بنام خدا

طراحی و ساخت ریموت کنترل چند کاناله دارای خاصیت ضبط و ارسال کد (توسط AVR)

میثم ابراهیمی

تابستان ۱۳۹۱

فهرست

فهرست

- **مدولاسیون DPIM**
- **گیرنده های مادون قرمز و مدولاسیون**
- **انواع مدولاسیون**
- **گیرنده فرستنده های RF**
- **فصل چهار(ارتباط سریال سنکرون یا همزمان):**
- **ارتباط سریال آسنکرون یا غیر همزمان**
- **باود یا بیت ریت**

فهرست

مشخصات پروژه

- در این پروژه ما قصد داریم که کد های مشخصی که دارای فرم های خاصی هستند را توسط یک گیرنده مادون قرمز دریافت کرده و آن را دیکد کنیم.
- این کدها توسط یک فرستنده که می تواند ریموت یک دستگاه خاص صوتی و یا تصویری باشد ارسال می شود و یا یک فرستنده مادون قرمز که کدهایی را با مشخصه ای خاص ارسال می کند.
- ماژول ساخته شده ما باید بتواند این کد ها را دریافت و آنها را ذخیره ساخته، اما برای ذخیره کدهای دریافت شده باید به شکل و فرمت کلی آن اشراف داشت و آن را در حافظه میکرو ذخیره کرد.
- از حافظه EEPROMدر AVR استفاده می کنیم.
- همچنین پس از ذخیره این کدها باید بتوان آنها را استفاده کرد و ماژول ساخته شده می تواند جایگزین ریموت مورد نظر شود و داری چند کلید خواهد بود که این کدها را ذخیره می کند و در موقع لازم جایگزین ریموت می شود.

سخت افزار پروژه

-
- AVR ATMEGA32
- میکرو کنترلر ساده و قابل استفاده در پروژه به علت داشتن حافظه و رجیستر لازم .
-
- IRLED TRANSMITTER
- از نوع فرستنده های مورد استفاده در مادون قرمز که مدولاسیون لازم را انجام داده و کد را به صورت پالسی ارسال می کند .
-
- چشم پارسی (IRLED RECIEVER)
- گیرنده ای که امواج مادون قرمز را دریافت کرده و دارای برد لازم و دمدولاسیون مورد نظر است تا به میکرو کد صفر و یک را بدهد .
-
- REGULATOR 5V (7805)
- به دلیل بهره بردن از منبع ولتاژ که همان باتری قابل شارژ است ما نیاز داریم برای تغذیه موارد بالا از رگولاتور استفاده نماییم .

سخت افزار پروژه

-
- BATTERY
- باتری قابل شارژ تا زمینه کار و تغذیه مدار محیا باشد .
-
- PUSH BUTTON
- کلید استفاده شده در مدار که هر یک با فشار دادن به سمت داخل فعال می شود .
-
- LED
- لامپ های پر نور که در چند مورد با خاموش و روشن شدن حالت کاری مدار را شرح میدهد .
-
- CAPACITOR
- خازن ولتاژی که بیشتر جهت گرفتن نویز مدار و ارسال و دریافت کد سالم کاربرد دارد .
-
- RESISTOR
- مقاوتی که جهت پول اپ کردن مدار کاربرد دارند
-
- CRYSTAL
- کریستالی که جهت ایجاد فرکانس کاری لازم در میکرو به کار برده می شود .

مفاهيم اوليه در سيستم هاى مادون قرمز

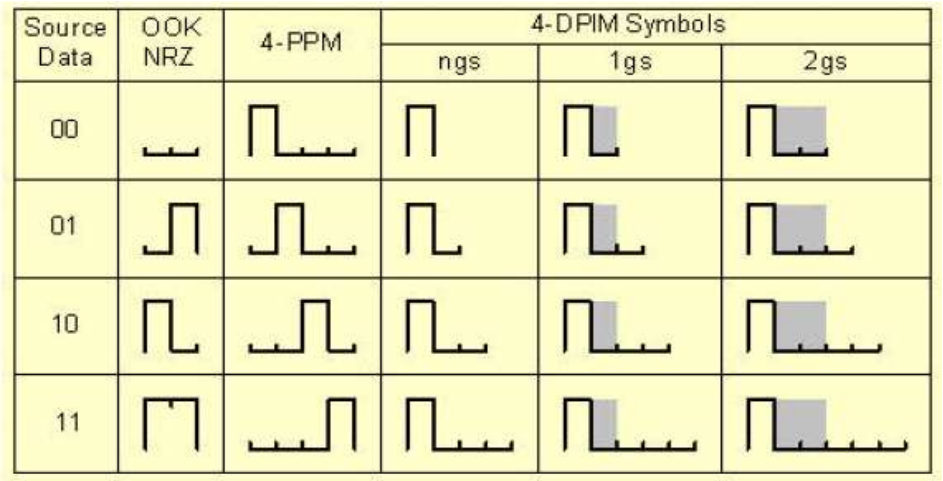
- انتشار IRشبيه نور مرئي است؛ بنابراين بدليل خيلي كوچك بودن طول موج، شكست نور نقش مهمي در آن ندارد. اگر مسير ديد مستقيم بين فرستنده و گيرنده با يك مانع كدر مسدود شود ارتباط بين آنها از طريق بازتاب برقرار مي گردد. چون ديوارها و سقف نوعا بين ٦٠ تا ٩٠ درصد نور ورودي را بطور پراكنده (Defuse) بازتاب مي كنند ، ارسال از راههاي غير از مسير ديد مستقيم امكان پذير مي باشد.

فصل سوم(بررسی تکنیک های مدولاسیون دیجیتال IR):

- **ویژگی های هر یک از فرمت های ارسالی دیجیتال**
- OOK(NRZ)پهناي باند كم ، داراي مؤلفه DCنداشتن قابليت سنكرون سازي ، عدم ارسال مي تواند با ارسال ۰ اشتباه گرفته شود ، نياز به توان ارسالي بالا دارد .
- NRZIوجود مولفه DCنداشتن قابليت سنكرون سازي، با يك جابجايي بين زمان بندي فرستنده و گيرنده در حين ارسال دنباله بزرگي از صفرها با يك ها امكان از دست دادن همزماني وجود دارد. اين روش براي كاربردهاي ارسال سيگنال مناسب نيست و در ضبط مغناطيسي ديجيتال بكار مي رود.
- كد منچستر مزيت مهم اين روش نداشتن مولفه DCو امكان سنكرون سازي بالا مي باشد (بدليل وجود گذار از حالت HIGH به LOWو برعكس در بيت هاي ۱ و ۰)

فصل سوم(بررسی تکنیک های مدولاسیون دیجیتال IR):

- **مدولاسيون DPIM**
- اخيراً يك تكنيك مدولاسيون پالسي در مخابرات نوري بي سيم معرفي شده است كه اطلاعات را در فاصله نسبي بين پالسهاي متوالي (شیارهای خالی) قرار مي دهد اين مدولاسيون DPIM يا PPMتفاضلي نام دارد (اين فاصله مضربي از يك پهناي زماني اوليه T است.) مي توان از يك يا چند شيار محافظ نيز براي اطمينان از عدم دريافت نادرست استفاده نمود.



- شكل (٥-٣) : تكنيك مدولاسيون پالسي

فصل سوم(بررسی تکنیک های مدولاسیون دیجیتال IR):

- DPIMدر مقايسه با PPMاز لحاظ ظرفيت انتقال و ملاحظات پهناي باند كارايي بيشتري دارد ولي از لحاظ توان ارسالي و نرخ خطاي بيت چنين نيست . به دلیل ساختار با طول متغیر دنباله کلمات در DPIM ارزیابی طیفي آن به سادگي PPM یا فرمتهای دیگر نیست.

فصل سوم(بررسی تکنیک های مدولاسیون دیجیتال IR):

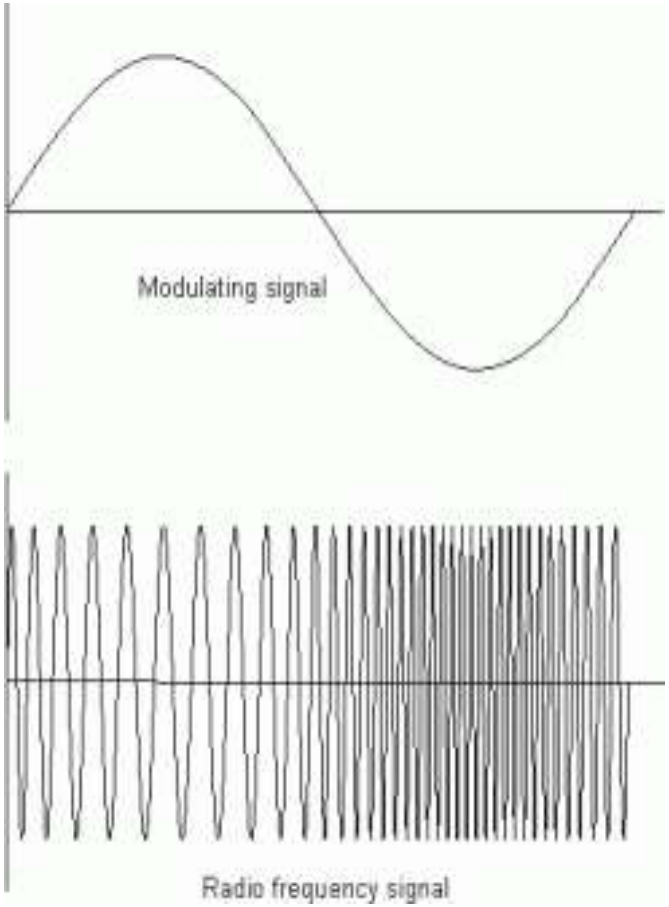
- **گیرنده های مادون قرمز و مدولاسیون**
- مطالبی که تا کنون مطرح شدند، زمینه ای برای ورود به بحث اصلی پروژه می باشند. در این قسمت، چگونگی برقراری ارتباط از طریق واسط های مادون قرمز را مورد بررسی قرار می دهیم.
- می دانیم برای برقراری یک ارتباط سریال، حداقل به دو سیم بین دو دستگاه فرستنده و گیرنده نیاز داریم.
- البته این در حالتی است که بخواهیم از یک رابط الکتریکی به عنوان محیط انتقال استفاده کنیم. در این پروژه، قصد داریم تا محیط انتقال را تغییر دهییم و بجای سیم، از امواج مادون قرمز بهره بریم. برای تغییر محیط انتقال باید به دو مساله توجه کنیم:
- تجهیزات قابل استفاده در محیط جدید یا همان گیرنده و فرستنده
- نوعی مدولاسیون برای انطباق با شرایط محیط انتقال جدید
- ابتدا مدولاسیون را مورد بررسی قرار می دهیم. همانطور که می دانیم، خروجی پایه TXD در میکروکنترلرها همواره یکی از سطوح منطقی صفر یا یک را داراست؛ حال آن که توسط فرستنده یا گیرنده مادون قرمز می توانیم یک موج مربعی با فرکانس خاص را ارسال یا دریافت کنیم.
- اگر کانال مخابراتی شامل فضای آزاد باشد در این صورت برای انتشار و دریافت سیگنال آنتن هایی مورد نیاز است طول این آنتن ها متناسب با طول موج سیگنال فرستاده شده است. بسیاری از سیگنال های صوتی دارای مولفه فرکانسی ۱۰۰ هرتز یا پایین تر هستند. برای ارسال این سیگنال ها اگر سیگنال مستقیما انتشار یابد به آنتنهایی با طول حدود 300km نیاز است. اما اگر از مدولاسیون برای سوار کردن سیگنال بر روی یک فرکانس حامل مثلا 100MHz استفاده کنیم در این صورت طول آنتن ها حدود یک متر خواهد بود.

فصل سوم(بررسی تکنیک های مدولاسیون دیجیتال IR):

- **انواع مدولاسیون**
- مدولاسیون دامنه (AM): سطح یا دامنه ی سیگنال حامل بر اساس تغییرات سیگنال پیام تغییر داده می‌شود.
- مدولاسیون فرکانس (FM): فرکانس سیگنال حامل بر اساس تغییرات سیگنال پیام تغییر داده می‌شود.
- مدولاسیون فاز (PM): فاز سیگنال حامل بر اساس تغییرات سیگنال پیام تغییر داده می‌شود.

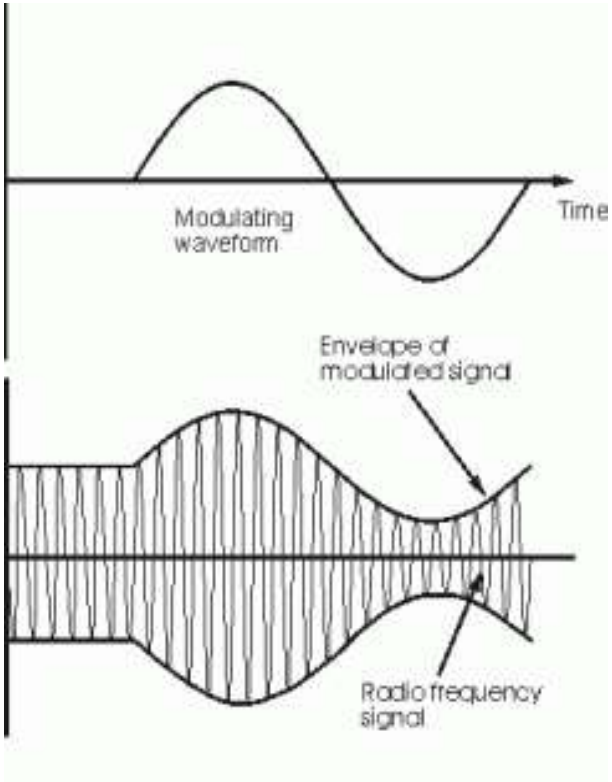
فصل سوم(بررسی تکنیک های مدولاسیون دیجیتال IR):

شکل (٦-٣) : fm مدولاسیون



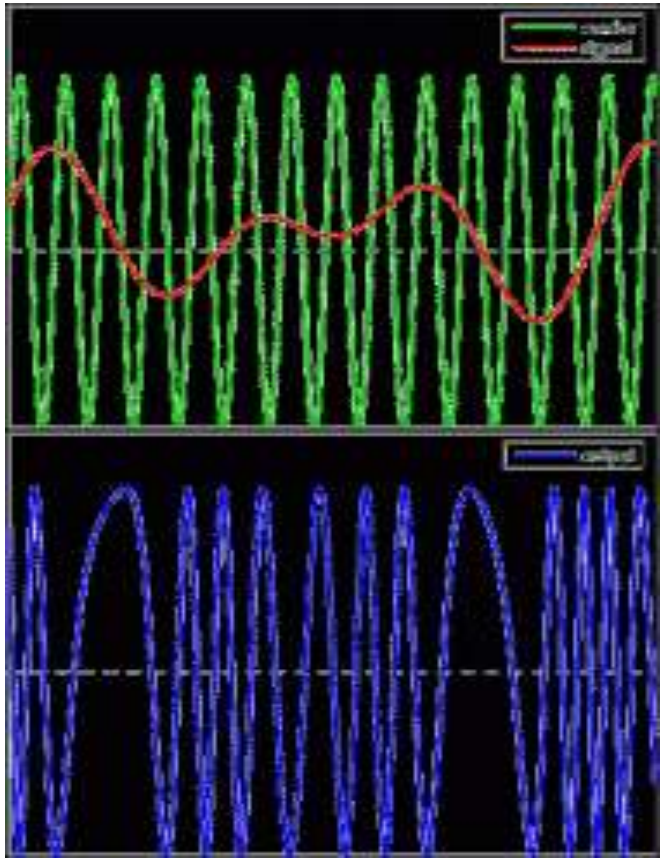
فصل سوم(بررسی تکنیک های مدولاسیون دیجیتال IR):

شکل (۷-۳) :
amمدولاسیون



فصل سوم(بررسی تکنیک های مدولاسیون دیجیتال IR):

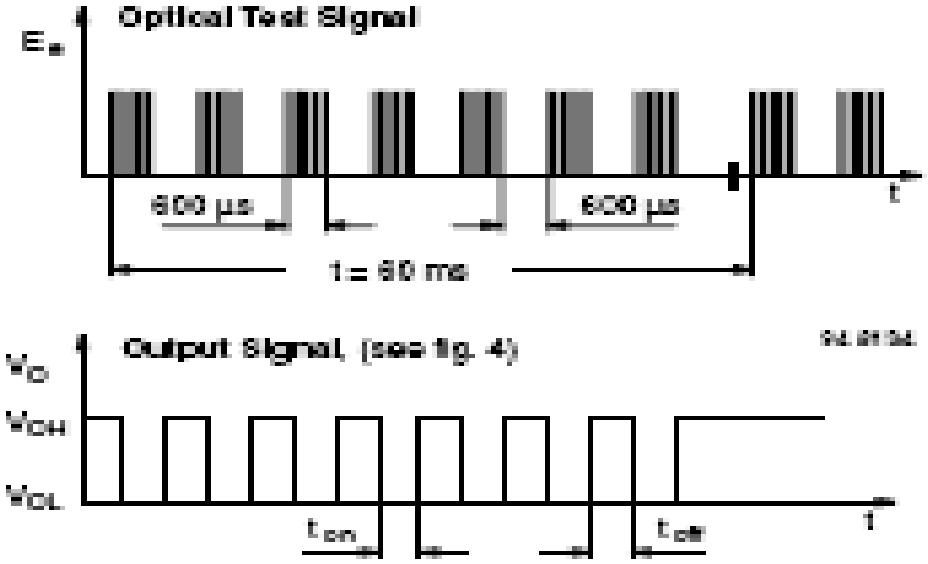
شکل (۸-۳) : مدولاسیون pm



فصل سوم(بررسی تکنیک های مدولاسیون دیجیتال IR):

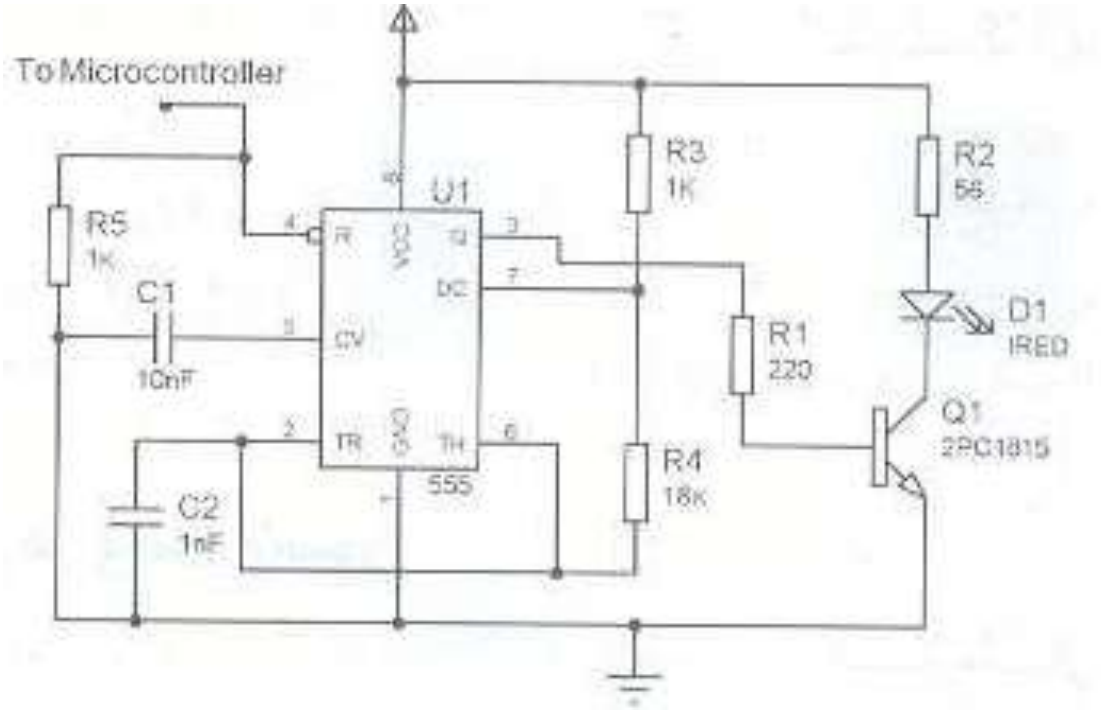
• گیرنده فرستنده های RF

- فرستنده و گیرنده های RF عموما برای ارسال و دریافت داده های آنالوگ و یا دیجیتال در باند فرکانسی رادیویی (۵۰۰ کیلو هرتز تا ۱۰۸ مگاهرتز) طراحی می گردند .
- مدولاسیون استفاده شده در این پروژه بسیار ساده و آسان است ؛ به این صورت که بجای صفر منطقی ، یک موج مربعی مادون قرمز را با فرکانس ثابت ارسال می کنیم و بجای یک ، هیچ سیگنالی ارسال نخواهد شد.
- شکل (۹-۳) : بایت مدوله



فصل سوم(بررسی تکنیک های مدولاسیون دیجیتال IR):

- ساخت مدار الکترونیکی که بتواند ما را به این هدف برساند ، چندان مشکل نیست در شکل زیر یک نمونه ساده و در عین حال کار آمد از چنین مداری آورده شده است :



شکل : (۱۰-۳) مدار مدولاسیون

فصل چهار(ارتباط سریال سنکرون یا همزمان):

- پس از کسب اطلاعات درباره پرتو مادون قرمز، اکنون نکاتی را در مورد انتقال اطلاعات به صورت سریال مطرح می کنیم؛ چرا که مبنای کار این پروژه، ارتباط سریال بین دو میکروکنترلر است.
- واسط سریال Serial Interface یکی از ویژگی ها و ابزارهای مفید در اکثر میکروکنترلرهای امروزی محسوب می شود. میکروکنترلرهای AVR نیز از این قاعده مستثنی نیستند و تقریبا در تمام آنها ابزار ارتباط سریال وجود دارد. اگر با میکروکنترلر آشنا باشید، حتما پایه های TXD (ارسال داده) و RXD (دریافت داده) را دیده اید. البته واسط سریال، یکی از چند نوع پروتکل ارتباطی است که در میکروکنترلرهای AVR به کار گرفته شده اند. ارتباط SPI,GTAG,12C را نیز می توان به عنوان سایر واسط های ارتباطی نام برد.
- در یک تقسیم بندی کلی، ارتباط سریال را می توان به دو دسته ارتباط همزمان یا سنکرون و غیرهمزمان یا آسنکرون افراز کرد.
- در ارتباط سریال به صورت سنکرون، حداقل به سه سیم پالس ساعت، داده و زمین برای برقراری ارتباط بین دو وسیله نیاز است.
- پالس ساعت، وظیفه همزمانی ارسال و دریافت داده را بر عهده دارد. فرستنده، همزمان با قرار دادن اطلاعات روی پایه Data، پالس ساعت را نیز روی پایه Clk تولید می کند و گیرنده با دریافت لبه پالس ساعت (لبه بالا رونده یا پایین رونده)، اطلاعات ارسالی را به صورت بیت به بیت دریافت می کند.
- ارتباط سریال سنکرون معمولا زمانی استفاده می شود که بخواهیم حجم وسیعی از اطلاعات را منتقل کنیم.
- البته در بعضی اوقات،به دلیل نیاز به دست کم سه سیم ارتباطی بین فرستنده و گیرنده، ترجیح می دهیم که از نوع دیگر ارتباط یعنی ارتباط آسنکرون استفاده کنیم؛ چرا که در این نوع ارتباط می توان تنها با دو سیم، بین گیرنده و فرستنده ارتباط برقرار کرد.

فصل چهار(ارتباط سریال سنکرون یا همزمان):

- **ارتباط سریال آسنکرون یا غیر همزمان**

- همان طور که اشاره شد، در ارتباط یک طرفه آسنکرون، تنها به دو سیم (داده و زمین) نیاز داریم؛ زیرا در این نوع ارتباط برای ارسال و دریافت اطلاعات، به جای پالس ساعت از مفهوم سرعت انتقال بیت یا باود Boud استفاده می کنیم. به این ترتیب، فرستنده، اطلاعات را با همان سرعتی که روی خط می گذارد که گیرنده دریافت می کند و ما در این بین به یک قالب بندی تعریف شده برای ابتدا و انتهای داده ارسالی نیاز داریم.
- معمولا یک بیت به عنوان بیت ابتدا، یک بیت به عنوان بیت انتها و ۸ بیت به عنوان قالب اطلاعات ارسالی با بایت ارسالی در نظر گرفته می شود؛ بنابراین در قالب بندی گفته شده برای ارسال یک بایت اطلاعات مجبور هستیم ۱۰ بیت ارسال کنیم. بنا به قرارداد، بیت ابتدا همواره صفر و بیت انتها همواره یک است . زمانی که هیچ داده ای ارسال نمی شود، فرستنده، پایه TXD را به سطح یک می برد و در نتیجه گیرنده با دریافت اولین صفر پس از بیکاری خط، متوجه می شود که باید برای دریافت یک بایت آماده شود و با توجه به یکسان بودن میزان ارسال و دریافت، ۸ بیت بعدی را یکی پس از دیگری دریافت می کند تا اینکه بایت ارسالی به طور کامل دریافت شود. گاهی اوقات یک بیت توازن نیز برای تشخیص خطای احتمالی در ارسال و دریافت اطلاعات به قالب بندی فوق اضافه می شود که در نتیجه بیت های مورد نیاز برای ارسال یکی افزایش می یابد.

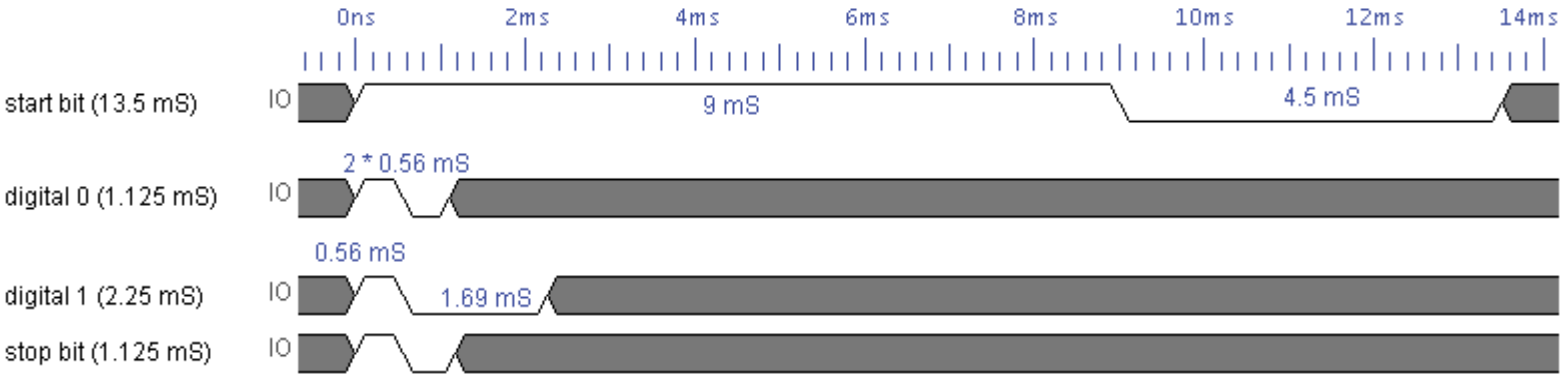
فصل چهار(ارتباط سریال سنکرون یا همزمان):

- **باود یا بیت ریت**
- باود، همان سرعت انتقال اطلاعات است که با واحد بیت بر ثانیه اندازه گیری می شود. مثلا، وقتی می گوییم دستگاهی با میزان باود ۹۶۰۰ کار می کند، یعنی در زمان ارسال، در هر ثانیه قادر به انتقال ۹۶۰۰ بیت می باشد یا به عبارت دیگر، زمان ارسال هر بیت ثانیه است و در نتیجه برای انتقال یک بایت داده، با قالب بندی یک بیت ابتدا و یک بیت انتها و بدون بیت توازن که در مجموع شامل ۱۰ بیت می شود، زمانی معادل ثانیه لازم است.
- شاید چنین تصور شود که می توان توسط دستگاهی با ۹۶۰۰ باود، ۹۶۰ بایت را در مدت یک ثانیه منتقل کرد؛ اما دراصل چنین کاری ممکن نیست، چرا که بین دو ارسال متوالی، زمانی به اندازه زمان بیکاری (زمان مرده) وجود دارد که در نتیجه آن نمی توان به میزان ۹۶۰ بایت در ثانیه رسید.
- در ارتباط سریال آسنکرون، باود در هر دو وسیله گیرنده و فرستنده باید برابر باشد. البته اختلاف چند درصدی میزان باود نیز قابل قبول است، به شرطی که از حد معین تجاوز نکند؛ چون در غیر این صورت اطلاعات ارسالی را نمی توان به درستی دریافت کرد.
- از یک دیدگاه دیگر، می توان ارتباط سریال را به سه نوع یک طرفه، نیمه دو طرفه و تمام دو طرفه تقسیم کرد. در ارتباط یک طرفه، داده ها توسط یک دستگاه ارسال و توسط دیگری دریافت می شوند.
- در این نوع ارتباط دستگاهی که به عنوان فرستنده استفاده شده است، اطلاعات را ارسال می کند و از چگونگی دریافت یا از درستی و نادرستی آن هیچ اطلاعی ندارد. ماوس های سریال در کامپیوترهای شخصی، از این نوع ارتباط استفاده می کنند.
- در ارتباط تمام دوطرفه، هر دو دستگاه می توانند هم عمل ارسال و هم عمل دریافت را انجام دهند. در چنین ارتباطی می توان مسئله آشکارسازی و تصحیح خطا را مطرح کرد؛ در نتیجه، افزودن بیت توازن و سایر روش های آشکارسازی خطا به قاب ارسال داده ها مورد توجه قرار می گیرد. پرینترهای سریال از این نوع ارتباط بهره می برند.
- نوع دیگر ارتباط نیم دو طرفه است که در آن هر دو دستگاه می توانند عمل ارسال و دریافت را، البته نه به طور همزمان، انجام دهند. در واقع در هر لحظه، یک دستگاه فرستنده و دیگری گیرنده است.
-

فصل پنج(فرمت کلی کد ها و نحوه دیکد کردن آنها):

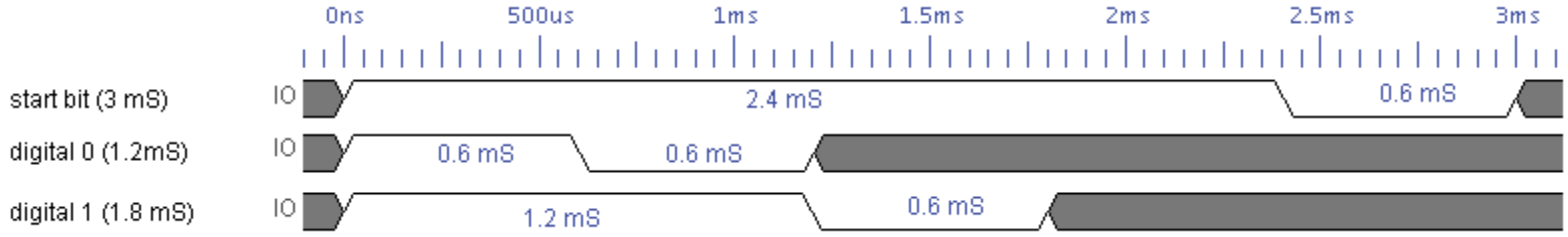
Waveforms

2-NE(



4-

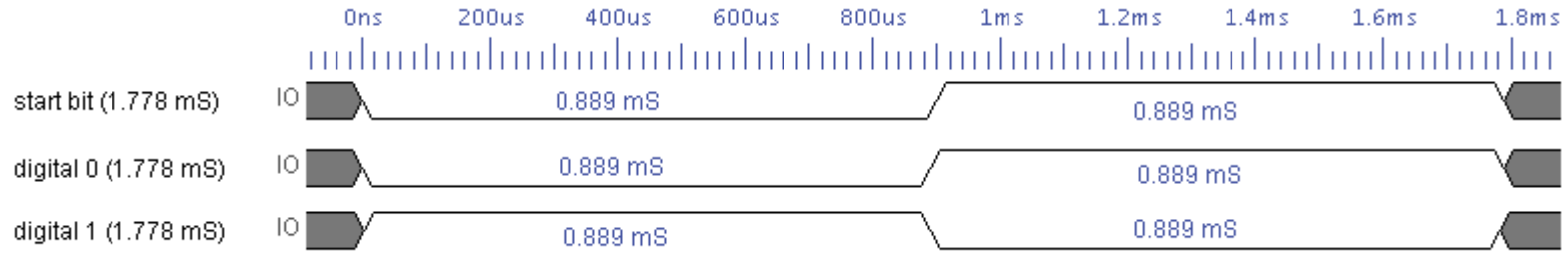
SIRCS



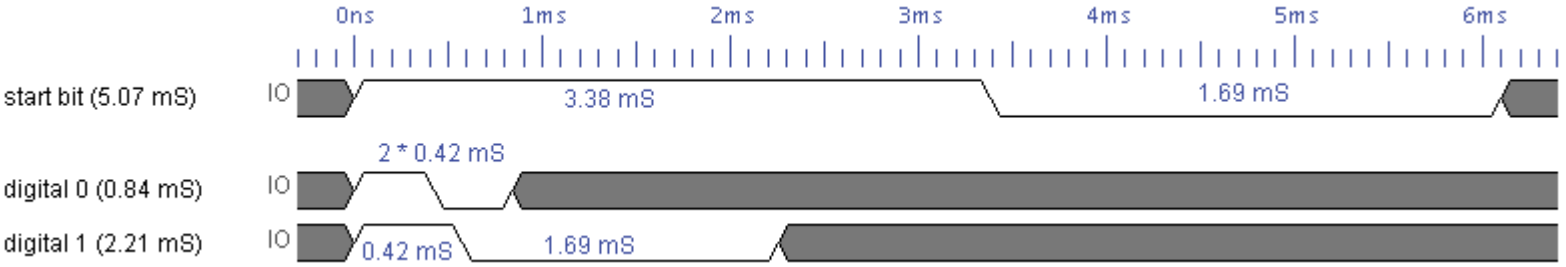
فصل پنج(فرمت کلی کد ها و نحوه دیکد کردن آنها):

- **Waveforms**

5-RC5



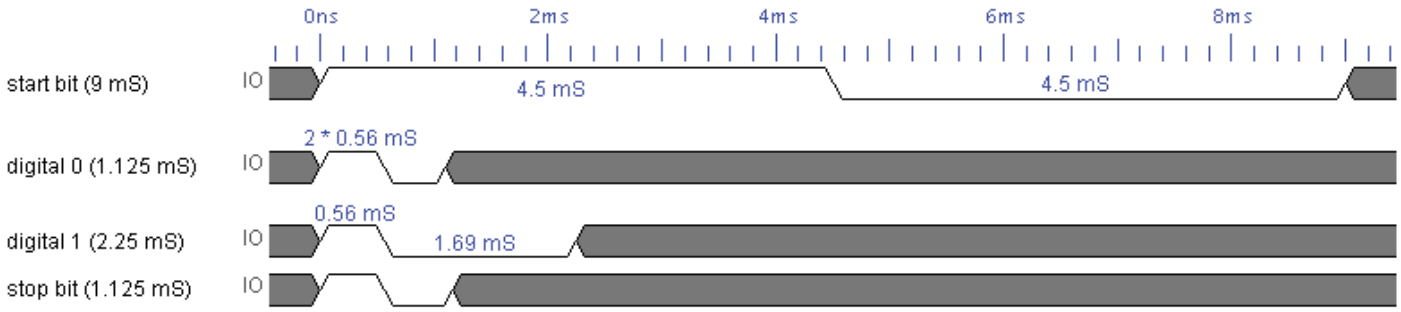
7-JAPAN



فصل پنج(فرمت کلی کد ها و نحوه دیکد کردن آنها):

- **Waveforms**

8-SAMSUNG



- کنترل ها کلا از دو پرتکل استفاده می کنند (کار به نحوه ی فرستادن داریم نه طول پالس چون قرار است دستگاه خودش طول پالس را یاد بگیرد)
- این که کنترل اول یه فرمان می فرستد سپس مدام پشت سر هم می گوید فرمان ارسال شده را اجرا کند.
- این که یک کد را ارسال می کند و دوباره باز همان را تکرار می کند (فکر می کنم تلویزیون های قدیمی اینطور باشند)

فصل شش(پروگرام لازم جهت راه اندازی AVR):

```
#include <tiny2313.h>
#include <delay.h>
#include <sleep.h>
#include <stdio.h>
unsigned char i,x;
interrupt [EXT_INT0] void ext_int0_isr(void)
{
for(i=0;i<2;i++);
i=i;
i=i;
PORTD.0=~PORTD.0;
i=i;
i=i;
i=i;
}
interrupt [EXT_INT1] void ext_int1_isr(void)
{
if(PIND.5==0)
     x=1;

  if(PIND.6==0)
    x=2;

  if(PINB.0==0)
    x=3;

  if(PINB.1==0)
    x=4;

  if(PINB.2==0)
    x=5;

  if(PINB.3==0)
    x=6;

  if(PINB.4==0)
    x=7;

  if(PINB.5==0)
    x=8;

  if(PINB.6==0)
    x=9;

  if(PINB.7==0)
    x=10;

}
while (1)
      {

      if(x==1)
        {
          putsf("H01");
          x=0;
          powerdown();
        }

      if(x==2)
        {
          putsf("H02");
          x=0;
          powerdown();
        }

      if(x==3)
        {
          putsf("H03");
          x=0;
          powerdown();
        }

      if(x==4)
        {
          putsf("H04");
          x=0;
```

فصل شش(پروگرام لازم جهت راه اندازی AVR):

```
    powerdown();
  }

if(x==5)
  {
    putsf("H05");
    x=0;
    powerdown();
  }

if(x==6)
  {
    putsf("H06");
    x=0;
    powerdown();
  }

if(x==7)
  {
    putsf("H07");
    x=0;
    powerdown();
  }

if(x==8)
  {
    putsf("H08");
    x=0;
    powerdown();
  }

if(x==9)
  {
    putsf("H09");
    x=0;
    powerdown();
  }

if(x==10)
                {
                  putsf("H10");
                  x=0;
                  powerdown();
                }
              #asm("wdr")
};
}
```

فصل شش(پروگرام لازم جهت راه اندازی AVR):

- **شرح برنامه فرستادن**
- در این برنامه وقفه خارجی صفر به صورت low level تعریف شده است بنابراین با زدن هر یک از کلید ها تا زمانی که کلید مورد نظر پایین است زیر روال مربوط به این وقفه فراخوانی و اجرا می شود .
- در زیر روال وقفه ابتدا پورت A و پورت B را می خوانیم سپس کنترل می کنیم که کدام یک از بیت های پورت A یک شده است، در نهایت اگر یکی از کلید های متصل به این پورت زده شده باشد کد مربوط به آن کلید توسط دستور Printbin #2,x ارسال خواهد شد. البته به این شرط که در این لحظه هیچ یک از کلید های متصل به پورت B فشرده نشده باشد. همین مطلب در مورد پورت B نیز صادق است .
- به این ترتیب اگر دو یا چند کلید به طور همزمان فشار داده شوند هیچ کدی ارسال نخواهد شد .
- همان طور که گفته شد وقفه حساس به سطح در نظر گرفته شده است در نتیجه تا زمانی که دست ما روی کلید باشد کد مورد نظر به طور متناوب ارسال خواهد شد زمان بین دو ارسال متوالی ۴۰ میلی ثانیه در نظر گرفته شده است البته در مورد کدهای ۱۱ و ۱۲ این زمان ۸۰ میلی ثانیه خواهد بود بدان دلیل که این دو عدد برای تنظیم یک کمیت گسسته – مثل تنظیم نور یک لامپ به صورت PWM– در گیرنده در نظر گرفته شده اند در نتیجه در گیرنده کمیت مورد نظر با سرعت یک پله در هر ۸۰ میلی ثانیه افزایش یا کاهش خواهد داشت. کدهای ۱ تا ۹ به ترتیب برای ارسال اعداد ۱ تا ۹ کد ۱۰ برای ارسال عدد ۰ و کد ۱۳ برای روشن و خاموش کردن دستگاه گیرنده در نظر گرفته شده اند بنابراین اگر گیرنده روشن باشد با زدن این کلید خاموش می شود (و برعکس) .
- زمانی که هیچ کلیدی را نزده باشیم وقفه ای اتفاق نمی افتد و برنامه در حلقه DO...LOOP اجرا می شود در نتیجه میکرو کنترلر با دستور POWER DOWN به حالت مصرف توان کم می رود .
- بدیهی است که با زدن هر یک از کلیدها یک وقفه به میکرو کنترلر اعمال و میکرو کنترلر از حالت مصرف توان کم خارج می شود و عملیات مورد نظر را انجام می دهد .

فصل شش(پروگرام لازم جهت راه اندازی AVR):

- در حالت در یافت

```
#include <tiny2313.h>
#include <stdio.h>
#include <delay.h>
#include <string.h>

unsigned char s[3],i;

interrupt [USART_RXC] void
usart_rx_isr(void)
{

  s[i]=UDR;
  i++;
  if(i==3)
    {
      i=0;
    }
}

while (1)
      {

        if(strcmpf(s,"H01")==0)
          {

           PORTD.3=1;
           delay_ms(50);
           PORTD.3=0;
           delay_ms(100);
            if(PIND.4==1)
                {
                  PORTD.5=1;
                  delay_ms(100);
                  PORTD.5=0;
                  delay_ms(100);
                  i=0;
                  s[0]=0;
                  s[1]=0;
                  s[2]=0;
                  UDR=0;
                }

            else if(PIND.4==0)
              {
                  PORTD.5=~PORTD.5;
                  i=0;
                  s[0]=0;
                  s[1]=0;
                  s[2]=0;
                  UDR=0;
              }

          } // chanel 1

        if(strcmpf(s,"H02")==0)
          {

           PORTD.3=1;
           delay_ms(50);
           PORTD.3=0;
           delay_ms(100);
          if(PIND.4==1)
              {
                PORTD.6=1;
                delay_ms(100);
                PORTD.6=0;
                delay_ms(100);
```

فصل شش(پروگرام لازم جهت راه اندازی AVR):

```
      i=0;
      s[0]=0;
      s[1]=0;
      s[2]=0;
      UDR=0;
    }

  else if(PIND.4==0)
    {
      PORTD.6=~PORTD.6;
      i=0;
      s[0]=0;
      s[1]=0;
      s[2]=0;
      UDR=0;
    }

} // chanel 2

if(strcmpf(s,"H03")==0)
{

  PORTD.3=1;
 delay_ms(50);
 PORTD.3=0;

  delay_ms(100);
if(PIND.4==1)
    {

      PORTB.0=1;
      delay_ms(100);
      PORTB.0=0;
      delay_ms(100);
```

```
      i=0;
      s[0]=0;
      s[1]=0;
      s[2]=0;
      UDR=0;
    }

  else if(PIND.4==0)
    {

      PORTB.0=~PORTB.0;
      i=0;
      s[0]=0;
      s[1]=0;
      s[2]=0;
      UDR=0;
    }

 } // chanel 3

if(strcmpf(s,"H04")==0)
 {
 PORTD.3=1;
  delay_ms(50);
  PORTD.3=0;


 delay_ms(100);
 if(PIND.4==1)
     {

      PORTB.1=1;
      delay_ms(100);
      PORTB.1=0;
```

فصل شش(پروگرام لازم جهت راه اندازی AVR):

```
        delay_ms(100);
        i=0;
        s[0]=0;
        s[1]=0;
        s[2]=0;
        UDR=0;
      }

   else if(PIND.4==0)
      {

        PORTB.1=~PORTB.1;
        i=0;
        s[0]=0;
        s[1]=0;
        s[2]=0;
        UDR=0;
      }

} // chanel 4

if(strcmpf(s,"H05")==0)
{

PORTD.3=1;
 delay_ms(50);
 PORTD.3=0;

delay_ms(100);
if(PIND.4==1)
      {
        PORTB.2=1;
        delay_ms(100);
        PORTB.2=0;
```

```
        delay_ms(100);
        i=0;
        s[0]=0;
        s[1]=0;
        s[2]=0;
        UDR=0;
      }

   else if(PIND.4==0)
      {
        PORTB.2=~PORTB.2;
        i=0;
        s[0]=0;
        s[1]=0;
        s[2]=0;
        UDR=0;
      }

} // chanel 5

if(strcmpf(s,"H06")==0)
{

PORTD.3=1;
 delay_ms(50);
 PORTD.3=0;

delay_ms(100);
if(PIND.4==1)
      {
        PORTB.3=1;
        delay_ms(100);
        PORTB.3=0;
        delay_ms(100);
```

فصل شش(پروگرام لازم جهت راه اندازی AVR):

```
            i=0;
            s[0]=0;
            s[1]=0;
            s[2]=0;
            UDR=0;
         }

      else if(PIND.4==0)
         {
            PORTB.3=~PORTB.3;
            i=0;
            s[0]=0;
            s[1]=0;
            s[2]=0;
            UDR=0;
         }

   } // chanel 6

#asm("wdr")


  };
}
```

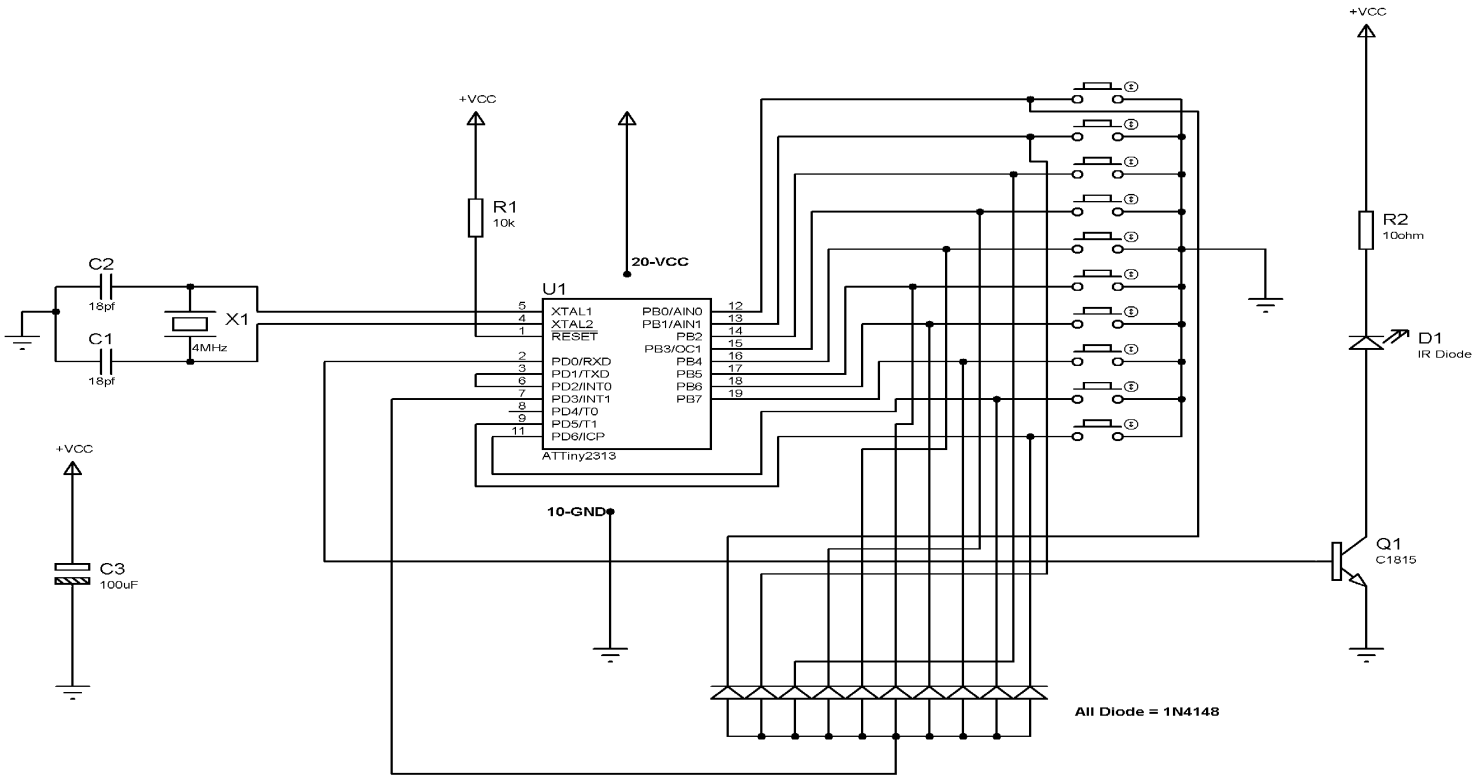
فصل شش(پروگرام لازم جهت راه اندازی AVR):

- ## شرح برنامه دریافت

- برنامه در گیرنده طوری طراحی شده است که به محض دریافت یک بایت از طریق واسط سریال وقفه پورت سریال کنترل برنامه را به زیرروال مورد نظر می برد. در ابتدای این زیرروال بیت B1 برابر صفر می شود و در ادامه اگر بایت دریافتی جدید که در متغیر RCC_B قرار می گیرد با بایت دریافتی پیشین که در متغیر C قرار گرفته است برابر باشد، B1 برابر صفر می شود .
- به این ترتیب می توانیم با کنترل B1 تشخیص دهیم که آیا بایت دریافتی برای اولین بار دریافت می شود یا خیر. اهمیت این کار زمانی مشخص می گردد که بخواهیم تعداد بایت های یکسان را که با زدن کلیدی در فرستنده به طور متناوب ارسال می شوند تنها یکبار مورد استفاده قرار دهیم.
- به عنوان مثال فرض کنید کلید مربوط به عدد ۷ در فرستنده زده شده باشد، در این صورت ممکن است کد مورد نظر با توجه به زمان تناوب ۴۰ میلی ثانیه برای ارسال، دهها بار فرستاده شود به عبارت دیگر تا زمانی که کاربر این کلید را فشار داده است کد مربوط به طور مداوم ارسال می گردد .
- این در حالی است که می خواهیم با زدن کلید ۷ (برای مثال) این عدد تنها برای یکبار روی LCD نمایش داده شود .
-
- (۶-۱)
- توسط زیر روال سرزیر تایمر صفر زمان بندی های لازم برای قسمتهای مختلف برنامه انجام می دهد ، تایمر صفر در TINY26 یک تایمر ۸ بیتی است و تقسیم کلاک نیز ۱۰۲۴ در نظر گرفته شده است بنابراین با توجه به فرکانس 8Mhz به عنوان پالس ساعت سیستم زمان سرریز تایمر صفر چنین بدست می آید :
- (۶-۲)
- فرکانس کلاک تایمر : زمان سرریز تایمر :
- در ابتدای زیرروال وقفه پورت سریال متغیر D1 را با عدد سه و متغیر B2 را با عدد صفر مقدار دهی می کنیم. بلافاصله تایمر صفر ریست می شود و با دستور Start Timer0 شروع به کار می کند .
- به این ترتیب بعد از گذشت زمان ۳۳ میلی ثانیه تایمر تا انتها می شمارد و سرریز اتفاق می افتد .
- در این زمان کنترل برنامه به زیرروال وقفه تایمر می رود. در زیر روال مذکور مقدار D1 کنترل می شود . هرگاه این مقدار برابر صفر شود با توجه به مقدار B2 سه حالت مختلف می تواند رخ دهد. به این ترتیب اگر B۲ برابر صفر باشد (حالت یک) ، LED متصل به PA2 به مدت کوتاهی روشن و سپس خاموش می شود بدین معنی که کاربر بعد از فشار دادن یک دکمه در فرستنده، در حال حاضر دکمه ها را رها کرده است. حال D1 برابر ۳۰ وB2 برابر با ۲ قرار داده می شود و تایمر دوباره راه اندازی می گردد در نتیجه کاربر حدود یک ثانیه فرصت دارد تا برای تکمیل عدد مربوط به انتخاب کانال مورد نظر دکمه بعدی را فشار دهد

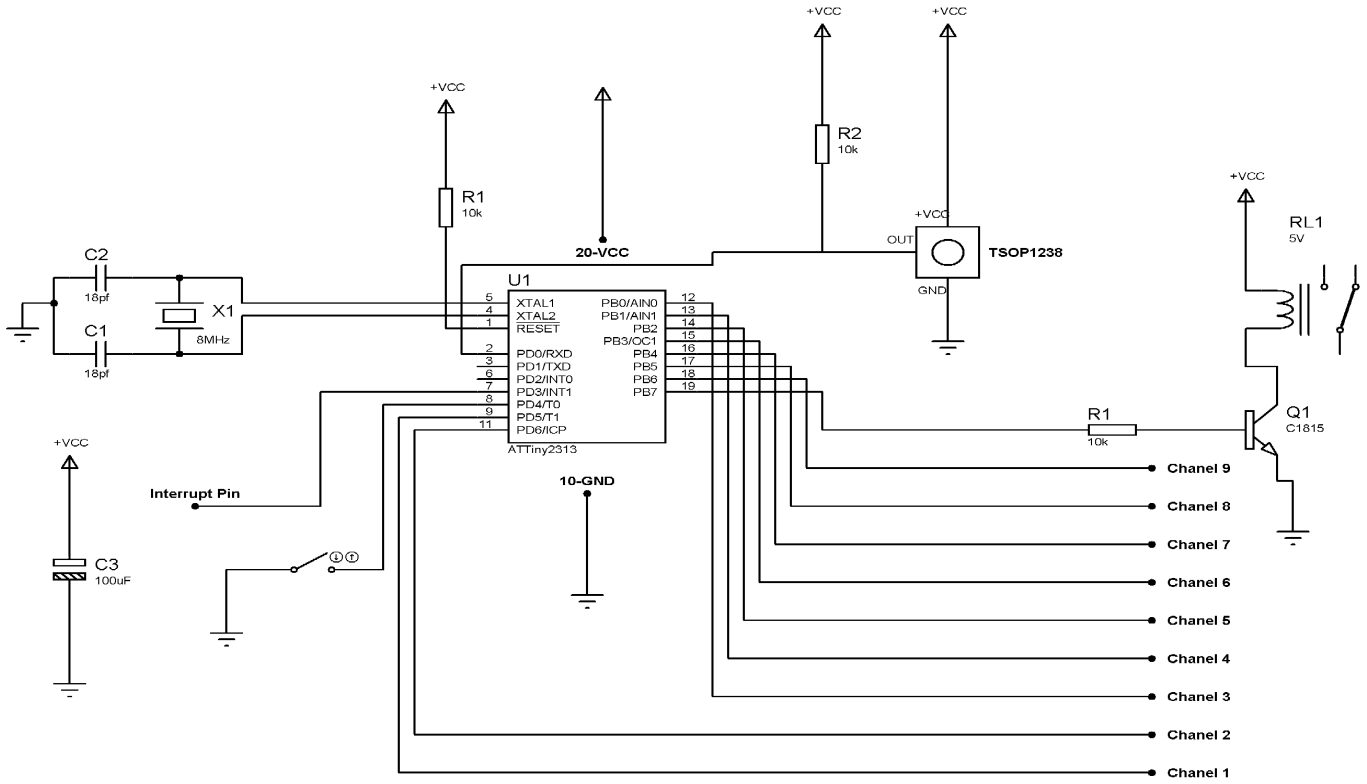
فصل شش(پروگرام لازم جهت راه اندازی AVR):

- شماتیک مدار فرستنده



فصل شش(پروگرام لازم جهت راه اندازی AVR):

- شماتیک مدار گیرنده



طرح جامع

- پس از ارائه مطالب بالا و نشان دادن چند طرح مختلف در این زمینه و پس از رسیدن به برنامه کلی و جامع به این طرح رسیدیم.
- این طرح مجموعه ای متشکل از یک میکرو Atmega32 و یک گیرنده سه پایه و یک فرستنده و مجموعه ای از چند مقاومت و خازن و LED است و همچنین از رگولاتور ۵ ولتی و یک منبع تغذیه بهره برده ایم.
- شرح این مدار به این شکل است که شامل یک کلید است که مدار را روشن و خاموش می کند.
- پس از روشن شدن مدار کلیدی وجود دارد که به اصطلاح قفل مدار محسوب می شود و وقتی فشار داده شود مدار را آماده می سازد که بتواند کد را دریافت کند .
- پس از این که مدار آماده دریافت و ذخیره کد شد یک ریموت و یا فرستنده مادون قرمز کد را به ماژول ما ارسال می نماید و این ماژول این کد را توسط یک گیرنده سه پایه مادون قرمز دریافت کرده و پس از فیلترینگ و مدولاسیون لازم آن را به میکرو ارسال می نماید.
- میکرو کد را به این شکل دریافت می کند و آن را در حافظه EEPROM ذخیره می سازد و در موقع لازم از آن بهره می برد.
- روش بهره بردن و ذخیره سازی این کد توسط میکرو به این شکل است که فرمت کلی آن را مشاهده و با توجه به شکل کد که شامل هیدر و دستور و آدرس است آن را ذخیره می نماید و صفر و یک هایی به وجود می آورد که هر کدام یک شکل پالس ایجاد می نماید

طرح جامع

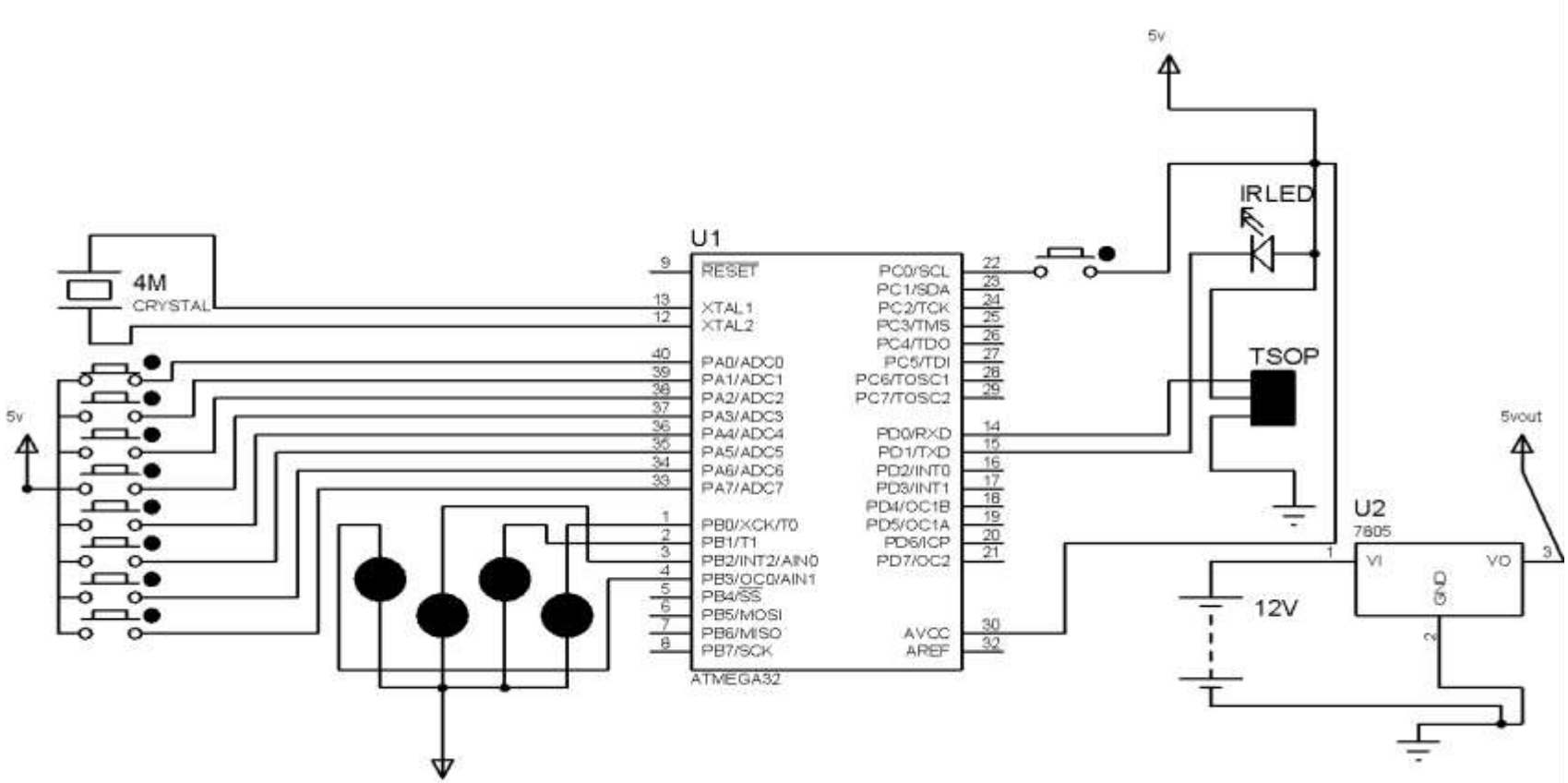
- می توان فرمت کلی کد داده شده را بازسازی کرد و آن را ارسال نمود تا دستگاه ما هم کد ها را ذخیره سازد و هم از این کد ها در موقع لزوم بهره ببرد.
- این کد های دریافتی ابتدا وارد رجیستر شده و سپس وارد حافظه میکرو می شود و آنگاه که بخواهیم از آن بهره ببریم با زدن یک کلید و سپس یک کردن یک رجیستر کد را وارد رجیستر نماییم تا براحتی بتوانیم از آن کلید بهره ببریم و آن را جایگزینی برای کلیدهای ریموت های دیگر محسوب کنیم.
- پس از زدن کلید ها برای اینکه نشان دهیم مدار آماده دریافت کد است یک LED روشن شده و در هنگام دریافت کد LEDدیگری روشن شده و پس از دریافت و ارسال به رجیستر هر دو LED خاموش می شود.
- در هنگام استفاده از مدار و ارسال کد ما از یک فرستنده دو پایه ی مادون قرمز بهره می بریم و این کد از رجیستر منتقل می شود و به صورت پالس های مدوله شده از فرستنده ارسال می شود.
- برای تغذیه مدار از یک رگولاتور ۵ ولت بهره می بریم که ولت لازم برای روشن شدن میکرو و سایر LED ها را برای ما فراهم می کند.
- این رگولاتور باید توسط باتری تغذیه شود .
- این باتری را از نوع باتری های قابل شارژ مجدد انتخاب کردیم تا بتوانیم آن را هر موقع که لازم بود و شارژ آن تمام شد، دوباره آن را شارژ کنیم .
- در هنگامی که شارژ باتری کم شده و در نتیجه جریان مدار پایین آید، یک LED روشن شده و این اخطار را به ما نمایش می دهد و در هنگام شارژ کامل خاموش می شود.
- همچنین برای شارژ این باتری که از نوع باتری های موبایل است ما از یک سوکت شارژ استفاده کردیم و آن را توسط یک
- آداپتور شارژ می نماییم، در هنگام شارژLED دیگری روشن شده و صحت این امر را به ما نمایش می دهد .

طرح جامع

- در هنگام ارسال و دریافت کدها ما از چندین رجیستر استفاده کرده ایم تا بتوانیم هر تعداد کانال که بخواهیم داشته باشیم .
- این رجیسترها دارای اندازه خاصی هستند چرا که هم از حافظه EEPROM استفاده کرده ایم و هم از حافظه دیگر AVR و این امر سبب می شود که کدهای بالای ۲۵۶ بیت در رجیستر اول قرار نگیرد و باید آن را در سایر رجیستر ها قرار داد .
- در ادامه باید اشاره شود این ماژول قادر به دریافت و ارسال اکثر کدها با فرمت های مختلف می باشد و می تواند تعداد ۸ کانال دریافت داشته باشد .
- همچنین از خازن برای گرفتن نویز مدار استفاده کرده ایم تا در هنگام ارسال و دریافت میزان BIT ERROR RATE ما کم شده و کد به سلامت دریافت و ارسال شود .
- از مقاومتها برای PULL UP کردن پایه های میکرو و همچنین LED های ماژول استفاده می کنیم.
- همچنین یک کریستال 4Mhz به کار برده ایم که بتوانیم میزان BUOD لازم را تولید نماییم.

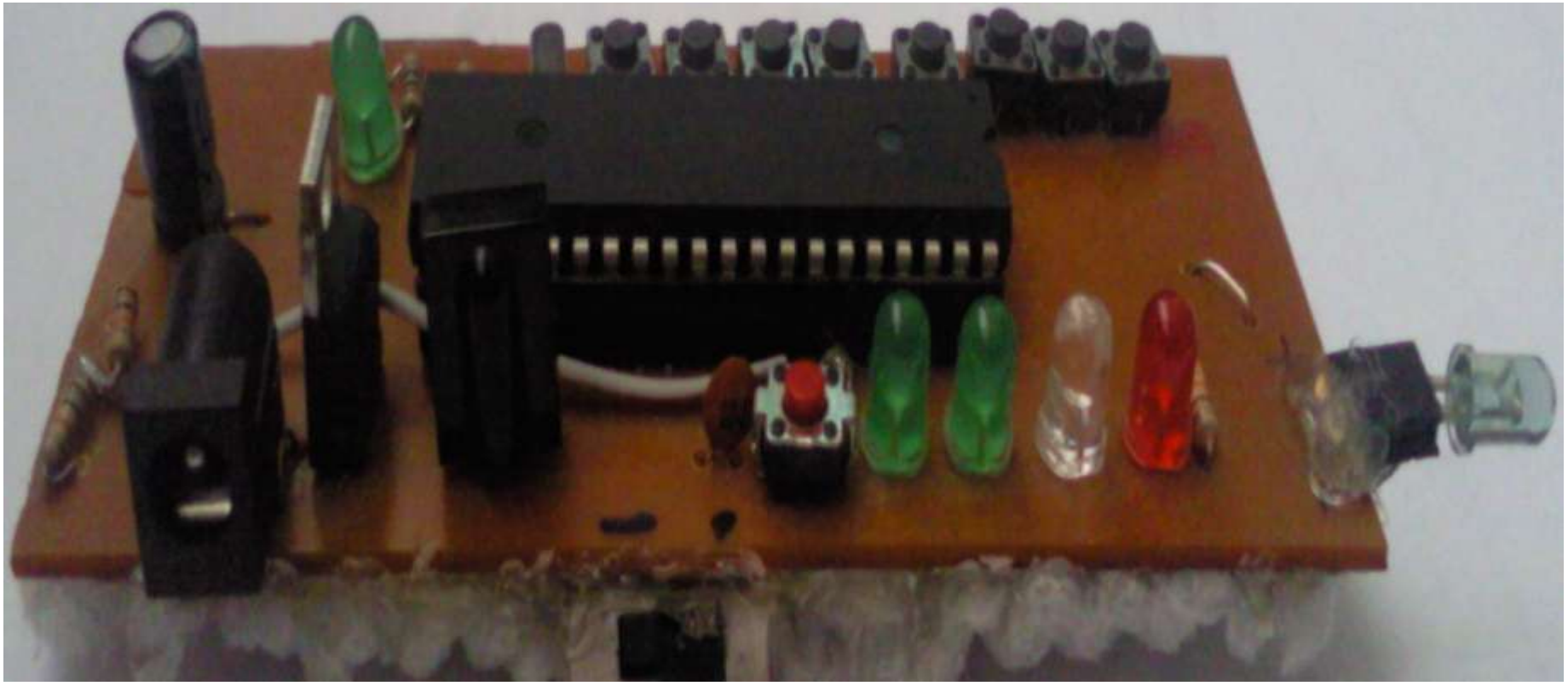
طرح جامع

- شمای کلی ماژول



طرح جامع

- نمای از ماژول ساخته شده



طرح جامع

- **نتایج وتحقیقات آتی**
- پروژه را به سمتی می توان پیش برد که از آن در موارد زیادی بهره برد .
- می توان در پروژه از کدهای ذخیره شده با فرمت خاص بهره برد تا بتوان در مواردی نظیر دزدگیر و ریموتهای کنترلی در لوازم کنترلی استفاده کرد .
- همچنین می توان با تقویت های لازم برد این ریموت را تا مقدار ۱۰۰ متر بهبود بخشید .

ضمیمه

- برنامه ای که نوشتم :

```
$regfile = "M32def.dat“
$crystal = 4000000

Config Lcdpin = Pin , Db4 = Pinb.1 , Db5
= Pinb.2 , Db6 = Pinb.4 , _
 Db7 = Pinb.5 , Rs = Pinb.6 , E = Pinb.7
Config Lcd = 16 * 2
Cursor Off
Cls
Lcd "*  ETRmodern   *"
Waitms 500
```

ضمیمه

- Deflcdchar 1 , 16 , 16 , 28 , 30 , 30 , 28 , 16 , 16
Deflcdchar 2 , 24 , 4 , 18 , 9 , 9 , 18 , 4 , 24
Deflcdchar 3 , 3 , 4 , 9 , 18 , 18 , 9 , 4 , 3
Deflcdchar 4 , 1 , 1 , 7 , 15 , 15 , 7 , 1 , 1

Config Debounce = 10
Config Pina.0 = Input
Config Pina.1 = Input
Config Pina.2 = Input
Config Pina.4 = Input
Config Pina.5 = Input

ضمیمه

```
Config Debounce = 10
Config Pina.0 = Input
Config Pina.1 = Input
Config Pina.2 = Input
Config Pina.4 = Input
Config Pina.5 = Input

Config Pina.3 = Output
Led Alias Porta.3

Config Pind.2 = Input
Set Portd.2
Pin_ir Alias Pind.2
```

ضمیمه

```
Config Pind.1 = Output                    'For IR Puls generator
Puls Alias Portd.1
Reset Puls

Config Timer0 = Timer , Prescale = 256
Stop Timer0
Enable Interrupts
Disable Int0
On Int0 Isr_int0

Config Timer2 = Timer , Prescale = 256
Enable Interrupts
Enable Timer2
```

ضمیمه

Timer2 = 0 •
On Ovf2 Puls_isr
Stop Timer2

Dim Num_level As Word , I As Word , Ad As Word , C As Word , D As Word
Dim Ad2 As Word
Dim Num_code As Byte , N As Byte , T1 As Byte , T2 As Byte , R As Byte
Dim F As Bit , Sw As Byte , G As Byte
Declare Sub R_eeprom
Declare Sub Send

ضمیمه

```
Sw = 0
Reset Led
Stop Timer0
Disable Int0
Cls
Lcd "     " ; Chr(1) ; Chr(2) ; " " ; Chr(3) ; Chr(4) ; "      "
Lowerline
Lcd "Send   & Recieve"
Do
  Debounce Pina.1 , 1 , Get_code
  Debounce Pina.0 , 1 , Remember
Loop
Main:
```

ضمیمه

```
Get_
 Stop Timer0
 F = 0
 I = 501
 Readeeprom Num_code , 5                    'Code number of
saved
 Waitms 4
 If Num_code = 255 Then Num_code = 0
 Incr Num_code
 Writeeeprom Num_code , 5                   'Code number of
saved
 Waitms 4
 Cls
 Lcd "NEW CODE " ; Num_code ; "    "code:
```

ضمیمه

```
'***  •
 Enable Int0
Enable Interupt 0
 Start Timer0
 Do
   Debounce Pina.1 , 0 , Rw_eeprom
 Loop
```

ضمیمه

```
'isr for infrared sensor
Isr_int0:
 Stop Timer0

 If F = 1 Then
   T2 = Timer0
   Out I , T2
   F = 0
   Incr I
 End If
```

ضمیمه

```
Timer0 = 0
Start Timer0

Bitwait Pin_ir , Set
Stop Timer0
T1 = Timer0

Timer0 = 0
Start Timer0

If F = 0 Then
  Out I , T1
  F = 1
  Incr I
End If

Toggle Led
Return
```

ضمیمه

```
Rw_eeprom:
 Disable Int0
 Stop Timer0
 Reset Led
 Cls : Lcd "Please Wait....."
 Lowerline : Lcd "Write in EEPROM "

 If Num_code = 1 Then
   D = 10
 Else
   Readeeprom D , 10                              'last home
number
   Waitms 4
```

ضمیمه

```
If D > 900 Then  •
    Cls : Lcd "Memory is Full  "
    Wait 2
    jmp Main
  End If

 End If
```

ضمیمه

```
Num_level = I - 501
C = D + 5
D = C + Num_level
Writeeeprom D , 10                    'last home number
Waitms 4

I = 501
For Ad = C To D
  R = Inp(i)
  Writeeeprom R , Ad
  Waitms 4
  Incr I
Next Ad
```

ضمیمه

```
C = C - 2    •
Writeeeprom Num_level , C
Waitms 4

Remember:
Reset Led
Readeeprom Num_code , 5                              'Code number of
saved
Waitms 4
If Num_code = 255 Then
  Cls : Lcd "NOT EXIST CODE  "
  Wait 1
  jmp Main
End If
Cls : Lcd "FOR SHOW LEVELS " : Lowerline : Lcd "PRESS KEY  ....."
```

ضمیمه

```
I = 0
N = 0
Sw = 0
Do
  Reset Porta.3
  Debounce Pina.1 , 1 , Select_code
  Debounce Pina.2 , 1 , Main
Loop

Select_code:
 Sw = 1
 Reset Led
 If N = 0 Then
   Ad = 8
```

ضمیمه

```
  Num_level = 0    •
 End If

 Incr N
 Ad = Ad + 5
 Ad = Ad + Num_level
 Ad2 = Ad + 2
 Readeeprom Num_level , Ad
 Waitms 4

 If Num_level > 1500 Then                    'usually Bits number
are 100 Bit
   Cls : Lcd "NOT EXIST       " : Lowerline : Lcd "      AFTER CODE"
```

ضمیمه

```
If N = 1 Then
  Ad = 0
Else
  Ad = Ad - 3
End If

Writeeeprom Ad , 10
Waitms 4
Decr N
Writeeeprom N , 5
Wait 1
N = 0
If Ad = 0 Then Jmp Main
jmp select_code
End If
```

ضمیمه

```
  Home : Lcd "Code Number " ; N ; "    "
  Locate 2 , 1 : Lcd "Level NUM :" ; Num_level ; "    "

Key:
  Reset Led
  Reset Puls
  Do
    Debounce Pina.0 , 1 , R_eeprom , Sub
    Debounce Pina.1 , 1 , Select_code
    Debounce Pina.2 , 1 , Main
    Debounce Pina.4 , 1 , Send , Sub
    Debounce Pina.5 , 1 , Delet
Loop
```

ضمیمه

- Puls_generat:

```
 F = 0
Start Timer2
Do
   If F = 1 Then
     Stop Timer2
     Reset Puls
     Goto Key
   End If
Loop
```

ضمیمه

```
Puls_isr:
  Stop Timer2
  Toggle Puls
  Incr I
  C = I - 500
  If C = D Then
    F = 1
  End If
  R = Inp(i)
  R = 256 - R
  Timer2 = R
  Start Timer2
Return
```

ضمیمه

```
Delet:
  Num_code = 255
  Writeeeprom Num_code , 5
  Waitms 4
  Cls : Lcd "DELET MEMORY ..."
  Wait 1
jmp main


End                                          'end program
```

ضمیمه

```
Sub R_eeprom
  Sw = 0
  Reset Led
  Home : Lcd "CODE" ; N ; " :" ; Num_level ; " Level"
  If I > Num_level Then
    Home : Lcd "Was Final LEVEL "
    Wait 1
    jmp Main
  End If
  Readeeprom R , Ad2
  Waitms 4
  Locate 2 , 1 : Lcd "Time P" ; Ad2 ; " = " ; R ; "    "
  Incr Ad2
  Incr I
End Sub R_eeprom
```

ضمیمه

```
Sub Send      •
  D = Ad2 + Num_level
  Select Case Sw
    Case 0:
      Jmp Select_code
    Case 1:
      Reset Led
      Cls : Lcd "CODE : " ; N ; "         "
      Lowerline : Lcd "Please wait ...."
      I = 500
      For Ad = Ad2 To D
        Incr I
        Readeeprom R , Ad
        Waitms 4
        Out I , R
      Next Ad
      Sw = 2
      Cls : Lcd "Press Key ......"
    Case 2:
      I = 500
      jmp puls_generat
  End Select

End Sub Send
      •
```