.



Turn-On Time – C_X, C_Y = 0.1 µF, Time Scale = 2 ms/div

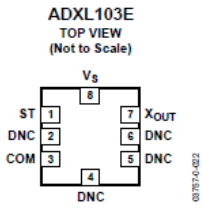
از آنجا که آی سی ADXL203 یک آی سی شتاب سنج است، لذا شتاب زمین بر این آی سی اعمال می شود. پس در حالات مختلف قرارگیری نسبت به زمین مقدار ولتاژ خروجی

متفاوت می باشد. به همین دلیل در این پروژه مقدار ولتاژ dc (ثابت) خروجی را صفر می کنیم و در بخش میکروکنترلر تنها تغییرات ولتاژ را لحاظ می کنیم.



Output Response vs. Orientation

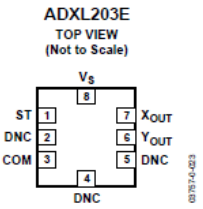
پین های این دو حسگر به شکل زیر است:



ADXL103 8-Lead CLCC

ADXL103 8-Lead CLCC Pin Function Descriptions

Pin No.	Mnemonic	Description
1	ST	Self Test
2	DNC	Do Not Connect
3	COM	Common
4	DNC	Do Not Connect
5	DNC	Do Not Connect
6	DNC	Do Not Connect
7	X_{OUT}	X Channel Output
8	V_S	3 V to 6 V

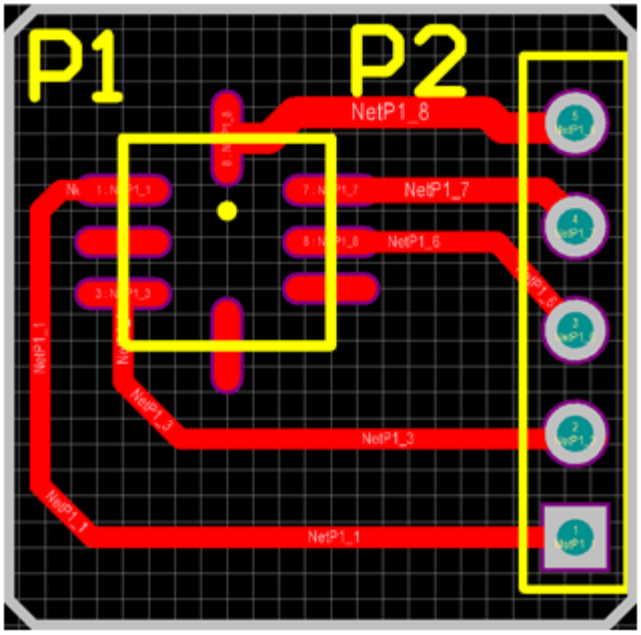


ADXL203 8-Lead CLCC

ADXL203 8-Lead CLCC Pin Function Descriptions

Pin No.	Mnemonic	Description
1	ST	Self Test
2	DNC	Do Not Connect
3	COM	Common
4	DNC	Do Not Connect
5	DNC	Do Not Connect
6	Y_{OUT}	Y Channel Output
7	X_{OUT}	X Channel Output
8	V_S	3 V to 6 V

در این پروژه استفاده از آی سی ADXL103 به دلیل اندازه گیری شتاب تنها در یک جهت بهینه تر از آی سی ADXL203 است، چرا که اندازه گیری شتاب ارتعاشات گلو در یک بعد کفایت می کند، اما به دلیل آنکه در بازار الکترونیک تهران این آی سی موجود نبود مجبور به انتخاب آی سی ADXL203 شدم که چون این آی سی MSD می باشد برای استفاده از آن مدار چاپی زیر را طراحی کردم:



طراحی مدار چاپی

شکل مدار چاپی حسگر

۳-۴-۲-میکروکنترلر:

با توجه به نیازی که برای تحقق اهداف در این پروژه مطرح گردید مصالحه بدین صورت گردید که میکروکنترلر AVR و از نوع Mega AVR (ATmega) مدل ATmega32 برای طراحی دستگاه انتخاب شود.

AVR ها، میکروکنترلرهای ۸ بیتی از نوع CMOS با توان مصرفی پایین هستند که بر اساس ساختار پیشرفته ی RISC ساخته شده اند.AVR ها با ساختار RISC، دستورات را تنها در یک پالس ساعت اجرا می نمایند و به این ترتیب می توان به ازای هر یک مگا هرتز، یک مگا دستور را در ثانیه (MIPS) اجرا کرده و برنامه را از لحاظ سرعت پردازش و نیز مصرف توان بهینه نمود.

AVR ها ۳۲ رجیستر همه منظوره و مجموعه دستورات قدرتمندی را شامل می گردند. تمامی این ۳۲ رجیستر مستقیما به ALU متصل شده اند. بنابراین دسترسی به دو رجیستر در یک سیکل ساعت هم امکان پذیر است. این ساختار موجب می گردد که سرعت آن نسبت به میکروکنترلرهای CISC تا ۱۰ برابر هم افزایش یابد.

خانواده ی میکروکنترلرهای AVR، تراشه های پیشرفته با امکانات جانبی کامل هستند. این میکروکنترلرها به سه دسته تقسیم می شوند:

۱- Tiny AVR (AT tiny)

۲- Classic AVR (AT90S)

۳- Mega AVR (ATmega)

تفاوت این سه دسته به امکانات موجود در آنها مربوط می شود. Tiny AVR ها غالبا تراشه هایی با تعداد پایه و مجموعه دستورات کمتری نسبت به Mega AVR ها می باشند و به عبارتی از لحاظ پیچیدگی حداقل امکانات را دارند.Mega AVR ها حداکثر امکانات را دارند و Classic AVR ها جایی بین این دو نوع قرار می گیرند.

اصول همه ی میکروکنترلرهای AVR یکی است و تفاوت آنها در میزان حافظه ی موجود بر روی تراشه و امکانات جانبی می باشد. علی رغم این که در اکثر AVR ها حداکثر فرکانس داخلی ۱۶ مگا هرتز است ولی در بعضی از انواع (مثلا ATtiny13) حداکثر فرکانس تایمر تا ۶۴ مگا هرتز افزایش یافته است. لذا با توجه به نیازهای این پروژه مناسب دیدم که میکروکنترلر ATmega32 متناسب با این پروژه است. خصوصیات این نوع میکروکنترلر به شرح زیر است:

- دارای ۱۳۰ دستور قدرتمند است که اکثر آنها در یک سیکل ساعت اجرا می شوند.
- 32K بایت حافظه ی Flash قابل برنامه ریزی داخلی.
- مجهز به قسمت Boot Loader.
- ۱۰۲۴ بایت حافظه ی EEPROM داخلی.
- 2K بایت حافظه ی SRAM داخلی.

امکانات جانبی:

- ارتباط JTAG، شامل اسکن کردن امکانات جانبی، حمایت از دیباگ کردن تراشه و برنامه ریزی حافظه های Flash و EEPROM، فیوز بیت ها و بیتهای قفل.

- دو تایمر/کانتر ۸ بیتی با تقسیم کننده ی فرکانسی مجزا و دارای مد Compare.
- یک تایمر/کانتر ۱۶ بیتی با تقسیم کننده ی فرکانسی مجزا و دارای مدهای Compare و Capture.
- دارای RTC (Real-Time Clock) با اسیلاتور مجزا.
- چهار کانال PWM، ۸ کانال ADC ۱۰ بیتی، ۸ کانال تک پایه، ۷ کانال تفاضلی در بسته بندی TQFP، ۲ کانال تفاضلی با بهره ی قابل تنظیم در 1x، 10x و یا 200x.
- ارتباط سریال دو سیمه (Two Wire).
- USART سریال قابل برنامه ریزی.
- Watchdog قابل برنامه ریزی با اسیلاتور مجزا.
- مقایسه کننده ی آنالوگ داخلی.
- اسیلاتور RC داخلی کالیبره شده.
- دارای شش مد Idle : Sleep، ADC Noise Reduction، Power-save، Power-down، Standby و Extended Standby.

ولتاژ عملیاتی:

- 4.5V تا 5.5V

فرکانس کاری

- 0MHz تا 16MHz

خطوط I/O و انواع بسته بندی

- ۳۲ خط ورودی /خروجی قابل برنامه ریزی
- ۴۰ پایه در نوع PDIP و ۴۴ پایه در انواع TQFP و MLF.

فیوز بیتهای ATmega32:

این میکروکنترلر دو بایت فیوز دارد. جدول های زیر چگونگی قرار گیری فیوزها در این دو بایت، عملکرد و وضعیت پیش فرض هر یک از فیوز بیتها را نشان می دهند.

Fuse High Byte

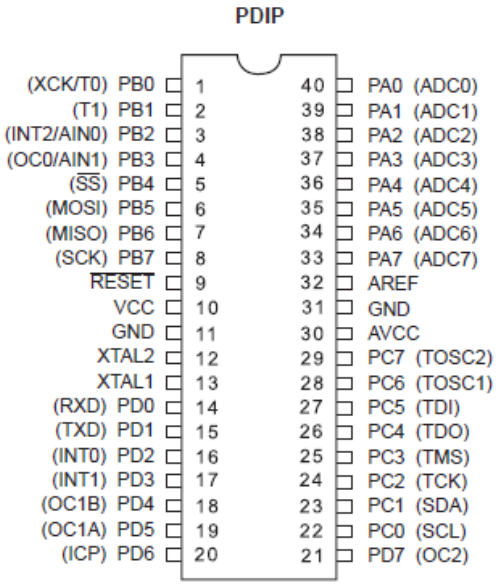
Fuse High Byte	Bit No.	Description	Default Value
OCDEN[4]	7	Enable OCD	1 (unprogrammed, OCD disabled)
JTAGEN	6	Enable JTAG	0 (programmed, JTAG enabled)
SPIEN[1]	5	Enable SPI Serial Program and Data Downloading	0 (programmed, SPI prog. enabled)
CKOPT[2]	4	Oscillator options	1 (unprogrammed)
EESAVE	3	EEPROM memory is preserved through the Chip Erase	1 (unprogrammed, EEPROM not preserved)
BOOTSZ1	2	Select Boot Size (see Table 100 for details)	0 (programmed)[3]
BOOTSZ0	1	Select Boot Size (see Table 100 for details)	0 (programmed)[3]
BOOTRST	0	Select reset vector	1 (unprogrammed)

بایت با ارزش فیوزبیتهای ATMega32

Fuse Low Byte

Fuse Low Byte	Bit No.	Description	Default Value
BODLEVEL	7	Brown-out Detector trigger level	1 (unprogrammed)
BODEN	6	Brown-out Detector enable	1 (unprogrammed, BOD disabled)
SUT1	5	Select start-up time	1 (unprogrammed)(1)
SUT0	4	Select start-up time	0 (programmed)(1)
CKSEL3	3	Select Clock source	0 (programmed)(2)
CKSEL2	2	Select Clock source	0 (programmed)(2)
CKSEL1	1	Select Clock source	0 (programmed)(2)
CKSEL0	0	Select Clock source	1 (unprogrammed)(2)

بایت کم ارزش فیوز بیتهای ATMega32



ترکیب بندی PDIP

۳-۴-۲-۱- واحد مبدل آنالوگ به دیجیتال (ADC) :

خروجی آی سی ADXL203 یک ولتاژ آنالوگ است که برای تجزیه و تحلیل آن توسط میکروکنترلر نیاز است که به دیجیتال تبدیل شود که این عمل توسط واحد ADC میکروکنترلر صورت می گیرد، لذا در اینجا این واحد و رجیسترهای آن را معرفی میکنیم و در نهایت مقدار رجیسترها را برای این پروژه به دست می آوریم.

ADC یک ولتاژ ورودی آنالوگ را از طریق تبدیلهای متوالی به یک مقدار دیجیتال ۱۰ بیتی تبدیل می کند که این کار در میکروکنترلرهای AVR از طریق تقریبهای متوالی صورت می گیرد.در میکروکنترلر ATMega32 ۸ کانال ورودی تک پایه (single Ended) مولتی پلکس شده جهت تبدیل ولتاژ آنالوگ به مقدار دیجیتال وجود دارد که در بسته بندی PDIP پین های ۳۳ تا ۴۰ میکرو را شامل می شود(شکل ؟)؛ همچنین پایه های دیگر مرتبط با واحد ADC پینهای AREF و AVCC می باشند. پین AREF ولتاژ مرجع ADC (V_{REF})می باشد که محدوده ی ولتاژ آنالوگ ADC را مشخص می کند. در کانالهای تک پایه، زمانی که ولتاژ از V_{REF} بالاتر باشد، نتیجه ی تبدیل به صورت 0x3FF خواهد بود. V_{REF} می تواند به صورت AVCC، ولتاژ مرجع 2.56v داخلی یا پایه ی خارجی AREF مشخص گردد. از آنجا که پایه ی خارجی AREF به صورت مستقم به ADC متصل است، می توان جهت تثبیت ولتاژ مرجع در برابر نویز از یک خازن 100nf بین پایه AREF و زمین استفاده کرد.

اگر از یک منبع ولتاژ ثابت متصل به پایه AREF استفاده کنیم، نمی توان از ولتاژهای مرجع دیگر استفاده کرد ولی اگر از اعمال ولتاژ خارجی به پایه ی AREF صرف نظر کنیم، می توانیم در داخل برنامه بین ولتاژهای مرجع AVCC و 2.56v یکی را انتخاب کرد که در

اینجا با توجه به محدوده ی ولتاژ آنالوگ داده شده به ADC، 5?v را به AVCC اعمال کردیم.

در واحد ADC میکروکنترلر ATMega32، ۷ کانل ورودی دیفرانسیلی و کانالهای دیگری وجود دارند که در این پروژه از آنها استفاده نشده است و لذا در این باره توضیحی داده نخواهد شد.

محدوده ولتاژ ورودی ADC از ۰ تا V_{CC} است که کمترین مقدار خروجی واحد ADC نشان دهنده ی مقدار موجود در پایه ی GND و حداکثر مقدار آن ولتاژ پایه AREF، منهای یک LSB می باشد.

خروجی آی سی ADXL203 به پین ۴۰ که پین ADC0 می باشد متصل می کنیم.

برای استفاده از واحد ADC میکرو باید رجیسترهای ADMUX (ADC Multiplexer Selection Register) و ADCSCR (ADC Control and status Register) مقدار دهی شوند که در ادامه به بررسی این دو رجیستر می پردازیم.

رجیستر ADMUX :

Bit	7	6	5	4	3	2	1	0	
	REFS1	REFS0	ADLAR	MUX4	MUX3	MUX2	MUX1	MUX0	ADMUX
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

بیتهای ۷:۶- REFS1:0(Reference Selection bit)

این بیتها مطابق جدول زیر یک ولتاژ مرجع را برای ADC انتخاب می کنند.

Voltage Reference Selections for ADC

REFS1	REFS0	Voltage Reference Selection
0	0	AREF, Internal Vref turned off
0	1	AVCC with external capacitor at AREF pin
1	0	Reserved
1	1	Internal 2.56V Voltage Reference with external capacitor at AREF pin

جدول انتخاب ولتاژ مرجع برای ADC

ولتاژ مرجع را AVCC انتخاب کردیم، بنابراین مقدار این دو بیت را ۰۱ قرار می دهیم که در این حالت پین AREF را با یک خازن 100nf به زمین اتصال می دهیم.

بیت ۵- ADLAR(ADC Left Adjust Result)

این بیت بر روی نحوه ی نمایش نتیجه ی تبدیل ADC در رجیستر داده ی ADC(ADCH & ADCL) تاثیر می گذارد. با یک کردن این بیت نتیجه از سمت چپ منظم می شود و در غیر این صورت از سمت راست منظم خواهد شد. به دلیل آنکه به دقت بیش از ۸ بیت نیاز نیست، مقدار این بیت را ۱ قرار می دهیم تا مقدار رجیستر در ۸ بیت ADCH قرار گیرد.

بیتهای ۴:۰- MUX4:0

مقادیر این بیتها ترکیب اتصال ورودی های آنالوگ متصب به ADC را مشخص می کند. در این پروژه خروجی حسگر به پین ADC0 متصل است، لذا مقدار این پنج بیت را ۰۰۰۰۰ قرار می دهیم.

Input Channel and Gain Selections

MUX4..0	Single Ended Input	Positive Differential Input	Negative Differential Input	Gain
00000	ADC0	N/A		
00001	ADC1			
00010	ADC2			
00011	ADC3			
00100	ADC4			
00101	ADC5			
00110	ADC6			
00111	ADC7			

انتخاب بهره و کانال ورودی

با توجه با توضیحات بالا، مقدار رجیستر ADMUX برابر 0x60 (بر مبنای هگز) خواهد بود.

رجیستر ADCSCR :

Bit	7	6	5	4	3	2	1	0	
	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0	ADCSRA
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

بیت۷- ADEN(ADC Enable)

با یک کردن این بیت ADC فعال می شود و با صفر کردن آن ADC خاموش می شود.

بیت ۶- ADSC(ADC Start Conversion)

در حالت Single Conversion، برای شروع هر تبدیل باید این بیت یک شود ولی در حالت Free Running تنها برای شروع اولین تبدیل این بیت باید یک گردد. تا زمانی که عمل تبدیل پایان نیافته باشد، بیت ADSC یک باقی خواهد ماند و با کامل شدن عمل تبدیل این بیت صفر می گردد. در این پروژه از مد Free Running استفاده می کنیم، لذا مقدار این بیت یک است.

بیت ۵- ADATE (ADC Auto Trigger Enable)

با نوشتن یک در این بیت، ADC در حالت تریگر خودکار فعال می شود و عمل تبدیل را در لبه ی مثبت سیگنال تریگر انتخاب شده آغاز می کند. منبع تریگر به کمک تنظیم بیتهای ADTS در رجیستر SFIOR انتخاب می شود.

بیت ۴- ADIF (ADC Interrupt Flag)

این بیت زمانی یک می شود که تبدیل آنالوگ به دیجیتال کامل شده و محتوای رجیستر داده ی ADC بازنویسی شده باشد. در این صورت اگر قبلا وقفه ی ADC فعال شده باشد، روتین وقفه اجرا می شود.

بیت ۳- ADIE (ADC Interrupt Enable)

زمانی که این بیت و بیت I واقع در رجیستر SREG یک گردند، وقفه ی مربوط به اتمام تبدیل ADC فعال می شود. به دلیل استفاده نکردن از وقفه در این پروژه، مقدار این بیت صفر است.

بیتهای ۲:۰- ADPS2:0 (ADC Prescaler Select)

این بیتها فرکانس کلاک ورودی به ADC که خود تقسیمی از فرکانس XTAL می باشد را مشخص می کنند. جدول زیر انتخاب تقسیم کننده ی فرکانسی به کمک این ۳ بیت را نشان می دهد:

ADC Prescaler Selections

ADPS2	ADPS1	ADPS0	Division Factor
0	0	0	2
0	0	1	2
0	1	0	4
0	1	1	8
1	0	0	16
1	0	1	32
1	1	0	64
1	1	1	128

انتخاب ضریب تقسیم فرکانس ADC

از آنجا که برای داشتن حداکثر تفکیک پذیری باید فرکانس پالس ساعت ADC بین 50KHz الی 200KHz باشد، تنها می توان دو مقدار ضریب تقسیم ۶۴ و ۱۲۸ را که به ترتیب پالسهایی با فرکانسهای 187.5KHz و 93.75KHz برای واحد ADC فراهم میکنند را انتخاب کرد. در این پروژه فرکانس 187.5KHz را انتخاب می کنیم و لذا مقدار این سه بیت ۱۱۰ (بر مبنای دودویی) خواهد بود.

با توجه به توضیحات بالا مقدار رجیستر ADCSCR را برابر 0xC6 قرار می دهیم.

اگر ولتاژ خروجی آی سی ADXL203 را V_{IN} (ولتاژ ورودی ADC) بگیریم، نتیجه ی تبدیل ADC از رابطه ی زیر به دست می آید:

$$ADC = \frac{V\,IN \times 1024}{V\,REF}$$

در این صورت 0x000 بیانگر زمین آنالوگ و 0x3FF بیانگر ولتاژ مرجع انتخاب شده منهای یک LSB خواهد بود.بدین صورت برای ولتاژهای خروجی آی سی ADXL203 مقدار متناظر ADC آن را بدست آورده و در برنامه ی میکرو استفاده می کنیم. مثلا به ازای ولتاژ خروجی 2.7v مقدار متناظر ADC، ۵۵۳ (بر مبنای ده تایی) است. در برنامه ی میکرو مقدار V_{IN}را از رابطه ی فوق بدست می آوریم.

$$V\,IN = \frac{ADC \times V\,REF}{1024}$$

و سپس شرطهای لازم جهت دستیابی به آنچه که در این پروژه می خواهیم را بر روی مقدار V_{IN}اعمال می کنیم.

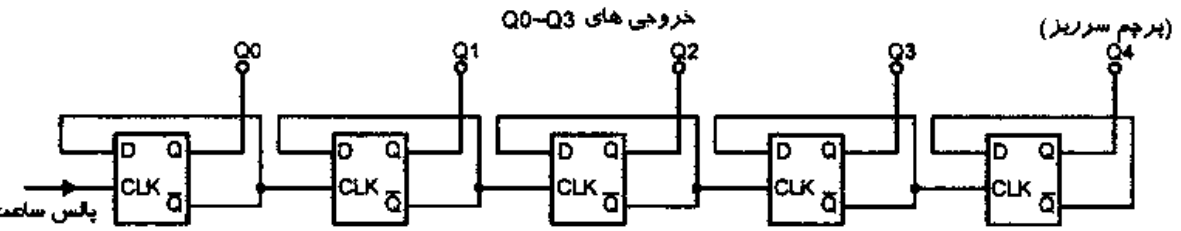
۳-۴-۲-۲- تایمر / کانتر :

تایمرها یکی از مهمترین امکانات میکروکنترلرها به حساب می آید که عملا عدم استفتده آز آنها در اکثر پروژه ها اجتناب ناپذیر است. در میکروکنترلرهای AVR، تعداد و قابلیتهای تایمر/کانترهای موجود در مدلهای مختلف با هم تفاوت دارد، به طوری که ساده ترین نوع میکروکنترلر، تنها یک تایمر/کانتر هشت بیتی دارد، در صورتی که پیشرفته ترین نوع، دو

تایمر/کانتر هشت بیتی و چهار تایمر/کانتر شانزده بیتی دارد. با این وجود اصول و قابلیتهای این تایمرها به میزان زیادی به یکدیگر شباهت دارد. در میکروکنترلر ATMega32 ، دو تایمر/کانتر ۸ بیتی (تایمر/کانتر صفر و دو) و یک تایمر/کانتر ۱۶ بیتی(تایمر/کانتر یک) وجود دارد.

در این بخش به معرفی اجمالی تایمر/کانتر و رجیسترها و کاربرد آن در این پروژه پسنده می کنیم.

تایمر و کانترها از تعدادی فلیپ فلاپ که به صورت پشت سر هم قرار گرفته و به صورت یک شمارنده ی آسنکرون عمل می کند تشکیل شده اند.



شمارنده آسنکرون

در این شمارنه، پایه های Q3-Q0 به عنوان خروجی استفاده می شود. با فرض وارد نمودن مقدار ۰۰۰۰ و با اعمال پالس ساعت، شمارنده شروع به شمارش می نماید و حداکثر تا مقدار ۱۱۱۱ بالا می رود. پس از این حالت و با اعمال پالس ساعت بعدی، بیت سرریز (Q4) فعال می شود که نشان دهنده ی این است که شمارنده به حداکثر مقدار خود رسیده است. هنگامی که پالس ساعت اعمالی به این شمارنده یک مقدار مشخصی داشته باشد، می توان با در نظر گرفتن مدت زمان شمارش شده از مقدار xxxx تا ۱۱۱۱ و سپس سرریز

شدن شمارنده، زمانهای معینی را ایجاد کرد که در این حالت شمارنده به صورت تایمر عمل می نماید. به عنوان نمونه برای شکل فوق در صورتی که پالس ساعت اعمالی دارای فرکانس ۱ هرتز باشد و مقدار اولیه ی شمارنده ۰۰۰۰ در نظر گرفته شود، حداکثر ۱۶ ثانیه طول می کشد تا شمارنده سر ریز شود.

هنگامی که پالس ساعت اعمالی به از منبع معین و منظمی نباشد (منبع خارجی) می توان از شمارنده برای شمارش وقایع خارجی، نظیر عبور تعداد قطعات از جلوی یک سنسور نوری استفاده نمود که در این حالت شمارنده به صورت کانتر به کار گرفته می شود.

در این پروژه تنها از تایمر استفاده می شود و کانتر نقشی در پروژه ندارد. در این پروژه از تایمر برای محاسبه زمان صحبت بیمار در طول یک دوره ی پزشکی و همچنین تشخیص مدت بیش از حد صحبت بیمار استفاده می شود. پس در پروژه دو بخش شرطی داریم: یک اینکه در صورت تشخیص صوت (برقراری شرط مقدار ADC(0) در دستور شرطی) تایمر روشن شود و دو اینکه اگر مقدار ذخیره شده تایمر در متغیر مربوطه بیش از مقدار پیش فرض کاربر شد، دستگاه آلارم دهد.

تایمر/کانترها در چهار مد زیر قابل استفاده هستند:

۱- مد نرمال
۲- مد CTC (Clear Timer On Compare Match)
۳- مد Fast PWM (Fast Pulse Width Modulation)
۴- مد Phase Correct PWM

در این بخش تنها دو مد نرمال و CTC را شرح می دهیم، چرا که مدهای دیگر در این پروژه مطرح نیستند.

مد نرمال: این مد ساده ترین مد عملکرد تایمر/کانتر است. در این حالت تایمر/کانتر همیشه به صورت افزایشی کار می کند و زمانی که محتوای رجیستر TCNTn به حداکثر مقدار خود (0xFF در تایمر/کانتر ۸ بیتی و 0xFFFF در تایمر/کانتر ۱۶ بیتی) برسد، پرچم TOVn (Timer Overflow) را یک می کند و در صورت فعال بودن وقفه ی سر ریز تایمر، روتین وقفه را نیز اجرا می نماید و محتوای رجیستر TCNTn را نیز به مقدار صفر تغییر می دهد.

مد CTC : در این مد زمانی که محتوای تایمر/کانتر (TCNTn)، با رجیستر OCRn برابر شود، TCNTn صفر می گردد. در حقیقت در این مد رجیستر OCRn مقدار حداکثر تایمر/کانتر و رزولوشن آن را تعیین می کند. در این حالت پس از هر بار رسیدن محتوای تایمر/کانتر به حداکثر مقدار خود که توسط رجیستر OCRn مشخص می شود، پرچم OCFn (Output Compare) یک می گردد و در صورتی که از قبل تطابق مقایسه (Compare Match) فعال شده باشد، روتین وقفه اجرا می گردد. اصولا از این وقفه برای تغییر محتوای OCRn استفاده می شود. در این مد می توان پایه ی OCn (Output Compare) را طوری تنظیم کرد تا در زمان تطابق مقایسه (Compare Match) تغییر وضعیت دهد.در این صورت یک موج مربعی با Duty Cycle 50% ، بر روی پایه ی OCn تولید خواهد شد که فرکانس آن با تغییر محتوای رجیستر OCRn قابل تغییر است.

در ادامه رجیسترهای تایمر/ کانتر صفر را به اجمال تشریح می کنیم. رجیسترهای تایمر/ کانتر دو شبیه رجیسترهای تایمر/ کانتر صفر است.

رجیستر TCCR0 (Timer / Counter Control Register)

Bit	7	6	5	4	3	2	1	0	
	FOC0	WGM00	COM01	COM00	WGM01	CS02	CS01	CS00	TCCR0
Read/Write	W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

بیت ۷ - FOC0 (Force Output Campare)

این بیت تنها زمانی فعال است که میکرو در یکی از مدها، غیر از مدهای PWM قرار داشته باشد. در این صورت زمانی که در این بیت یک منطقی نوشته شود، بلافاصله یک سیگنال تساوی مقایسه به واحد تولید موج فرستاده می شود که بسته به چگونگی تنظیم بیتهای COM0 1:0 پایه ی خروجی OC0 را تغییر می دهد.

بیتهای ۶ و ۳ -WGM0 1:0 (Waveform Generation Mode)

از این بیتها مطابق جدول زیر برای تعیین مد عملکرد تایمر استفاده می شود.

Waveform Generation Mode Bit Description

Mode	WGM01 (CTC0)	WGM00 (PWM0)	Timer/Counter Mode of Operation	TOP	Update of OCR0	TOV0 Flag Set-on
0	0	0	Normal	0xFF	Immediate	MAX
1	0	1	PWM, Phase Correct	0xFF	TOP	BOTTOM
2	1	0	CTC	OCR0	Immediate	MAX
3	1	1	Fast PWM	0xFF	TOP	MAX

نحوه ی ایجاد موجهای مختلف توسط تنظیم بیتهای WGM0

بیتهای ۵ و ۴ –COM0 1:0 (Compare Match Output Mode)

این دو بیت رفتار پایه ی خروجی OC0 را کنترل می کنند. در صورتی که یکی از بیتهای COM0 1:0 و یا هر دوی آنها یک شوند، پایه ی OC0، وظیفه ی خود به عنوان I/O را از دست می دهد و توسط تایمر کنترل می شود.

Compare Output Mode, non-PWM Mode

COM01	COM00	Description
0	0	Normal port operation, OC0 disconnected.
0	1	Toggle OC0 on compare match
1	0	Clear OC0 on compare match
1	1	Set OC0 on compare match

انواع عملکرد پایه ی خروجی OC0 در مدهای بدون PWM

بیتهای ۰ تا ۲ —CS02 (Clock Select)

این سه بیت با توجه به جدول زیر منبع کلاک تایمر / کانتر صفر را مشخص می کنند.

انتخاب کلاک تایمر صفر به کمک بیتهای CS0

CS02	CS01	CS00	Description
0	0	0	No clock source (Timer/Counter stopped).
0	0	1	$clk_{I/O}$/(No prescaling)
0	1	0	$clk_{I/O}$/8 (From prescaler)
0	1	1	$clk_{I/O}$/64 (From prescaler)
1	0	0	$clk_{I/O}$/256 (From prescaler)
1	0	1	$clk_{I/O}$/1024 (From prescaler)
1	1	0	External clock source on T0 pin. Clock on falling edge.
1	1	1	External clock source on T0 pin. Clock on rising edge.

رجیستر Register) TCNT0 (Timer / Counter

	7	6	5	4	3	2	1	0	
	TCNT0[7:0]								TCNT0
ead/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
tial Value	0	0	0	0	0	0	0	0	

رجیستر TCNT0

این رجیستر محتوای فعلی تایمر / کانتر صفر را در خود جای می دهد.

رجیستر OCR0 (Output Compare Register)

	7	6	5	4	3	2	1	0	
	OCR0[7:0]								OCR0
ad/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
ial Value	0	0	0	0	0	0	0	0	

رجیستر OCR0

این رجیستر ۸ بیتی پیوسته با رجیستر TCNT0 مقایسه می شود و در صورت تساوی، می تواند به کمک واحد تولید موج برای تولید خروجی مورد نظر در پایه ی OC0 استفاده گردد.

۳-۴-۲-۳- مدیریت توان و مدهای Sleep :

یکی از قابلیتهای میکروکنترلرهای AVR مدیریت توان و انواع مدهای Sleep در آن می باشد که به کاربر این امکان را می دهد تا در شرایط مختلف توان مصرفی AVR را بر حسب نیاز کنترل نماید. به کمک مدهای Sleep می توان قسمتهای مورد نظر در میکروکنترلر را متوقف نمود تا باعث کاهش توان مصرفی شود. در AVR حداکثر شش مد Sleep وجود دارد که

میکروکنترلر ATMega32 دارای هر شش مد می باشد که در ابتدا این شش مد را توضیح می دهیم و سپس مد مناسب این پروژه را انتخاب می کنیم.

انواع مدهای Sleep به شرح زیر می باشند:

مد Idle :

در مد Idle ، CPU متوقف شده ولی واحدهای SPI و USART ، مقایسه کننده ی آنالوگ، ADC، 2 wire ، تایمر و Watchdog در صورت فعال بودن به کار خود ادامه می دهند. اصولا در این مد، پالس ساعت FLASH و CPU متوقف شده و به بقیه پالس ساعتها اجازه ی فعالیت داده می شود. در مد Idle، CPU را می توان توسط وقفه های خارجی و داخلی مانند سرریز تایمر، وقفه ی مربوط به USARTو غیره بیدار نمود.

مد کاهش نویز ADC :

در این مد CPU متوقف می شود ولی به واحدهای ADC، وقفه های خرجی، 2 wire، تایمر ۲ و Watchdog اجازه ی فعالیت داده می شود. اساسا در این مد پالس ساعتهای CPU، I/O و FLASH متوقف شده و به بقیه ی پالس ساعتها اجازه ی فعالیت داده می شود. این مد برای کاهش نویز در زمان فعالیت ADC استفاده می شود و نمونه برداری را با دقت بیشتری انجام می دهد. هنگامیکه ADC فعال باشد، با وارد شدن به این مد عمل تبدیل به صورت اتوماتیک آغاز می شود. علاوه برا وقفه ی ADC، بازنشانی خارجی، بازنشانی Watchdog، بازنشانی Brown-Out، وقفه ی 2 wire، وقفه ی تایمر ۲، وقفه

ی آمادگی EEPROM و وقفه های خارجی می توانند CPU را از مد کاهش نویز ADC بیدار نمایند.

مد Power-Down :

در این مد نوسان ساز خارجی متوقف می شود در حالیکه وقفه های خارجی، 2wire و Watchdog به فعالیت خود ادامه می دهند. اساسا این مد تمام پالس ساعتهای تولید شده را متوقف کرده و فقط اجازه ی عملکرد به ماژولهای آسنکرون را می دهد. به منظور بیدار کردن میکروکنترلر از این مد می توان از بازنشانی خارجی Watchdog، بازنشانی Brown-Out، یک وقفه ی تطابق آدرس 2 wire یا یک وقفه ی خارجی استفاده نمود. در زمان بیدار شدن AVR از مد Power-Down ، CPU زمانی را به نام Startup برای پایدار شدن نوسانساز ایجاد می کند که این زمان توسط فیوز بیت CKSEL قابل تنظیم می باشد.

مد Power-Save :

عملکرد این مد شبیه عملکرد مد Power-Down بوده و تنها تفاوتی که دارد در این است که تایمر ۲ می تواند به صورت غیر همزمان کار کند (بیت AS2 در رجیستر ASSR یک شود).

مد Standby :

این مد فقط هنگام استفاده از کریستال یا نوسانساز خارجی در دسترس است. مد Standby تقریبا مشابه ی مد Power-Down است بجز اینکه AVR پس از گذشت شش سیکل ساعت از این مد بیدار می شود.

مد Extended Standby :

این مد نیز همچون مد Standby فقط در زمان استفاده از کریستال یا نوسانساز خارجی در دسترس است. این مد تقریبا مشابه ی مد Power-Save بوده و با این تفاوت که AVR پس از شش سیکل ساعت از این مد بیدار می شود.

رجیستر مد Sleep :

رجیستر کنترل MCU (MCUCR)

Bit	7	6	5	4	3	2	1	0	
	SE	SM2	SM1	SM0	ISC11	ISC10	ISC01	ISC00	MCUCR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

SE تواناساز Sleep :

جهت ورود AVR به مد Sleep باید این بیت یک شود.

0 ... SM2 بیتهای انتخاب مدهای Sleep :

این بیتها طبق جدول زیربه منظور استفاده از هر یک شش مد Sleep انتخاب می شود.

Sleep Mode Select

SM2	SM1	SM0	Sleep Mode
0	0	0	Idle
0	0	1	ADC Noise Reduction
0	1	0	Power-down
0	1	1	Power-save
1	0	0	Reserved
1	0	1	Reserved
1	1	0	Standby(1)
1	1	1	Extended Standby(1)

بیتهای SM0:2 جهت انتخاب یکی از شش مد Sleep

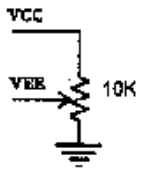
۳-۴-۳- نمایشگر:

نمایشگری که در این پروژه استفاده شده است یک LCD کاراکتری است.LCDهای کاراکتری عموما ۱۴ یا ۱۶ پایه دارند که وظایف این پایه ها به قرار جدول زیر است:

پایه‌های LCD های کارکتری و عملکرد آنها

شمارهٔ پایه	سمبل	I/O	عملکرد
۱	GND	-----	0V
۲	V_{CC}	-----	5V
۳	V_{EE}	-----	تنظیم شدت (Contrast)
۴	R_S	ورودی	انتخاب رجیستر
۵	R/W	ورودی	خواندن / نوشتن
۶	E	ورودی / خروجی	فعال سازی
۷	DB0	ورودی / خروجی	باس دادهٔ ۸ بیتی
۸	DB1	ورودی / خروجی	باس دادهٔ ۸ بیتی
۹	DB2	ورودی / خروجی	باس دادهٔ ۸ بیتی
۱۰	DB3	ورودی / خروجی	باس دادهٔ ۸ بیتی
۱۱	DB4	ورودی / خروجی	باس دادهٔ ۸ بیتی
۱۲	DB5	ورودی / خروجی	باس دادهٔ ۸ بیتی
۱۳	DB6	ورودی / خروجی	باس دادهٔ ۸ بیتی
۱۴	DB7	ورودی / خروجی	باس دادهٔ ۸ بیتی
۱۵	K	----	تنظیم نور زمینه کاتد
۱۶	A	----	تنظیم نور زمینه آند

پایه های ۱ و ۲ به ترتیب به زمین و ولتاژ 5v+ متصل می شوند. پایه ی ۳ میزان شدت کارکترها بر روی LCD را مشخص می کند که برای تنظیم دقیق آن می توان از یک پتانسیومتر به شکل زیر استفاده کرد.



مدار لازم برای تنظیم شدت LCD

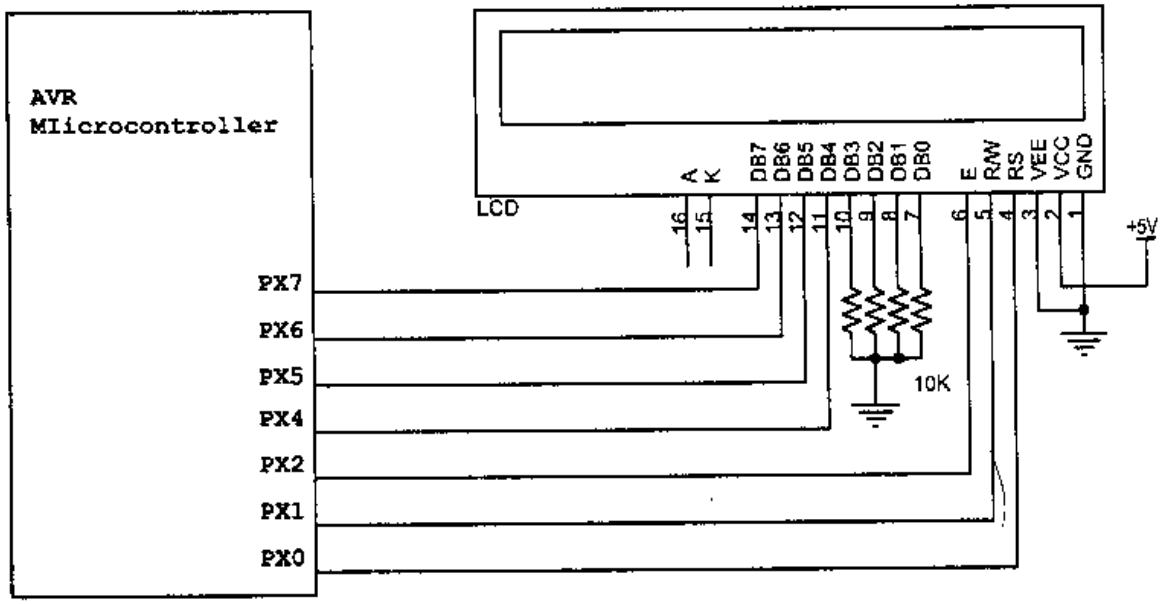
پایه های ۴ ، ۵ و ۶ نیز به کنترل LCD مربوط می شوند که باید توسط میکروکنترلر تنظیم گردند.از پایه های ۷ تا ۱۴ به عنوان یک باس ۸بیتی برای انتقال اطلاعات مابین LCD و میکروکنترلر استفاده می شود که برای تبادل اطلاعات بین میکروکنترلر و LCD، دو روش وجو دارد:

۱- ارتباط باس ۴ سیمه

۲- ارتباط باس ۸ سیمه

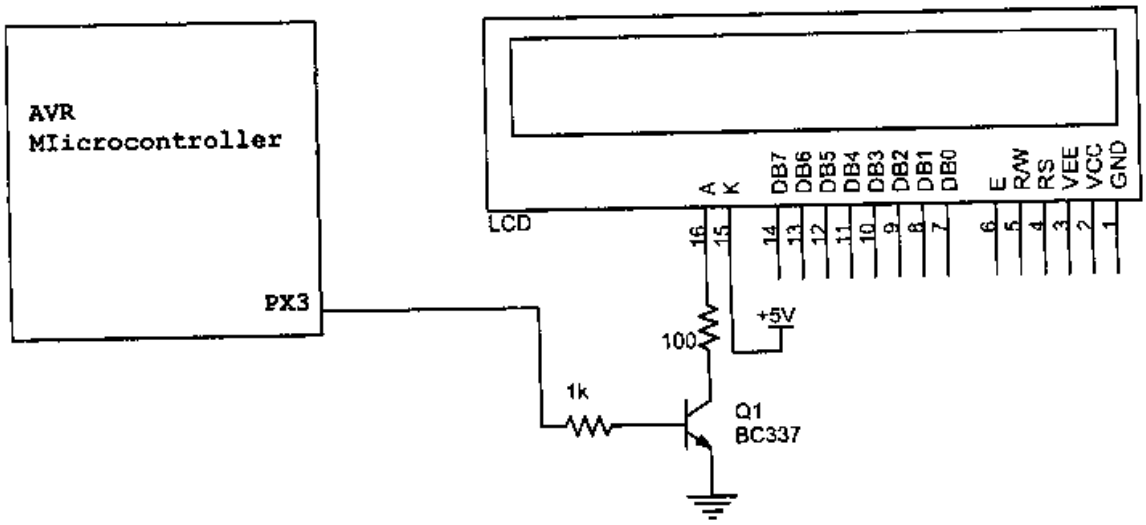
که معمولا از روش اول در ارتباط با میکروکنترلرهای AVR استفاده می شود. در این روش پایه های ۷، ۸، ۹ و ۱۰ بدون استفاده خواهند بود که می توان بدون اتصال آنها را رها کرد ولی از آنجا که ممکن است در محیطهای نویزی، این پایه ها تحت تاثیر قرار بگیرند، بهتر است تا این پایه ها را با مقاومت $10K\Omega$ به زمین متصل کرد.

شکل زیر نحوه ی اتصال LCD به میکروکنترلر را نشان می دهد:



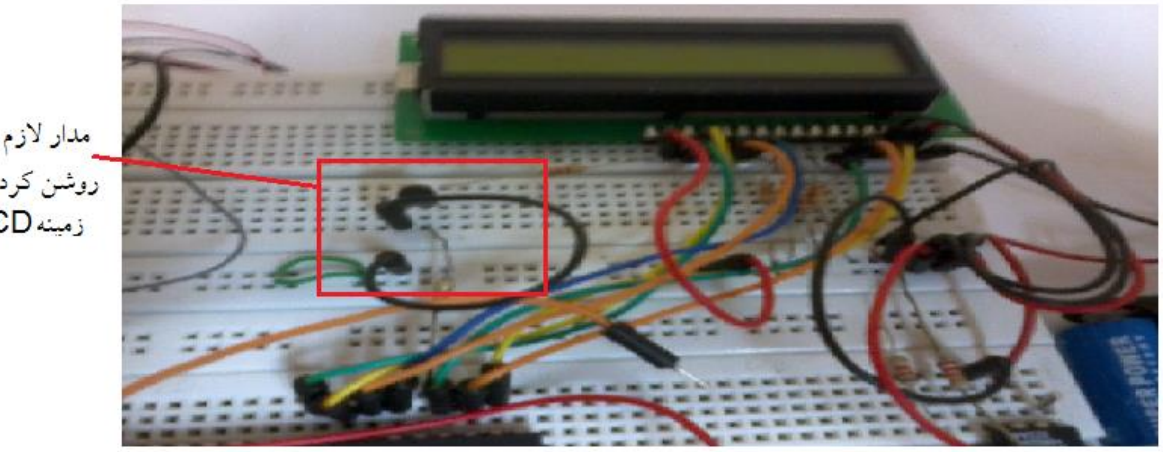
نحوهٔ اتصال LCD به میکروکنترلر

در صورتی که LCD، ۱۶ پایه باشند، پایه های ۱۵ و ۱۶ جهت روشن کردن نور زمینه ی LCD استفاده می گردند. شکل زیر مدار لازم برای روشن کردن نور زمینه ی LCD را نشان می دهد:



مدار لازم برای روشن کردن نور زمینهٔ LCD (Backlight)

نمایشگری که در این پروژه مورد استفاده واقع شد، LCD کاراکتری 16*2 بود.



نمایشگر LCD ۲*۱۶

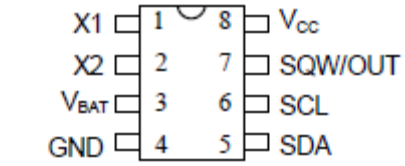
۳-۴-۴- بخش RTC (تاریخ و زمان دستگاه):

در طراحی لازم است که تاریخ و زمان دستگاه حتی در مواقع خاموشی حفظ بماند تا پزشک بتواند به کمک آن معالجات کلینیکی را انجام دهد، لذا به مداری نیاز است که به کمک آن بتوان زمان و تاریخ را به روزرسانی کرد. بدین منظور از مدار RTC میکرو همراه با آی سی مربوطه (DS1307) استفاده کردیم.

معرفی تراشه ی DS1307 :

DS1307 یک تراشه ی RTC سریال، با توان مصرفی کم و خروجی های BCD با ۵۶ بایت حافظه ی SRAM است. آدرس و داده به صورت سریال و از طریق باس I^2C منقل می شوند. این تراشه اطلاعات ثانیه، دقیقه، ساعت، روز هفته، روز، ماه و سال را در خود نگه می دارد. علاوه بر این تعداد روزهای ماه، برای ماههایی با کمتر از ۳۱ روز و همچنین سال کبیسه به صورت خودکار در این تراشه در نظر گرفته می شوند. تقویم به صورت میلادی و ساعت به صورت ۲۴ ساعته و یا ۱۲ ساعته با نمایانگر AM/PM تنظیم می گردد. این آی سی یک مدار داخلی برای تشخیص تغذیه دارد که در صورت قطع تغذیه ی تراشه، به صورت خودکار تغذیه ی خود را از باتری کمکی تامین می کند.

توضیح پایه های آی سی DS1307 :



DS1307 8-Pin DIP (300-mil)

: X1 , X2

این دو پایه باید به کریستال کوارتز 32.768 KHz متصل شوند که معمولا به این کریستال، کریستال ساعتی گفته می شود.

: V_{BAT}

این پایه ورودی تغذیه ی پشتیبان (کمکی) است که معمولا به یک باتری ۳ ولتی لیتیومی یا هر منبع انرژی دیگری با ولتاژی بین ۲٫۵ تا ۳٫۵ ولت متصل می شود. در صورت قطع تغذیه ی DS1307 یک باتری 4.8 mAhr و یا بیشتر، اطلاعات داخلی تراشه را حداقل برای ۱۰ سال حفظ خواهد کرد.

: GND

این پایه به زمین متصل می شود.

: SDA (Serial Data Input/Output)

این پایه، خط داده در باس I^2C است. پایه ی SDA، به صورت Open-Drain است و به یک مقاومت Pull-up خارجی نیاز دارد.

: SCL (Serial Clock Input)

این پایه، خط ورودی کلاک در باس I^2C است که برای سنکرون کردن انتقال داده ها در ارتباط سریال به کار می رود. از آنجایی که این پایه به صورت Open-Drain است به یک مقاومت Pull-up خارجی نیاز دارد.

: SQW/OUT (Square Wave / Output)

زمانی که در بیت SQWE یک نوشته شود، پایه ی SQW/OUT یک موج مربعی با فرکانس 1 Hz، 4KHZ، 8KHz یا 32KHz تولید می کند. این پایه به صورت Open-Drain است و به یک مقاومت Pull-up خارجی نیاز دارد. این پایه می تواند با V_{CC} یا V_{BAT} کار کند.

: V_{CC}

این پایه منبع تغذیه ی اولیه ی تراشه است که محدوده ی قابل قبول این تغذیه بین ۴٫۵ تا ۵٫۵ ولت است.

تقویم و ساعت:

اطلاعات تقویم و زمان، با خواندن رجیسترهای مربوطه در تراشه به دست می آیند و با نوشتن مقادیر مناسب در این رجیسترها، می توان آنها را تنظیم کرد. محتوای رجیسترهای

این تراشه به صورت BCD است. رجیستر مربوط به روز هفته در نیمه شب افزایش می یابد و مقادیر مربوطه به روزهای هفته توسط کاربر تعریف می شوند و باید پشت سر هم باشد.

رجیسترهای داخلی DS1307 :

	BIT7							BIT0	
00H	CH	10 SECONDS			SECONDS				00–59
	0	10 MINUTES			MINUTES				00–59
	0	12/24	10 HR A/P	10 HR	HOURS				01–12 00–23
	0	0	0	0	0	DAY			1–7
	0	0	10 DATE		DATE				01–28/29 01–30 01–31
	0	0	0	10 MONTH	MONTH				01–12
	10 YEAR				YEAR				00–99
07H	OUT	0	0	SQWE	0	0	RS1	RS0	

رجیسترهای داخلی تراشه ی DS1307

رجیستر کنترل DS1307:

BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
OUT	0	0	SQWE	0	0	RS1	RS0

این رجیستر برای کنترل نحوه ی عملکرد پایه ی SQW/OUT استفاده می شود.

بیت ۷- OUT (Output Control)

در زمانی که موج مربعی خروجی پایه ی SQW/OUT غیرفعال باشد، این بیت سطح خروجی این پایه را کنترل می کند. اگر این بیت یک باشد سطح پایه ی SQW/OUT برابر یک و در غیر این صورت صفر خواهد بود.

بیت ۴- SQWE (Square – Ware Enable)

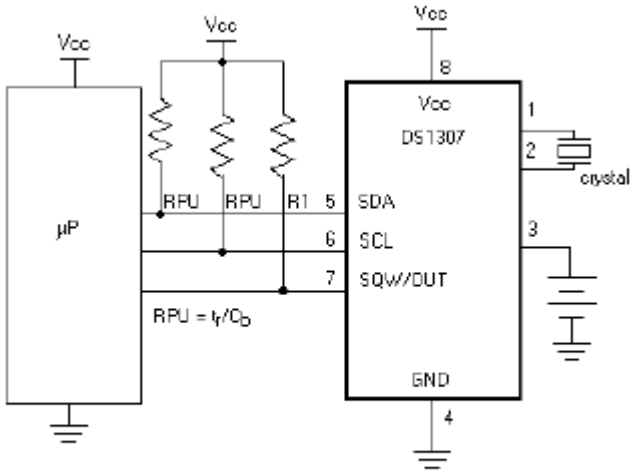
زمانی که این بیت یک شود، خروجی موج مربعی در پایه ی SQW/OUT فعال می شود که در این صورت فرکانس تولید موج مربعی توسط بیتهای RS1 و RS0 تعیین می گردد.

بیتهای ۰ و ۱ –RS1:0 (Rate Select)

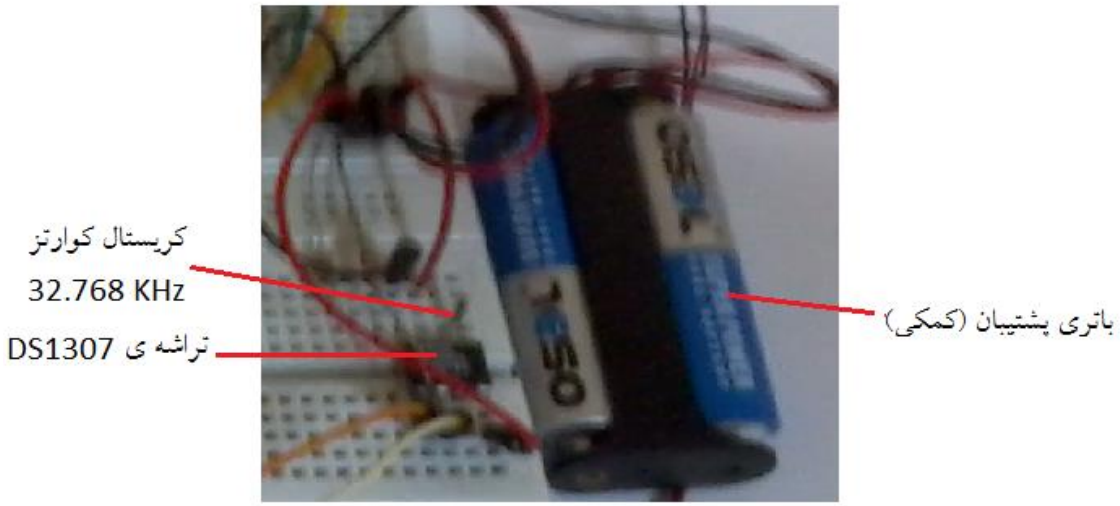
در صورتی که موج مربعی خروجی در پایه ی SQW/OUT فعال شده باشد، این دو بیت به صورت جدول زیر فرکانس موج مربعی را مشخص می کند.

RS1	RS0	SQW OUTPUT FREQUENCY
0	0	1Hz
0	1	4.096kHz
1	0	8.192kHz
1	1	32.768kHz

مدار اتصال تراشه ی DS1307 به میکروکنترلر:



مدار اتصال تراشه DS1307 به میکروکنترلر



مدار مربوط به اتصال DS1307 به میکروکنترلر

برنامه ی مربوط به راه اندازی آی سی DS1307 در قسمت کد دستگاه ذکر شده است.

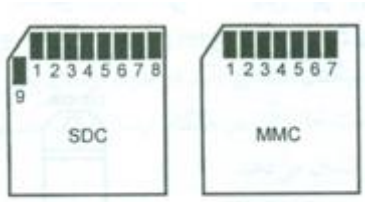
۳-۴-۵- بخش حافظه دستگاه:

حافظه ی EEPROM میکروکنترلر برای ذخیره ی کلی اطلاعات کافی است ولی در اتمام دوره ی پزشکی لازم است که اطلاعات به پزشک منتقل شود که بدین منظور می توان از ارسال سریال به رایانه و یا USB و یا کارتهای حافظه ی MMC یا SDC استفاده کرد که در این پروژه از کارت حافظه ی SDC استفاده کرده ایم. در ادامه کارتهای حافظه را معرفی خواهیم کرد.

MMC(Multi Media Card) نوعی کارت حافظه از نوع FLASH بوده و معمولا دارای ظرفیتهای بالا تا چندین گیگابایت می باشد که از کاربردهای آن می توان در موبایل، MP3 Playerها، دوربینهای دیجیتال و PAD,s ها نام برد.

SDC (Secure Digital Memory Card) نوعی حافظه است که معمولا برای دستگاههای موبایل به کار می رود. کارت حافظه ی SDC متناسب با کارت حافظه MMC طراحی شده است.

MMC/SDC دارای یک میکروکنترلر داخلی جهت عملیاتهای پاک کردن، خواندن و نوشتن کنترل خطا در حافظه ی FLASH می باشند.



کارتهای حافظه

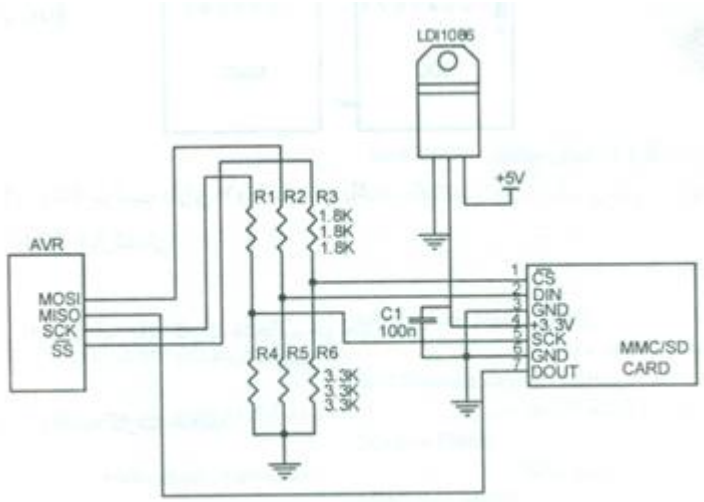
کارتهای حافظه MMC و SDC به ترتیب دارای ۷ و ۹ پایه می باشند. پایه های کارت MMC عبارتند از:

۱. پایه ی CS : با فعال کردن (Low) این پایه کارت انتخاب خواهد شد.
۲. پایه ی SDI : ورودی سریال داده به کارت حافظه.
۳. پایه ی GND : اتصال به زمین.
۴. پایه ی VS : برای تغذیه کارت حافظه باید ولتاژ مابین ۲/۷ تا ۳/۶ ولت به این پایه اعمال نمود.
۵. پایه ی SCLK : پالس ساعت ورودی به کارت حافظه.
۶. پایه ی GND : اتصال به زمین.
۷. پایه ی SDO : خروجی سریال داده از کارت حافظه.
۸. پایه ی DAT1
۹. پایه ی DAT2

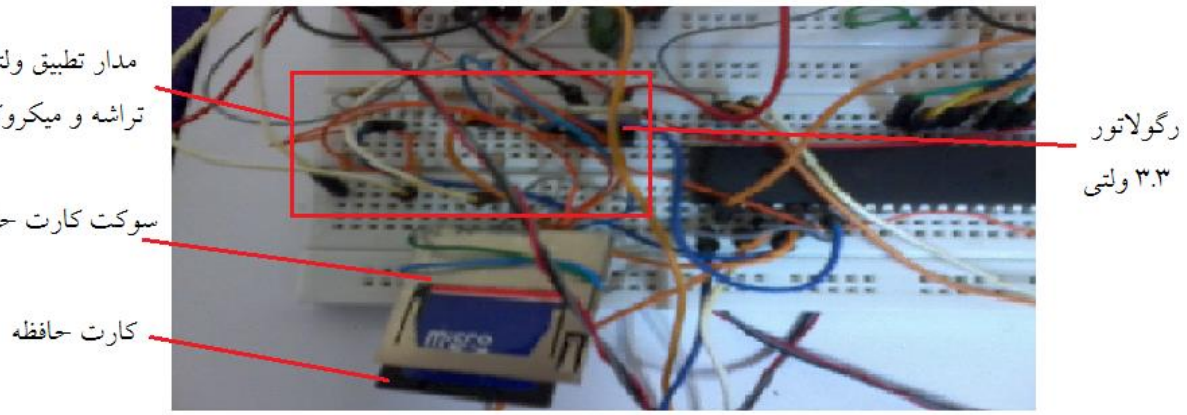
پایه های کارت SDC مشابه با کارت MMC طراحی شده و فقط دو پایه اضافه تر نسبت به MMC دارد که در این پروژه از این دو پایه استفاده نمی کنیم.

اتصال کارت حافظه به میکروکنترلر:

کارتهای حافظه برای فعالیت به یک ولتاژ تغذیه بین ۲٫۷ تا ۳٫۶ ولت با حداکثر جریان دهی ۱۰۰ میلی آمپر نیاز دارند، لذا بدین منظور از رگولاتور ولتاژ ۳٫۳ ولتی مانند LF33 استفاده کردیم. شکل اتصال کارت حافظه به میکروکنترلر مطابق زیر است:



اتصال کارت حافظه به میکروکنترلر



مدار اتصال کارت حافظه به میکروکنترلر

ولتاژ تغذیه میکروکنترلر ۵ ولت است در حالی که ولتاژ تغذیه کارت حافظه ۳٫۳ ولت

است، لذا برای تطبیق بین این دو ولتاژ از یک شبکه ی تقسیم کننده ی مقاومتی استفاده شده است.

برنامه کاربردی برای ارتباط میکروکنترلر با کارت حافظه MMC/SDC به صورت زیر است که از کتاب AVR پرتویی فر اسکن شده است:

```
#include  <mega32.h>
#include  <delay.h>

#define  MMC_CS_PORT       PORTB
#define  MMC_CS_DDR        DDRB
#define  MMC_CS_PIN        4

// MMC commands (taken from sandisk MMC reference)
#define  MMC_GO_IDLE_STATE          0     // initialize card to SPI-type access
#define  MMC_SEND_OP_COND           1     // set card operational mode
#define  MMC_SEND_CSD               9     // get card's CSD
#define  MMC_SEND_CID               10    // get card's CID
#define  MMC_SEND_STATUS            13
#define  MMC_SET_BLOCKLEN           16    /*Set number of bytes to
                                            transferperblock*/
#define  MMC_READ_SINGLE_BLOCK      17    // read a block
#define  MMC_WRITE_BLOCK            24    //write a block
#define  MMC_PROGRAM_CSD            27
#define  MMC_SET_WRITE_PROT         28
#define  MMC_CLR_WRITE_PROT         29
```

```
#define MMC_SEND_WRITE_PROT      30
#define MMC_TAG_SECTOR_START     32
#define MMC_TAG_SECTOR_END   33
#define MMC_UNTAG_SECTOR     34
#define MMC_TAG_ERASE_GROUP_START 35     // Sets beginning of erase group
                                         (mass erase)
#define MMC_TAG_ERARE_GROUP_END36 // Sets end of erase group (mass erase)
#define MMC_UNTAG_ERASE_GROUP  37 // Untag (unset) erase group (mass erase)
#define MMC_ERASE    38                  // Perform block/mass erase
#define MMC_CRC_ON_OFF     59            // Turns CRC check on/off
// R1 Response bit-defines
#define MMC_R1_BUSY     0x80             // R1 response: bit indicates card is busy
#define MMC_R1_PARAMETER        0x40
#define MMC_R1_ADDRESS          0x20
#define MMC_R1_ERASE_SEQ        0x10
#define MMC_R1_COM_CRC          0x08
#define MMC_R1_ILLEGAL_COM      0x04
#define MMC_R1_ERASE_RESET      0x02
#define MMC_R1_IDLE_STATE       0x01
// Data Start tokens
#define MMC_STARTBLOCK_READ      0xFE  /*when received from card,
                                    indicates that ablock of data will follow*/
#define MMC_STARTBLOCK_WRITE     0xFE      /*when sent to card, indicates that
                                         a block of data will follow*/
#define MMC_STARTBLOCK_MWRITE    0xFC
// Data Stop tokens
#define MMC_STOPTRAN_WRITE       0xFD
// Data Error Token values
#define MMC_DE_MASK              0x1F
#define MMC_DE_ERROR             0x01
#define MMC_DE_CC_ERROR          0x02
#define MMC_DE_ECC_FAIL          0x04
#define MMC_DE_OUT_OF_RANGE      0x04
#define MMC_DE_CARD_LOCKED       0x04
// Data Response Token values
#define MMC_DR_MASK              0x1F
#define MMC_DR_ACCEPT            0x05
#define MMC_DR_REJECT_CRC            0x0B
#define MMC_DR_REJECT_WRITE_ERROR    0x0D
```

```
void mmcInit(void);
unsigned char mmcReset(void);
unsigned char mmcSendCommand(unsigned  char cmd, unsigned long int  argument);
unsigned char mmcRead(unsigned long int  sector, unsigned char* buffer);
unsigned char mmcWrite(unsigned long int  sector, unsigned char* buffer);
unsigned char mmcCommand(unsigned  char cmd, unsigned long int  argument);

//-----------------------------------
void spiInit()
{
   PORTB.7=1;  // set SCK hi
   DDRB.7=1;   // set SCK as output
```

```
    DDRB.6=0;   // set MISO as input
    DDRB.5=1;   // set MOSI as output
    DDRB.4=1;   // SS must be output for Master mode to work
    // SPI initialization
    // SPI Type: Master
    // SPI Clock Rate: 250.000 kHz
    // SPI Clock Phase: Cycle Half
    // SPI Clock Polarity: Low
    // SPI Data Order: MSB First
    SPCR=0x52;
    SPSR=0x00;
}
//-----------------------------------------
unsigned char spiTransferByte(unsigned char data)
{
    unsigned char received = 0;
    SPDR = data;
    while(!(SPSR & (1<<7)));
    received = SPDR;
    return (received);
}
//-----------------------------------------
void mmcInit(void)
{
    // initialize SPI interface
    spiInit();
    // release chip select
    MMC_CS_DDR.MMC_CS_PIN=1;
    MMC_CS_PORT.MMC_CS_PIN=1;
}
//-----------------------------------------
unsigned char mmcReset(void)
{
    unsigned char i;
    unsigned char retry;
    unsigned char r1=0;
    retry = 0;
    do
    {
        // send dummy bytes with CS high before accessing
        for(i=0;i<10;i++)   spiTransferByte(0xFF);
        // resetting card, go to SPI mode
        r1 = mmcSendCommand(MMC_GO_IDLE_STATE, 0);
        retry++;
        if(retry>10) return -1;
    } while(r1 != 0x01);

    retry = 0;
    do
    {
        // initializing card for operation
        r1 = mmcSendCommand(MMC_SEND_OP_COND, 0);
        retry++;
```

```
        if(retry>100) return -1;
    } while(r1);
    // turn off CRC checking to simplify communication
    r1 = mmcSendCommand(MMC_CRC_ON_OFF, 0);
    // set block length to 512 bytes
    r1 = mmcSendCommand(MMC_SET_BLOCKLEN, 512);
    return 0;
}
//-----------------------------------------
unsigned char mmcSendCommand(unsigned char cmd, unsigned long int argument)
{
    unsigned char r1;
    // assert chip select
    MMC_CS_PORT.MMC_CS_PIN=0;
    // issue the command
    r1 = mmcCommand(cmd, argument);
    // release chip select
    MMC_CS_PORT.MMC_CS_PIN=1;
    return r1;
}
//-----------------------------------------
unsigned char mmcRead(unsigned long int sector, unsigned char* buffer)
{
    unsigned char r1;
    unsigned int i;

    MMC_CS_PORT.MMC_CS_PIN=0;
    r1 = mmcCommand(MMC_READ_SINGLE_BLOCK, sector<<9);
    // check for valid response
    if(r1 != 0x00)
        return r1;
    // wait for block start
    while(spiTransferByte(0xFF) != MMC_STARTBLOCK_READ);
    // read in data
    for(i=0; i<0x200; i++)
        *buffer++ = spiTransferByte(0xFF);
    // read 16-bit CRC
    spiTransferByte(0xFF);
    spiTransferByte(0xFF);
    MMC_CS_PORT.MMC_CS_PIN=1;
    return 0;
}
//-----------------------------------------
unsigned char mmcWrite(unsigned long int sector, unsigned char* buffer)
{
    unsigned char r1;
    unsigned int i;

    MMC_CS_PORT.MMC_CS_PIN=0;
    // issue command
    r1 = mmcCommand(MMC_WRITE_BLOCK, sector<<9);
    if(r1 != 0x00)
        return r1;
```

```
        // send dummy
        spiTransferByte(0xFF);
        // send data start token
        spiTransferByte(MMC_STARTBLOCK_WRITE);
        // write data
        for(i=0; i<512; i++)
            spiTransferByte(*buffer++);
      // write 16-bit CRC (dummy values)
      spiTransferByte(0xFF);
      spiTransferByte(0xFF);
      // read data response token
      r1 = spiTransferByte(0xFF);
      if( (r1&MMC_DR_MASK) != MMC_DR_ACCEPT)
          return r1;
      while(!spiTransferByte(0xFF));
      // release chip select
      MMC_CS_PORT.MMC_CS_PIN=1;
      // return success
      return 0;
}
//-----------------------------------------------------
unsigned char mmcCommand(unsigned char cmd, unsigned long int argument)
{
    unsigned char r1;
    unsigned char retry=0;
    //send command
    spiTransferByte(cmd | 0x40);
    spiTransferByte(argument>>24);
    spiTransferByte(argument>>16);
    spiTransferByte(argument>>8);
    spiTransferByte(argument);
    spiTransferByte(0x95);// crc valid only for MMC_GO_IDLE_STATE
                          // end command
    // wait for response
    // if more than 8 retries, card has timed-out
    // return the received 0xFF
    while((r1 = spiTransferByte(0xFF)) == 0xFF)
    if(retry++ > 8) break;
    // return response
    return r1;
}
//-----------------------------------------------------
char write_buf[512],read_buf[512];
void main(void)
{
    unsigned int i;
    unsigned long int sector;
    mmcInit();
    mmcReset();
    for(i=0;i<512;i++)
        write_buf[i]=i;
    sector=5;       //sector 5
    mmcWrite(sector,write_buf);  // write to sector 5
```

```
    delay_ms(1);
    mmcRead(sector,read_buf);     // read of sector 5
    while(1);
}
```

۳-۴-۶- منبع تغذیه:

منبع تغذیه دستگاه از سری کردن دو باتری ۳ ولتی لیتیومی (قابل شارژ) تشکیل شده است که باید حداقل قابلیت تامین انرژی دستگاه برای ۴۸ ساعت را داشته باشد. مقدار جریان دهی منبع تغذیه ۷۰۰ میلی آمپر است.

در قسمتهایی از دستگاه جهت تامین انرژی برخی از المانها باید کاهش ولتاژ داشته باشیم. برای مثال جهت تامین انرژی ویبراتور و بازر از یک مقاومت سری با آنها برای کاهش ولتاژ استفاده می کنیم، چرا که ولتاژ کاری ویبراتور و بازر حدودا باید بین ۱٫۲ تا ۲٫۴ باشد.

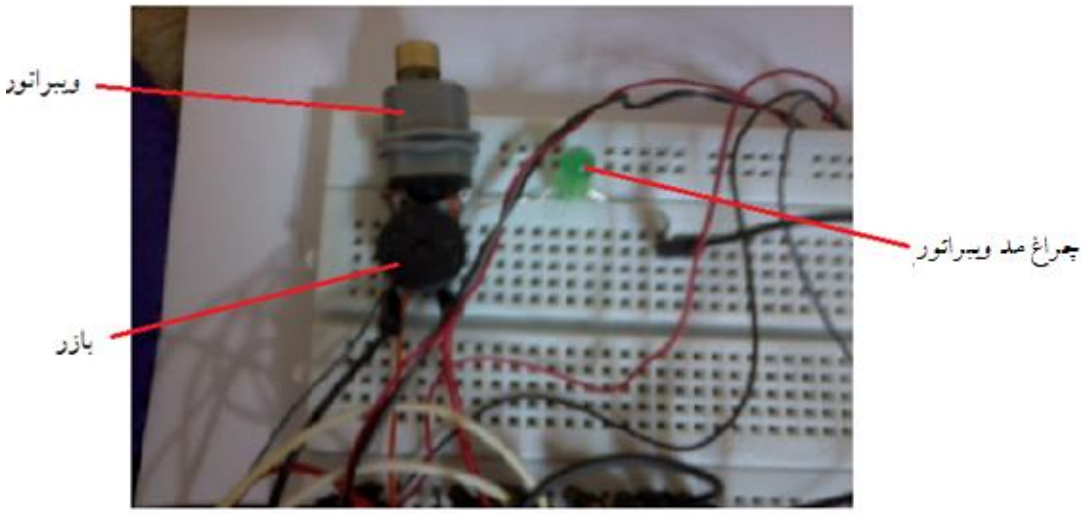
۳-۴-۷- آلارم:

برای دستگاه سه مد آلارم جهت کاربر پسندتر بودن دستگاه در نظر گرفته شده است که عبارتند از:

۱. بازر (فقط صوتی)
۲. ویبراتور (فقط لرزش)
۳. بازر- ویبراتور (صوتی توام با لرزش)

هر یک از سه مد فوق به کاربر اجازه می دهد که در شرایطهای مختلف از مد متناسب با آن استفاده کند، مثلا در مکانهای شلوغ و پر سر و صدا می تواند از مد سوم استفاده کند. کاربر این مدها را از طریق کیبورد دستگاه انتخاب می کند، لذا در برنامه باید سوئیچینگ برای سه مد را در نظر بگیریم که در صورت فشار دادن یکی از سه کلید مربوطه مد مناسب انتخاب شود.

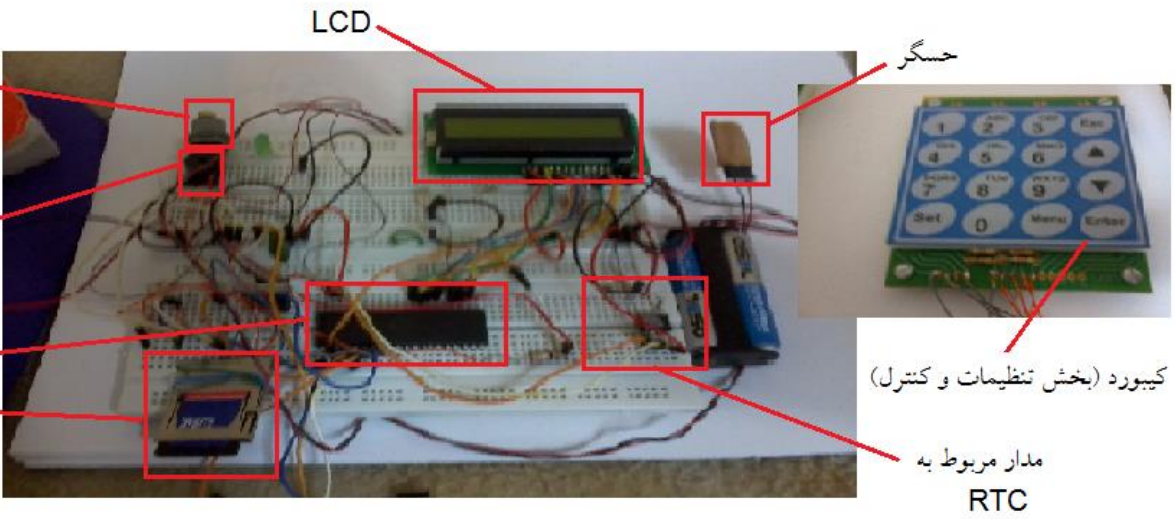
در مد دوم و سوم چراغ سبز رنگی به نشانه اینکه ویبریشن داریم، روشن می شود.



بخش آلارم دستگاه

بحث و نتیجه گیری:

در شکل زیر مدار کامل دستگاه را مشاهده می کنید:



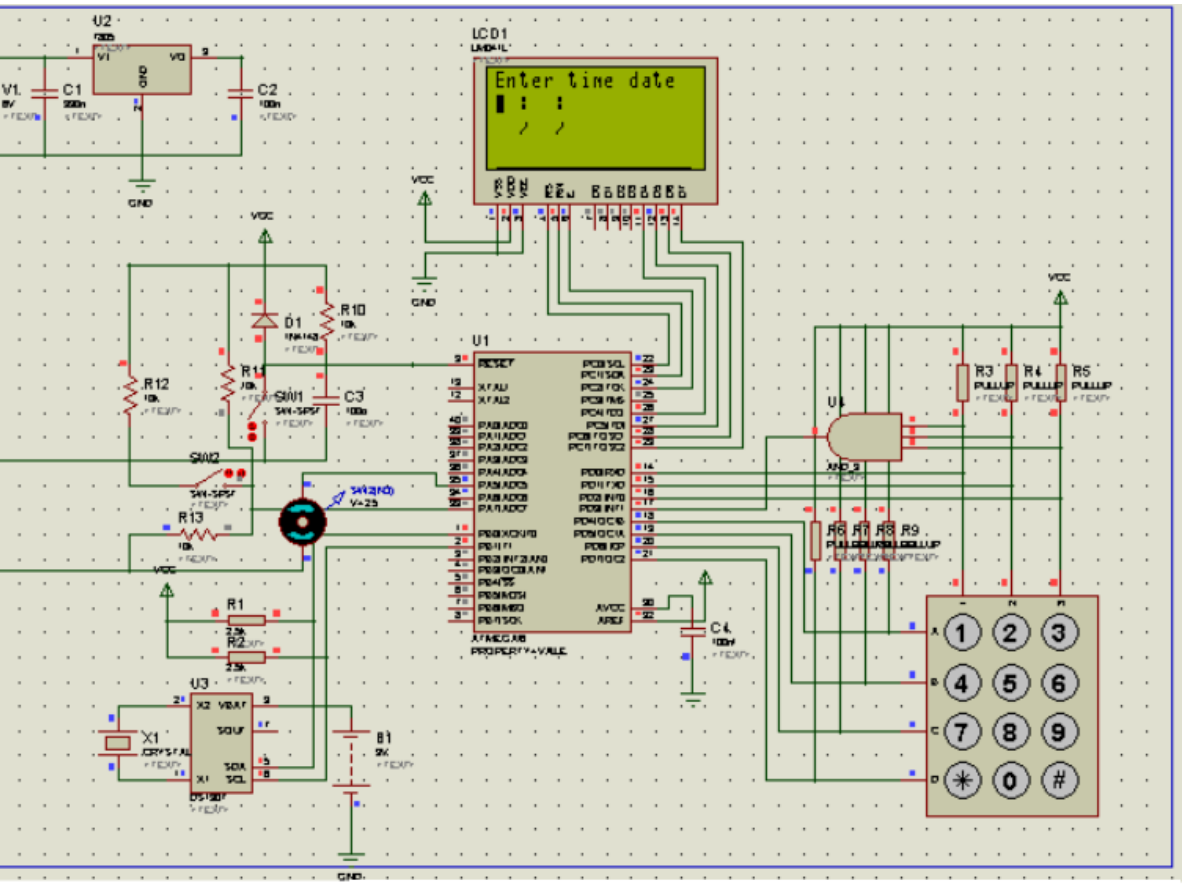
مدار کامل دستگاه

در شکل فوق قسمتهای مختلف دستگاه مشخص شده اند. از آنجا که در فصل قبل راجع به هر کدام از این بخشها توضیح داده شد، نیازی به بیان مجدد آن در اینجا نمی بینیم و فقط در یک نگاه به شرح کلی کارکرد دستگاه پسنده می کنیم:

کاربر بعد از روشن کردن دستگاه برای بار نخست نیاز است که تاریخ و ساعت را تنظیم نماید، سپس سنسور را در قسمت جلویی گردن خود (اندکی به سمت چپ و یا راست سیب انسان که لرزش در آنجا نمایاتر است) نصب کند. از آن به بعد دستگاه خود لرزش را حس می کند و مدت زمان و فرکانس را برای بیمار قابل کنترل می سازد.

دستگاه به لحاظ سخت افزار تکمیل گردید ولی در نرم افزار به مشکلاتی برخوردم.

طراحي دستگاه توسط نرم افزار Proteus با موفقيت شبيه سازي شد. در زير شماتيك دستگاه را توسط Proteus مشاهده مي كنيد :



شبيه سازي دستگاه با PROTEUS

کد (نرم افزار) دستگاه به صورت زیر است:

```
/*****************************************************

This program was produced by the

CodeWizardAVR V2.04.4a Advanced

Automatic Program Generator

© Copyright 1998-2009 Pavel Haiduc, HP InfoTech s.r.l.

http://www.hpinfotech.com


Project :

Version :

Date    : 09/17/2011

Author  : NeVaDa
```

Company : HamidCompany

Comments:

Chip type : ATmega16

Program type : Application

AVR Core Clock frequency: 8.000000 MHz

Memory model : Small

External RAM size : 0

Data Stack size : 256

*****/

#include <mega16.h>

```
// I2C Bus functions
#asm
  .equ __i2c_port=0x18 ;PORTB
  .equ __sda_bit=0
  .equ __scl_bit=1
#endasm
#include <i2c.h>

// DS1307 Real Time Clock functions
#include <ds1307.h>

// Alphanumeric LCD Module functions
#asm
  .equ __lcd_port=0x15 ;PORTC
```

```
#endasm

#include <lcd.h>

#include <delay.h>

#include <bcd.h>

#include <stdlib.h>



unsigned char keycode[4][3]={{1,2,3},{4,5,6},{7,8,9},{11,0,12}};

char  key,*key1;

char  mode=0,count=0;

char  date_time[12];



char findcolumn(char input)

{

switch(input)
```

```
    {
case(0x06):
return 0;
break;
case(0x05):
return 1;
break;
case(0x03):
return 2;
break;
    }
}

interrupt [EXT_INT1] void ext_int1_isr(void)
```

```
{
unsigned char row,column,input;
column=findcolumn(PIND & 0x07);
      delay_ms(20);


      PORTD.4=1;
      delay_ms(20);
input=PIND & 0x07;
if(input==0x07)
      {
row=0;
      }
else
      {
```

```
        PORTD &= 0x07;
        PORTD.5=1;
        delay_ms(20);
input=PIND & 0x07;
if(input==0x07)
        {
row=1;
        }
else
        {
            PORTD &= 0x07;
            PORTD.6=1;
            delay_ms(20);
input=PIND & 0x07;
```

```
if(input==0x07)
        {
row=2;
        }
else
        {
            PORTD &= 0x07;
            PORTD.7=1;
            delay_ms(20);
input=PIND & 0x07;
if(input==0x07)
            {
row=3;
            }
```

```
                PORTD &= 0x07;
            }
            PORTD &= 0x07;
        }
        PORTD &= 0x07;
    }
    PORTD &= 0x07;
key=keycode[row][column];
if(mode==1)
    {
      //_lcd_ready();
      //_lcd_write_data(0x0d);
      date_time[count]=key;
count++;
```

```
itoa(key,key1);
        lcd_puts(key1);
    }
if(key==11)
mode=1;
    //itoa(key,key1);
    //lcd_puts(key1);
    delay_ms(300);
}

unsigned char second,minute,hour,ihour,day,month,tempyear;
char    *asecond,*aminute,*ahour,*aday,*amonth,*atempyear;

void Date()
```

```
{
   lcd_clear();
   lcd_putsf("Enter time date");
   lcd_gotoxy(0,1);
   lcd_putsf("  :  :  ");

   //lcd_putsf("Enter date");
   lcd_gotoxy(0,2);
   lcd_putsf("  /  /  ");

while(count<=11)
   {
if(count%6<2)
        {
```

```
        lcd_gotoxy(count%6,(count/6+1));



    }
else if(count%6<4)
    {
      lcd_gotoxy(count%6+1,(count/6+1));
    }
else if(count%6<6)
    {
      lcd_gotoxy(count%6+2,(count/6+1));
    }
    _lcd_ready();
    _lcd_write_data(0x0d);
    delay_ms(50);
```

```
  }


mode=0;

count=0;

hour   = date_time[0]*10+date_time[1];

minute = date_time[2]*10+date_time[3];

second = date_time[4]*10+date_time[5];

day    = date_time[6]*10+date_time[7];

month  = date_time[8]*10+date_time[9];

tempyear   = date_time[10]*10+date_time[11];


   rtc_set_time(hour,minute,second);

   rtc_set_date(day,month,tempyear);
```

```
   lcd_clear();



   rtc_get_time(&hour,&minute,&second);

   rtc_get_date(&day,&month,&tempyear);




   //ihour=bcd2bin(hour);

itoa(hour,ahour);

   lcd_puts(ahour);

   lcd_putsf(":");



   //iminute=bcd2bin(minute);

itoa(minute,aminute);

   lcd_puts(aminute);
```

```
   lcd_putsf(":");


   //isecond=bcd2bin(second);
itoa(second,asecond);
   lcd_puts(asecond);


   //idate=bcd2bin(date);
itoa(day,aday);
   lcd_puts(aday);
   lcd_putsf("/");


   //imonth=bcd2bin(month);
itoa(month,amonth);
   lcd_puts(amonth);
```

```
   lcd_putsf("/");


   //itempyear=bcd2bin(tempyear);
itoa(tempyear,atempyear);
   lcd_puts(atempyear);
   delay_ms(700);
   lcd_clear();
   lcd_putsf("Normal work");
}


#define ADC_VREF_TYPE 0x20


// Read the 8 most significant bits
// of the AD conversion result
```

```
unsigned char read_adc(unsigned char adc_input)
{
ADMUX=adc_input | (ADC_VREF_TYPE & 0xff);
// Delay needed for the stabilization of the ADC input voltage
delay_us(10);
// Start the AD conversion
ADCSRA|=0x40;
// Wait for the AD conversion to complete
while ((ADCSRA & 0x10)==0);
ADCSRA|=0x10;
return ADCH;
}

// Declare your global variables here
```

```
void main(void)

{

// Declare your local variables here

char counter=0,adcoutput;

// 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=0x60;
```

```
// Port B 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

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=0x00;
```

```
// Port D initialization

// Func7=Out Func6=Out Func5=Out Func4=Out Func3=In

Func2=In Func1=In Func0=In

// State7=0 State6=0 State5=0 State4=0 State3=P State2=P State1=P

State0=P

PORTD=0x0F;

DDRD=0xF0;



// 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: Timer1 Stopped

// Mode: Normal top=FFFFh

// OC1A output: Discon.

// OC1B output: Discon.

// Noise Canceler: Off

// Input Capture on Falling Edge

// Timer1 Overflow Interrupt: Off

// Input Capture Interrupt: Off
```

```
// Compare A Match Interrupt: Off

// Compare B Match Interrupt: Off

TCCR1A=0x00;

TCCR1B=0x00;

TCNT1H=0x00;

TCNT1L=0x00;

ICR1H=0x00;

ICR1L=0x00;

OCR1AH=0x00;

OCR1AL=0x00;

OCR1BH=0x00;

OCR1BL=0x00;



// Timer/Counter 2 initialization
```

```
// Clock source: System Clock

// Clock value: Timer2 Stopped

// Mode: Normal top=FFh

// OC2 output: Disconnected

ASSR=0x00;

TCCR2=0x00;

TCNT2=0x00;

OCR2=0x00;



// External Interrupt(s) initialization

// INT0: Off

// INT1: On

// INT1 Mode: Low level

// INT2: Off
```

```
GICR|=0x80;

MCUCR=0x00;

MCUCSR=0x00;

GIFR=0x80;


// Timer(s)/Counter(s) Interrupt(s) initialization

TIMSK=0x00;


// Analog Comparator initialization

// Analog Comparator: Off

// Analog Comparator Input Capture by Timer/Counter 1: Off

ACSR=0x80;

SFIOR=0x00;
```

```
// ADC initialization

// ADC Clock frequency: 62.500 kHz

// ADC Voltage Reference: AREF pin

// ADC Auto Trigger Source: None

// Only the 8 most significant bits of

// the AD conversion result are used

ADMUX=ADC_VREF_TYPE & 0xff;

ADCSRA=0x87;



// I2C Bus initialization

i2c_init();



// DS1307 Real Time Clock initialization

// Square wave output on pin SQW/OUT: Off
```

```
// SQW/OUT pin state: 0

rtc_init(0,0,0);

//rtc_set_time(2,13,0);

//rtc_set_date(16,9,99);




// LCD module initialization

lcd_init(16);

lcd_putsf("Hi");

lcd_gotoxy(0,1);

lcd_putsf("Welcome");

delay_ms(200);

// Global enable interrupts

#asm("sei")
```

```
lcd_clear();

lcd_putsf("Normal work");

while (1)

    {

    // Place your code here

if(mode)

Date();

adcoutput=read_adc(7);

adcoutput=(adcoutput*5)/255;

        //itoa(adcoutput,asecond);

        //lcd_puts(asecond);



if(adcoutput > 3)

        {
```

```
counter++;
        //itoa(counter,asecond);
        //lcd_puts(asecond);
if(counter>=3)
        {
          PORTA.5=1;
          lcd_clear();
          lcd_putsf("Speaking");
        }
      }
else if(counter>=3)
      {
        PORTA.5=0;
counter=0;
```

```
            lcd_clear();

            lcd_putsf("Normal work");

        }

        delay_ms(1000);

    };

}
```

مراجع:

۱- ره افروز، امیر؛ میکروکنترلر AVR و کاربردهای آن، موسسه ی علمی فرهنگی نص، ۱۳۸۶

۲- پرتوی فر، محمد مهدی- مظاهریان، فرزاد- بیانلو، یوسف؛ مرجع کامل میکروکنترلرهای AVR، انتشارات نص، ۱۳۸۷

۳- دکتر ثمره، یدالله؛ آواشناسی زبان فارسی (آواها و ساخت آوایی هجا)، مرکز نشر دانشگاهی، ویرایش دوم، ۱۳۸۶

۴- پروژه ی کارشناسیطراحی و ساخت دستگاه سنجش گفتار، خانمها تیموری و رضایی، ۱۳۸۸