توجه :

ویرایش قبلی این مقاله قبلا در مجله ی PMM (مجله میکروکنترلر فارسی – Persian Microcontroller Magazine) شماره 10 و 11 منتشر شده بود. ویرایشی که همکنون مشاهده میکنید برای کتاب " مرجع کامل میکروکنترلرهای سری AT91SAM شرکت ATMEL" تهیه شده است .

برای کسب اطلاعات بیشتر د ر مورد این کتاب پست شماره 11 تاپیک زیر را مطالعه کنید :

http://www.iranmicro.ir/forum/showthread.php?t=12189

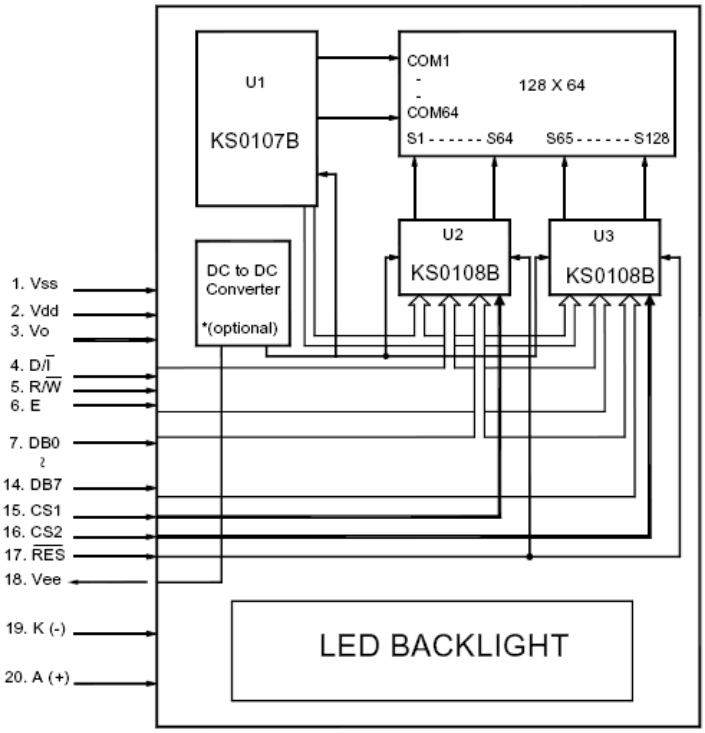
جهت ارتباط با نویسنده به آدرس زیر مراجعه کنید :

http://www.iranmicro.ir/forum/member.php?u=3948

راه اندازی LCD گرافیکی AE12864E (مبتنی بر تراشه ی KS0108)

LCD های گرافیکی کاربرد گسترده ای در دستگاه های الکترونیکی دارند ، ما از این نمایشگرها را برای ایجاد ارتباط بصری میان کاربر و سخت افزار استفاده میکنیم ، کاربر میتواند اطلاعات خروجی دستگاه را بر روی این نوع LCD مشاهده کرده و داده ی خود را توسط تاچ پد یا صفحه کلید یا ... به دستگاه وارد نماید .

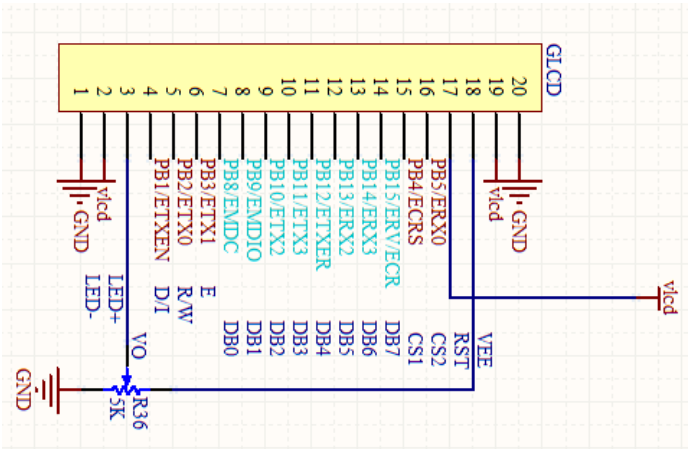
LCD گرافیکی که معمولا با نام LCD128*64 یا AE12864E یا AGM1264F در بازار موجود می باشد ، دارای دوکنترل کننده به نام KS0108B و یک کنترل کننده به نام KS0107B میباشد .کنترل کننده ی KS0108 میتواند 64 ستون از یک DOT MATRIX کریستال مایع را آدرس دهی و کنترل نماید . تراشه ی KS0107B وظیفه ی رفرش کردن سطر ها را بر عهده دارد. در زیر بلوک دیاگرام این نوع LCD را مشاهده میفرمایید :



در واقع LCD گرافیکی از دو DOT MATRIX که هر کدام دارای 64*64 نقطه میباشند تشکیل شده است ، با یک کردن پایه ی CS1 چیپ KS0108 وDOT MATRIX اول و با یک کردن پایه ی CS1 چیپ KS0108 وDOT MATRIX دوم انتخاب میشود . شما میتوانید با ارسال کد های که در ادامه آورده میشود از طریق خطوط D/I (RS) و R/W و EN (E) چیپ های موجود را کنترل کرده و داده ی خود را از طریق خطوط DB0 تا DB7 به آنها ارسال کنید و بر روی LCD نمایش دهید . در جدول زیر نام و کاربرد پایه های LCD مذکور آورده شده است :

شماره	نام پایه		کاربرد	محل اتصال
1	Vss	Ground	تغذیه ی منفی LCD	گراند مدار
2	Vdd	VCC	تغذیه ی مثبت LCD	ولتاژ 5 ولت
3	Vo	LCD contrast adjust	تنظیم کنتراست LCD	نقشه را ببینید
4	D/I	Data input	انتخاب نوع داده ی ورودی (کد کنترلری یا نمایشی)	پایه ای از میکرو
5	R/W	Data read	انتخاب نوع کار (خواندن یا نوشتن)	پایه ای از میکرو
6	E	Enable signal	فعال سازی LCD	پایه ای از میکرو
7	DB0	Data bit 0	پایه ی انتقال داده	پایه ای از میکرو
8	DB1	Data bit 1	پایه ی انتقال داده	پایه ای از میکرو
9	DB2	Data bit 2	پایه ی انتقال داده	پایه ای از میکرو
10	DB3	Data bit 3	پایه ی انتقال داده	پایه ای از میکرو
11	DB4	Data bit 4	پایه ی انتقال داده	پایه ای از میکرو
12	DB5	Data bit 5	پایه ی انتقال داده	پایه ای از میکرو
13	DB6	Data bit 6	پایه ی انتقال داده	پایه ای از میکرو
14	DB7	Data bit 7	پایه ی انتقال داده	پایه ای از میکرو
15	CS1	Chip selection for IC1	انتخاب KS0108 و DOT MATRIX اول	پایه ای از میکرو
16	CS2	Chip selection for IC2	انتخاب KS0108 و DOT MATRIX دوم	پایه ای از میکرو
17	RST	Reset	بازنشانی LCD	ولتاژ تغذیه
18	Vee	Power supply for LCD driving	خروجی ولتاژ منفی	نقشه را ببینید
19	BL-	Power Supply for BACKLIGHT	تغذیه ی منفی LED پشت LCD	گراند مدار
20	BL+	Power Supply for BACKLIGHT	تغذیه ی مثبت LED پشت LCD	ولتاژ تغذیه ی مثبت

برای اتصال lcd به میکرو به شماتیکی مطابق تصویر زیر نیاز خواهید داشت :



در زیر وضعیت پایه ها و عمل کرد کد های ارسالی به lcd را مشاهده میکنید :

Instruction	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	Function
Display on/off	L	L	L	L	H	H	H	H	H	L/H	خاموش یا روشن کردن کنترل کننده ی lcd ، در حالت خاموش داده در ram ذخیره نمیشود . L: OFF, H: ON
Set address (Y address)	L	L	L	H	Y address (0 - 63)						تنظیم کردن آدرس y در شمارنده ی y نمایشگر
Set page (X address)	L	L	H	L	H	H	H	Page (0 - 7)			تنظیم آدرس x در رجیستر x نمایشگر
Display start line (Z address)	L	L	H	H	Display start line (0 - 63)						نمایش داده ی موجود در ram نمایشگر در بالای صفحه
Status read	L	H	Busy	L	On / Off	Reset	L	L	L	L	خواندن وضعیت lcd BUSY : L: Ready / H: In operation ON/OFF : L: Display ON / H: Display OFF RESET : L: Normal / H: Reset
Write display data	H	L	Write data								نوشتن داده در حافظه ی ram نمایشگر
Read display data	H	H	Read data								خواندن داده ی موجود در حافظه ی ram نمایشگر

- همانگونه که قبلا نیز اشاره کردیم lcd های گرافیکی از دو نمایشگر مجزای 64*64 پیکسل تشکیل یشوند ، برای نمایش داده در هر بخش شما باید بعد از یک کردن cs مربوطه داده را مطابق جدول بالا به کنترل کننده ارسال نمایید .
- برای پاک کردن lcd باید کلیه پیکسل ها را با ارسال فضای خالی خاموش کنید .
- برای نمایش اشکال هندسی باید ، شکل را به پیکسل های تشکیل دهنده تجزیه کرده و سپس پیکسل ها را به صورت تک تک به lcd ارسال کنید .
- از آدرس میتوانید دیتاشیت تراشه ی KS0108 را دانلود نمایید :

http://www.alldatasheet.com/datasheet-pdf/pdf/37323/SAMSUNG/KS0108B.html

با استفاده از کتابخانه ی KS0108.h میتوانید با دستورات ساده و بدون نیاز به دانستن نحوه ی کار lcd ، با نمایشگر کار کنید ، همچنین با مطالعه ی این کتابخانه میتوانید اطلاعات بیشتر در مورد نحوه ی راه اندازی lcd بدست آورديد.

بررسی کتابخانه ی KS0108.h :

با فراخوانی این کتابخانه در برنامه ی خود ، میتوانید با استفاده از دستوراتی که در ادامه آورده شده است ، lcd گرافیکی را راه اندازی نمایید ، برای استفاده از این کتاب خانه باید آن را در مسیر زیر یا پوشه ی پروژه ی خود ، کپی کنید : (کتابخانه در پوشه ی پیوست مجله موجود میباشد)

Program Files\Keil\ARM\INC\Atmel.

(در پوشه های SAM7X و SAM7A3 و AT91SAM9G20 و سایر پوشه های موجود).

برای استفاده از این کتابخانه ، ابتدا باید آن را در برنامه فراخوانی کنید ، برای اینکار از دستور زیر استفاده می شود :

```
#include < ks0108.h>
```

با فراخوانی کتابخانه ، مجموعه دستورات زیر به نرم افزار keil اضافه میگردد :

```
#define GLCD_DATAPORT_x
#define KS0108_D0          y
#define KS0108_RS          y
#define KS0108_RW          y
#define KS0108_EN          y
#define KS0108_CS1         y
#define KS0108_CS2         y
```

در این دستورات x میتواند A به مفهموم پورت A و B به مفهوم پورت B باشد . همچنین Y شماره ی پایه های پورت X میباشد که به پایه های LCD گرافیکی متصل شده اند .

- در صورتی که کتابخانه ی PIO.H را به برنامه ی خود فراخوانی کنید میتوانید به جای ارقام از Pxy استفاده کنید (پروژه ی موجود در فایل پیسوت را مشاهده کنید)

دستور GLCD_Initialize:

```
GLCD_Initialize();
```

با این دستور میکرو کنترلر lcd گرافیکی را پیکربندی نموده و آن را برای دریافت دستورات آماده میکند .

دستور GLCD_WriteString:

```
GLCD_WriteString("String");
```

با این دستور میتواتنید یک رشته را به lcd ارسال کنید ، مثال :

```
GLCD_WriteString("|     Farzad     |");
```

دستور GLCD_GoTo :

```
GLCD_GoTo(x,y);
```

با این دستور مکان نمای lcd نمایش را از سطر y و ستون x شروع میکند . مثال :

```
GLCD_GoTo(0,4);
GLCD_WriteString("|   KS0108 Library  |");
```

دستور GLCD_ClearScreen :

```
GLCD_ClearScreen();
```

با اجرا شدن این دستور صفحه ی نمایش lcd پاک میشود .

دستور GLCD_Rectangle :

```
GLCD_Rectangle(x, y, w, L);
```

با این دستور میتوانید یک مربع یا مستطیل را در مختصات x و y و با طول L و عرضw ایجاد کنید . مثال :

```
GLCD_Rectangle(0, 0, 51, 63);
```

دستور GLCD_Line :

```
GLCD_Line(x1,y1,x2,y2);
```

توسط این دستور میتوانید خطی از مختصات x1,y1 تا مختصات x2,y2 رسم نمایید . مثال :

```
GLCD_Line(0,9,42,63);
```

دستور GLCD_Circle :

```
GLCD_Circle(x, y ,R);
```

با این دستور میتوانید دایره ای به شعاع R در مختصات x,y رسم کنید (مرکز دایره x,y میباشد) . مثال :

```
GLCD_Circle(96, 32 ,20);
```

دستور GLCD_WriteChar :

```
GLCD_WriteChar(Char);
```

توسط این دستور میتوانید یک متغیر از نوع Char را بر روی lcd نمایش دهید . مثال :

```
char i=23;
GLCD_WriteChar(i+48);
```

دستور GLCD_SetPixel :

```
GLCD_SetPixel(x , y, color);
```

توسط این دستور میتوانید یک پیکسل به مختصات x,y از صفحه ی نمایش را روشن (color=1) یا خاموش (color=0) کنید . مثال :

```
GLCD_SetPixel(5,6, 1);
```

پروژه :

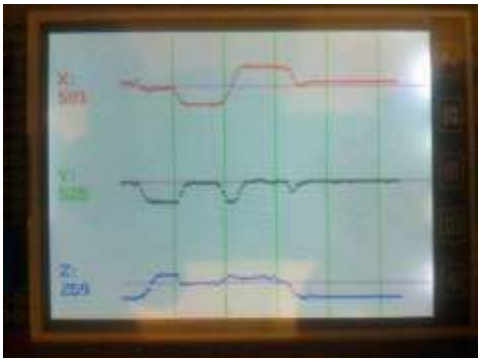
```
//-------------------------------------------------------------------------------------------- --------
// KS0108 driver library  for AT91sam7S/X microcontrollers
// www.farzadsw.ir
//--------------------------------------------------------------------------------------------
#include "AT91SAM7X256.h"
#include "delay.h"
 #define GLCD_DATAPORT_B
 #define KS0108_D0                    8
#define KS0108_RS          1
#define KS0108_RW          2
#define KS0108_EN          3
#define KS0108_CS1         4
#define KS0108_CS2         5
//#define KS0108_CS3  x
#include "ks0108.h"
int main(void)
{
 char i,x;
            *AT91C_PMC_PCER=0x0000000F;
            *AT91C_PIOA_PER=0x0000000F;
            *AT91C_PIOA_OER=0x0000000F;
```

```
GLCD_Initialize();
GLCD_ClearScreen();
GLCD_GoTo(0,0);
GLCD_WriteString("+------------------+");
GLCD_GoTo(0,1);
GLCD_WriteString("|      Farzad      |");
GLCD_GoTo(0,2);
GLCD_WriteString("|    Ahmadinezhad   |");
GLCD_GoTo(0,3);
GLCD_WriteString("|                  |");
GLCD_GoTo(0,4);
GLCD_WriteString("|   KS0108 Library  |");
GLCD_GoTo(0,5);
GLCD_WriteString("|   For AT91sam7    |");
GLCD_GoTo(0,6);
GLCD_WriteString("|  Microcontrollers |");
GLCD_GoTo(0,7);
GLCD_WriteString("+------------------+");
delay_s(3) ;
GLCD_ClearScreen();
for(i=0 ; i<8 ; i++)
 {
  GLCD_GoTo(6*i ,i);
  GLCD_WriteChar(i+48);
 }
delay_s(3) ;
GLCD_Rectangle(0, 0, 51, 63);
GLCD_Line(0,9,42,63);
GLCD_Line(6,0,49,54);
GLCD_Circle(96, 32 ,20);
delay_s(3) ;
```

```
GLCD_ClearScreen();
while(1){
  for(x=0 ; x<8 ; x++) {
    GLCD_GoTo(6*x ,i);
    GLCD_WriteChar(i+48);
    i++;
    }
}
 return 0;
}
```

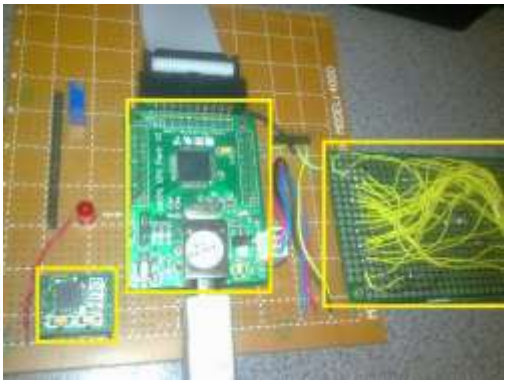
راه اندازی سنسور شتاب 3 بعدی mma7260 و اتصال آن به at9asam7s/x

در این پروژه از یک سنسور شتاب 3 بعدی به اسم MMA7260 ، ساخت شرکت Freescale ، استفاده شده است . این سنسور شتاب دارای 3 خروجی ولتاژ آنالوگ ، برای 3 محور x,y,z می‌باشد . از ویژگی های این سنسور میتوان به قیمت مناسب ، دقت قابل قبول و مقاومت بالا در برابر شوک اشاره کرد (تا 5000 g).



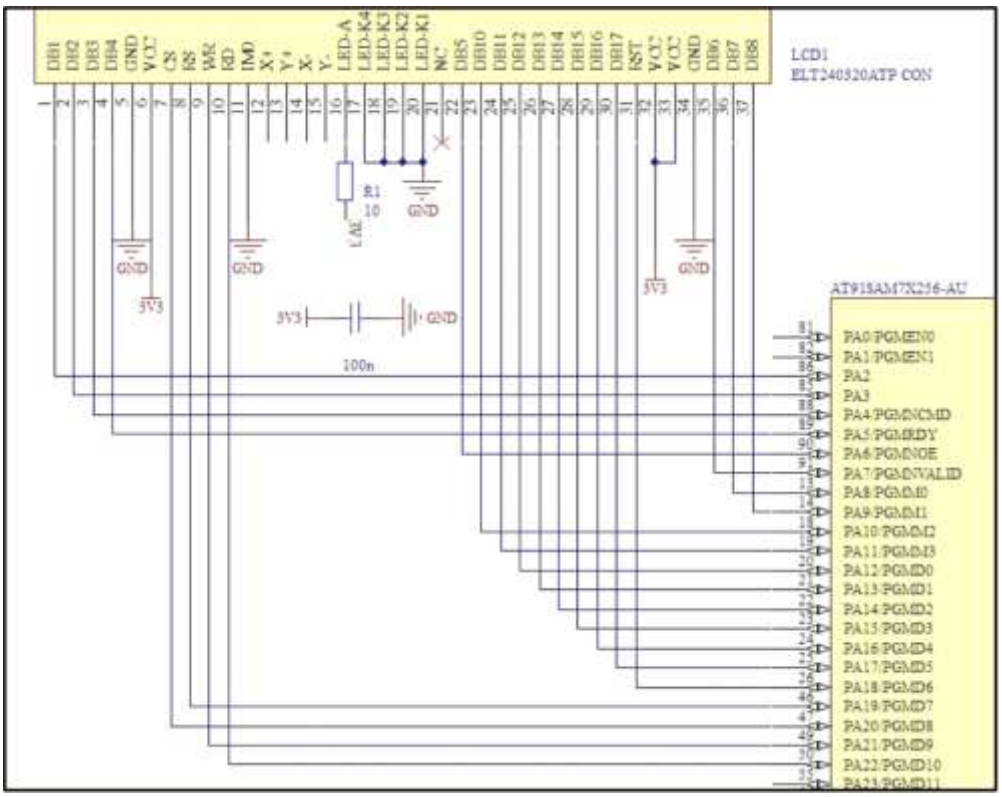
میکروکنترلر استفاده شده در این پروژه AT91SAM7S64 ، از سری S میکروکنترلرهای آرم شرکت اتمل است . سری S کاملا با سری X همخوانی داشته و تمامی کد های نوشته شده برای آن در میکروکنترلر های سری X بدون نیاز به هیچ تغییری ، قابل استفاده می‌باشند .

برای نمایش اطلاعات سنسور ها ، از یک LCD رنگی با رزولیشن Qvga ویا 320*240 که در بخش سوم بررسی شده است ، استفاده گردیده است . با توجه به نیاز های پروژه ، چند تابع به بخش هدر این نمایشگر اضافه شده ،که در ادامه مطلب تشریح می گردند.



نحوه اتصال LCD به میکرو کنترلر :

نحوه اتصال LCD به میکرو کنترلر به طور کلی مطابق شکل زیر می‌باشد:

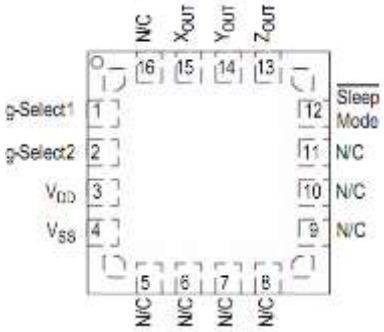


با این تفاوت که دیتا باس نمایشگر ، به جای PA2 از PA0 شروع می‌شود(مقدار آفست برابر 0 است) . و نیز پایه های کنترلی rst,rs,cs,wr,rd به ترتیب به پایه های 20–16 پورت A میکروکنترلر اتصال دارند. با توجه به این نحوه اتصال و اینکه LCD به صورت خوابیده (Landscape) تصاویر را نمایش می‌دهد ، در ابتدای برنامه اصلی تنظیمات به صورت زیر اعمال می‌شوند:

```
#define LANDSCAPE
/* LCD Pin Configuration */
#define TFTLCD_DATAPORT_A
#define TFTLCD_DATAPORT_OFFSET      0
#define TFTLCD_CONTROLPORT_A
#define TFTLCD_RST                  16
#define TFTLCD_RS                   18
#define TFTLCD_CS                   17
#define TFTLCD_WR                   19
#define TFTLCD_RD                   20
```

نحوه اتصال سنسور شتاب به میکرو کنترلر :

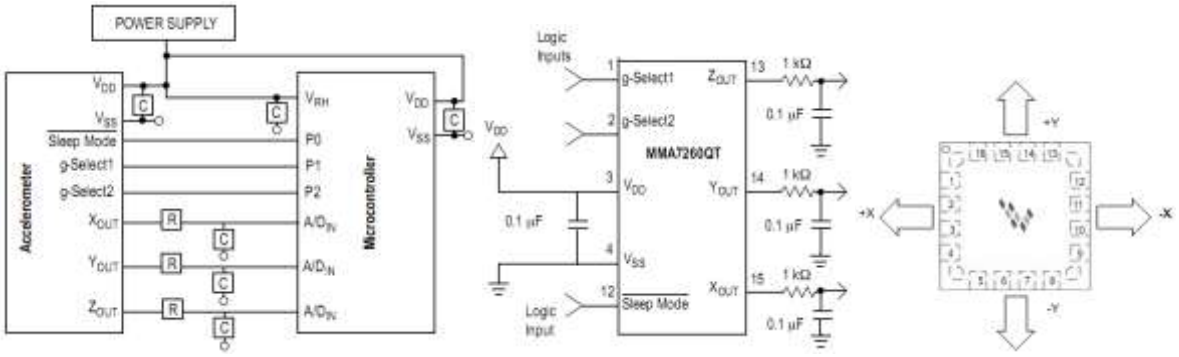
این سنسور علاوه بر سه پایه خروجی (که به ADC4,5,6 میکروکنترلر متصل شده اند) و 2 پایه تغذیه ، دارای 3 پایه کنترلی می‌باشد:



پایه Sleep mode برای قرار دادن سنسور در حالت غیر فعال و کم شدن مصرف انرژی آن استفاده می‌شود. در این پروژه با توجه به اینکه سنسور باید در تمامی مواقع فعال باشد ، این پایه به 1 منطقی متصل شده تا همیشه فعال باشد.

پایه های g-select1,2 برای انتخاب مقیاس و یا حساسیت سنسور استفاده میشوند . به این صورت که 0و1 بودن هر کدام از این پایه ها یکی از مقیاس های 1.5 ، 3 ، 4.5 ، 6 g را تعیین میکند. با توجه به اینکه مقیاس 1.5 g دارای بیشترین دقت بوده (800 mv/g) ، پایه های g-select1,2 هردو به زمین یا صفر منطقی متصل شده اند تا سنسور در مقیاس 1.5 g کار کند.

جهت محور های x,y,z ، نحوه بایاسینگ سنسور و همچنین نحوه اتصال آن به میکروکنترلر ، در شکل های زیر قابل مشاهده می‌باشند :



نرم افزار :

هدر adc.h :

از این هدر برای خواندن مقادیر آنالوگ سنسور و تبدیل آن به دیجتال ، استفاده می‌شود . این هدر از 2 تابع تشکیل شده است :

تابع config_adc(char adc) : این تابع برای تنظیم رجیستر های ADC میکرو کنترلر (فعال بودن یا نبودن وقفه ، prescaler و...) و نیز انتخاب کانالهای مورد نظر و فعال شدن آنها استفاده می‌شود .با توجه به اینکه در این پروژه از ADC4,5,6 برای خواندن مقادیر سنسور استفاده شده است ، از این تابع به صورت config_adc(0x70) استفاده شده است (70 هگز معادل فعال بودن کانالهای 4و5و6 است) .

تابع read_adc(char channel) : این تابع کانال adc مورد نظر را به عنوان آرگومان دریافت و مقدار خوانده شده را به صورت یک هدد صحیح برمی‌گرداند .

برای کسب اطلاعات بیشتر در مورد ADC مطالب بخش پنجم را بخوانید .

هدر functions.h :

این هدر برای راه اندازی و استفاده از LCD رنگی ، مورد استفاده قرار گرفته است . توابع و مشخصات این هدر در بخش سوم بررسی شده اند ، بنابراین از توضیح در مورد آنها خودداری می گردد .

برای نمایش اطلاعات بر روی LCD تنها یک تابع putsf در فایل هدر تعریف شده است . به دلیل اینکه آرگومان این تابع ، یک متغیر یا داده از نوع کارکتر است ، امکان استفاده از این تابع برای نمایش اعداد وجود ندارد . به عنوان مثالچنانچه یک متغیر از نوع int که دارای مقدار عددی 16 است را به این تابع بفرستیم ، عبارتی که در LCD نمایش داده خواهد شد ، کاراکتر اسلش ('/') خواهد بود ، نه عدد مورد نظر ما .

بنابراین برای نمایش اعداد در این LCD نیاز به یک تابع خواهد بود که عدد مورد نظر را به یک رشته از کاراکتر های متناظر ، تبدیل کند تا قابل نمایش بر روی LCD باشد . این کار با استفاده از تابع int2str میسر است . به دلیل

موجود نبودن کتابخانه عملیات بر روی رشته ها ، در نرم افزار keil . این تابع در همان هدر functions.h تعریف گردیده است :

```
void int2str(int n ,char *str)
   {

    str[0] = n/10000 ;
    n = n - str[0] * 10000 ;
    str[0] +=48;

    str[1] = n/1000  ;
    n = n - str[1] * 1000 ;
    str[1] +=48;

    str[2] = n/100  ;
    n = n - str[2] *100 ;
    str[2] +=48;

    str[3] = n/10  ;
    n = n - str[3] *10 ;
    str[3] +=48;

    str[4] = n+48;


   }
```

این تابع یک عدد صحیح و یک آرایه کاراکتری (رشته) را به عنوان ورودی دریافت می کند و رقم های این عدد صحیح را در خانه های آرایه کارکتری قرار می‌دهد . همانطور که مشاهده می‌شود این تابع دارای 2 اشکال می‌باشد :

قابل بسط دادن به متغیر های دیگر مثلا long نمی‌باشد . (به سادگی)

حاصل تبدیل همیشه یک رشته 5 کاراکتری می‌باشد . به عنوان مثال 123 به 00123 تبدیل می‌شود .

برای رفع این اشکالات یک تابع دیگر به نام intstr تعریف گردیده است ،که در پروژه نیز از آن استفاده شده است(تابع قبلی صرفا جهت آشنایی با روند کار معرفی گردید) . این تابع نیاز به استفاده از 2 تابع دیگر ، یکی برای محاسبه توان و یکی برای بدست آوردن تعداد ارقام ، دارد . این 3 تابع به صورت زیر تعریف گردیده اند :

```
long pow(int b , int p)
    {
     long result=1;
     for(;p>0;p--)result *=b;
     return result;
    }
```

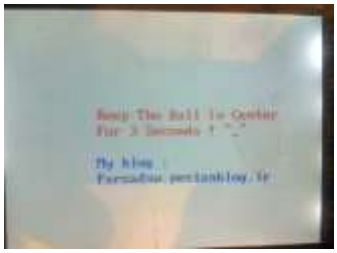
```
int digit(int n)
    {
     int i,j;
     for (i=4 ;i>0 ;i--)
        {
         j =  n/pow(10,i);
         if ( j != 0 )break;

         }
     return i+1;
    }
```

```
void intstr(int n ,char *str)
    {

     int i,d;
     d= digit(n)-1;
     for (i = 0 ; i< d ; i++)
         {
            str[i] = n / pow(10,d-i);
             n = n - str[i] * pow(10,d-i) ;
             str[i] +=48;
          }
     str[d] = n+48;

     for (i = d+1 ; i< 5 ; i++)
         {
             str[i] = 0;
          }

     }
```

این تابع نیز دارای عملکرد مشابهی با int2str میباشد ، با این تفاوت که با توجه به تعداد ارقام عمل تبدیل را انجام می‌دهد (متغییر d) و بقیه رشته را خالی می کند. همچنین به دلیل استفاده از حلقه ، میتوان به سادگی این تابع را برای اعداد بزرگتر از int نیز بسط داد .

برنامه اصلی :

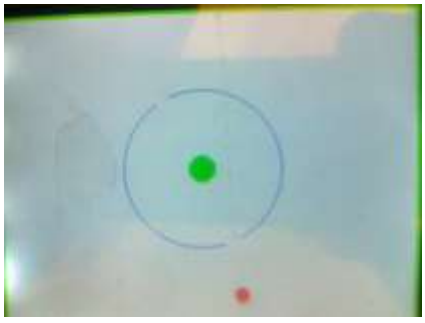
1. این برنامه از سه بخش اصلی تشکیل شده است ، ابتدا یک متن نمایش داده می‌شود .



```
tftlcd_gotoxy(11,6);
tftlcd_putsf("Keep The Ball In Center",RED,WHITE,1);

tftlcd_gotoxy(11,7);
tftlcd_putsf("For 3 Seconds ! ^_^",RED,WHITE,1);


tftlcd_gotoxy(11,9);
tftlcd_putsf("My blog :",BLUE,WHITE,1);

tftlcd_gotoxy(11,10);
tftlcd_putsf("Farzadsw.pesianblog.ir",BLUE,WHITE,1);


delay_ms(6000);
tftlcd_clear();
```

2. سپس یک توپ قرمز در صفحه نشان داده می‌شود که با توجه به شیب و جهت سنسور در صفحه حرکت می‌کند. هرگاه این توپ به مدت 3 ثانیه در مرکز صفحه(دایره سبز) نگه داشته شود ، قسمت بعدی برنامه اجرا می‌شود .

این قسمت از برنامه درحقیقت یک بازی ساده مشابه بازی هایی که در موبالهای جدید(که دارای سنسور شتاب هستند) می‌باشد. البته چون هدف ساخت یک بازی کامپیوتری نبوده ، سعی شده تا حد امکان ساده طراحی شود و هدف اصلی نشان دادن کارایی و قدرت سنسور شتاب برای ساخت یک کنترل دستی بوده است . در این قسمت از برنامه تنها از دومحور x,y استفاده شده . بنابراین با استفاده از سنسور های شتاب 2 بعدی نیز می‌توان این عملیات را انجام داد.



در ادامه ، کد این قسمت از برنامه آورده شده است و توضیحاتی در خصوص آن ارائه می‌گردد.

```
while(!ok)
{
tftlcd_draw_circle(bx,by,5,1,RED);
start_adc();
x=read_adc(4);
y=read_adc(5);
delay_ms(40);
tftlcd_draw_circle(bx,by,5,1,WHITE);
by += x/10 - 50 ;
bx += y/10 - 50 ;
if(by > 230)by -= 220;
if(by <10 )by += 220;
if(bx > 310)bx -= 300;
if(bx <10 )bx += 300;
if(bx <176 && bx>144 && by<136 && by>104)
    {
     tftlcd_draw_circle(160,120,10,1,GREEN);

     if(bx <170 && bx>150&& by<130 && by>110)
      {
        cun +=2;
        tftlcd_draw_circle(160,120,10,1,BLUE);
      }
    }

cun -- ;
if(cun<0)cun=0;

if(cun>40) ok=1;
}
```

کلیت کد به این صورت است که ابتدا برنامه داخل یک حلقه قرار می‌گیرد و تا وقتی که متغیر ok برابر 1 نشود داخل آن باقی می‌ماند. در ابتدا یک دایره به شعاع 5 به رنگ قرمز ایجاد می‌شود ، سپس مقادیر سنسور ها خوانده می‌شود و یک تاخیر کوچک اعمال می‌شود . با توجه به مقادیر سنسور ها مکان توپ در جهت x,y توسط متغیر bx,by تغییر می‌کنند(هرچه شیب بیشتر باشد میزان تغییرات بیشتر است) و در دور بعدی حلقه ، توپ قرمز در مکان جدید ترسیم می‌شود. با توجه به مقادیر bx,by تشخیص داده میشود که آیا توپ در مرکز صفحه و روی دایره سبز رنگ قرار دارد یا خیر . اگر توپ مدت معینی داخل محدوده مورد نظر باشد ok برابر 1 می‌شود و از حلقه خارج می‌شود و قسمت بعدی برنامه اجرا می‌شود.

در ادامه به جزئیات موجود در کد و پیشنهاداتی برای جالب تر شدن این برنامه اشاره می‌گردد:

- در هر دور (از حلقه) وقتی دایره قرمز ترسیم می‌شود ، یک تاخیر بر حسب میلی ثانیه ایجاد و سپس یک دایره سفید با همان ابعاد در همان محل دایره قرمز رسم می‌شود . هدف از این کار پاک کردن دایره قرمز

است . زیرا اگر این کار انجام نشود ، توپ به صورت متحرک دیده نمی‌شود ، بلکه مشاهده می‌شود که یک منحنی قرمز رنگ از رد توپ ایجاد شده است (مشابه حالتی که با نرم افزار های گرافیکی و ابزار brush یک خط یا منحنی ترسیم می‌شود) . البته خود این حالت می‌تواند در مواردی مفید و مطلوب باشد . حتی می‌توان با استفاده از یک رنگ دیگر به جای سفید ، یک دنباله رنگی برای توپ ایجاد کرد (مشابه ستاره دنباله دار)

- مقدار تاخیر ، فریم ریت را تعیین می‌کند . با مقدار 40 ، حدودا 15 فریم در ثانیه ایجاد می‌شود . علت این امر این است که تابع تاخیر ایجاد شده به هیچ عنوان دقیق نبوده و صرفا جهت ایجاد تاخیر استفاده شده است . با کم کردن این مقدار می‌توان فریمت ریت بالاتر و درنتیجه انیمیشنی روانتر ایجاد نمود . این برنامه با مقدار تاخیر 10 نیز امتحان شده است . دراین حالت فریم ریت تقریبا برابر 60 و حرکت توپ بسیار روان و طبیعی تر از حالت قبل به نمایش در آمد .
- نکته بعدی تقسیم مقدار خوانده شده از سنسور ها بر عدد 10 به منظور صرف نظر کردن از تغییرات کوچک سنسور و ایجاد یک سرعت مناسب برای جابه جایی توپ است.
- به وسیله 2 دستور شرطی بررسی می‌شود که آیا توپ از محدوده صفحه خارج شده است ؟ اگر این اتفاق افتاده باشد ، توپ را در طرف دیگر آن دیواره نمایش می‌دهیم . به عنوان مثال اگر موقعیت توپ در جهت y به 240 برسد ، موقعیت توپ در جهت y به مقدار 20 تغییر می‌کند . به این ترتیب توپ همیشه در صفحه وجود خواهد داشت و از آن خارج نمی‌شود. با توجه به اینکه یک حاشیه سبز رنگ در اطراف صفحه نمایش به عنوان دیوار ترسیم شده است ، برای توسعه این برنامه می‌توانید برنامه رو طوری تغییر بدهید که هنگام رسیدن توپ به دیوار ، به جای انتقال آن به دیوار رو برو ، به دیوار برخورد کرده و بازتاب کند . **راهنمایی** : برای ایجاد چنین حالتی باید به جای تغییر دادن مکان توپ بر حسب مقادیر سنسور ، سرعت توپ را تغییر بدهید و هنگام برخورد با دیوار ، سرعت توپ قرینه شود .
- قبل از رسیدن به حلقه این برنامه ، یک دایره آبی توخالی به شعاع بزرگ و یک دایره سبز توپر به شعاع کمتر در صفحه رسم شده است . بنابر این وقتی در داخل حلقه توپ از روی این دایره ها رد می‌شود این طور به نظر می‌رسد که آن محل گذر توپ ،پاک شده است(به دلیل رسم دایره سفید در آن محل) . برای جلو گیری از پاک شدن دایره سبز رنگ ، هنگام نزدیک شدن و رسیدن توپ به آن(توسط شرط بررسی

محدوده) ، این دایره دوباره ترسیم می‌شود. برای جالب تر شدن ، برنامه طوری نوشته شده است که وقتی توپ بر روی دایره سبز قرار گرفت ، رنگ دایره سبز به آبی تغییر می کند .

- در حال حاضر lcd دارای یک لایه گرافیکی است و علت پاک شدن تصویر پس زمینه با عبور توپ از آن نیز همین موضوع است . برای توسعه این بخش و ایجاد یک کتابخانه گرافیکی کارا ، باید برای صفحه نمایش ، چند لایه در نظر گرفت . شما می‌توانید به سادگی یک محیط گرافیکی 2 لایه ایجاد کنید . بدیهی است که این لایه های گرافیکی به صورت نرم افزاری ایجاد می‌شوند . **راهنمایی** : لایه گرافیکی به این صورت تعیین میشوند که برای هر تصویر در لایه بالاتر باید یک بافر حافظه در نظر گرفت که در آن وضعیت پیکسلهای لایه پایینی آن ذخیره می گردند. به عنوان مثال در این برنامه تنها کافیست که توسط یک بافر ، پیکسلهایی که توپ در فریم بعدی بر روی آنها قرار می گیرند را ذخیره ، و بعد از عبور توپ از آن ، آنها را باز گردانی کنید . کشیدن دایره سفید در محل قبلی توپ نیز حالت خاصی از این عمل است.
- با قرار گرفتن توپ در داخل محدوده مشخص شده به مقدار متغیر cun دو واحد اضافه می‌شود ولی در هر مرحله از حلقه یک واحد از آن کاسته می گردد . علت این روش برنامه نویسی این است که اگر توپ در ناحیه مرکزی نباشد ، اطمینان حاصل شود که مقدار cun بعد از چند مرحله صفر شود و عدد ثابتی باقی نماند . در حقیقت این اگوریتم یک Gauge را شبیه سازی می کند.
- برای کاربردی شدن و جالب تر شدن همین بازی ، می‌توانید از یک تصویر به جای یک دایره ساده به عنوان توپ استفاده کنید و یا وقتی توپ به محدوده مرکزی می‌رسد ، یک موتور ویبره ، با چرخش و ایجاد لرزش ، برد را بلرزاند و به بازی هیجان بیشتری بدهد !

3. در این قسمت از برنامه ، نمودار مقادیر هر 3 محور سنسور بر حسب زمان نمایش داده می‌شود . این عمل برای آشنایی با سنسور و کالیبره کردن آن بسیار مفید است زیرا می‌توانید مقادیر محور های سنسور را در لحظات قبل ببینید و اطلاعات ثبت می‌شوند و یکباره از بین نمی‌روند .

همانطور که در قطعه کد زیر مشاهده می‌کنید ، ابتدا مقادیر سنسورها خوانده می‌شوند و سپس توسط تابع intstr که قبلا توضیحات آن داده شد ، این مقادیر سنسور به یک رشته کارکتری تبدیل می‌شوند و در نمایشگر نشان داده می‌شوند. متغییر amud مقدار سنسور برای هر محور را به صورت مقیاس شده در خود نگه می‌دارد . متغیر cun نیز حکم زمان را دارد و در هر دور از حلقه ، افزایش می‌یابد. در نتیجه amud معرف مولفه ی عمودی نمودار یا y و cun معرف مولفه افقی نمودار یا x میباشد . با توجه به این مقادیر یک نقطه بر روی صفحه ترسیم می‌شود . مجموعه این نقاط ، یک نمودار را تشکیل می‌دهند.

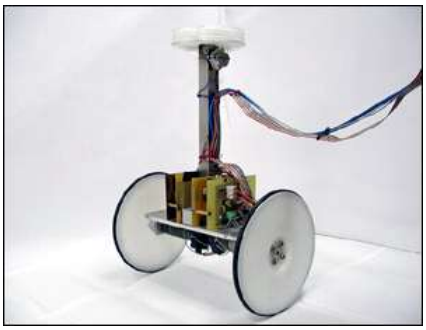
```
while(1)
    {

     start_adc();
     x=read_adc(4);
     y=read_adc(5);
     z=read_adc(6);

    intstr(x,str);
    tftlcd_gotoxy(1,2);
    tftlcd_putsf("X:" ,RED,WHITE,1);
    tftlcd_gotoxy(1,3);
    tftlcd_putsf(str ,RED,WHITE,1);
    amud = 80 - x/15;
    if(amud <0 || amud >237) amud = 0;

    tftlcd_draw_circle(cun , amud ,1,1, RED)  ;

    intstr(y,str);
    tftlcd_gotoxy(1,7);
    tftlcd_putsf("Y:",GREEN,WHITE,1);
    tftlcd_gotoxy(1,8);
    tftlcd_putsf(str,GREEN,WHITE,1);
    amud = 160 - y/15;
    if(amud <0 || amud >237) amud = 0;

    tftlcd_draw_circle(cun , amud ,1,1, BLACK)  ;

    intstr(z,str);
    tftlcd_gotoxy(1,12);
    tftlcd_putsf("Z:",BLUE,WHITE,1);
    tftlcd_gotoxy(1,13);
    tftlcd_putsf(str,BLUE,WHITE,1);
    amud = 239 - z/15;
    if(amud <0 || amud >237) amud = 0;

    tftlcd_draw_circle(cun , amud ,1,1, BLUE)  ;

    cun =cun+2;
```

در انتهای کد بالا یک دستور شرطی قرار داده می‌شود تا وقتی نمودار به انتهای نمایشگر رسید ، نمودار را مجددا از ابتدا ترسیم نماید ، چند خط کد نیز برای ترسیم محور های کمکی (grid) استفاده شده است . کد حاصل به صورت زیر است:

```
if(cun>318){
 cun=60;
 tftlcd_clear();
 tftlcd_draw_line(60,45,318,45,PURPLE);
 tftlcd_draw_line(60,125,318,125,PURPLE);
 tftlcd_draw_line(60,210,318,210,PURPLE);
 tftlcd_draw_line(103,1,103,238,GREEN);
 tftlcd_draw_line(146,1,146,238,GREEN);
 tftlcd_draw_line(189,1,189,238,GREEN);
 tftlcd_draw_line(232,1,232,238,GREEN);
 tftlcd_draw_line(275,1,275,238,GREEN);
    }
```

نتیجه گیری :

همانطور که در این مثال دیده شد ، از سنسور شتاب میتوان در ساخت دسته کنترلر استفاده کرد . محور های x, y مشابه قسمت دوم برنامه ، میتواند راستا و جهت حرکت یک ماشین یا ربات و نیز میزان سرعت حرکت آنرا تعیین کند . با استفاده از 3 محور این سنسور می‌توان برای اجسامی که حرکت 3بعدی دارند (مثل هواپیما) یک کنترلر ساخت . علاوه بر ساخت کنترلر ، از این سنسور در ساخت ربات هایی که به تعادل نیاز دارند ، استفاده زیادی می‌شود . روند کار مشابه قسمت دوم برنامه است ، با این تفاوت که نگه داشتن توپ در مرکز باید به طور اتوماتیک و توسط حرکت موتورها انجام گردد.



مواردی که در اینجا مطرح شد ،پیرامون استفاده از خاصیت سنجش شیب (tilt) این سنسور و آن هم به واسطه وجود شتاب ثقل g ، بوده است . اما علاوه بر آن ، با انتگرال گیری از مقادیر سنسور می‌توان سرعت نسبی جسم متحرک را نیز به طور حدودی بدست آورد . اما به دلیل وجود خطا (و جمع شدن این خطا ها هنگام انتگرال گیری) ، از این سنسور به تنهایی برای اندازی گیری سرعت و موقعیت جسم استفاده نمی‌شود .