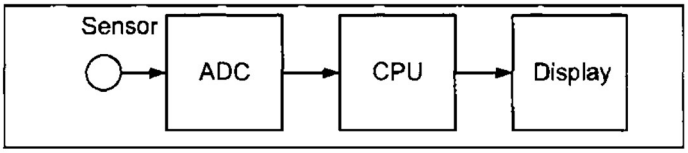
7 آشنایی با مبدل‌های آنالوگ به دیجیتال و دیجیتال به آنالوگ

برای اندازه گیری کمیت‌های فیزیکی مانند دما شدت صدا و شدت نور و ....از سنسورهای با خروجی آنالوگ در کنار مبدل‌های آنالوگ به دیجیتال[1] استفاده می‌شود.



شکل 7-1: اتصال یک ریزپردازنده به سنسور با استفاده از مبدل ADC

کمیت‌های فیزیکی با استفاده از سنسورها -به عنوان بخشی از مبدل یا ترنسدیوسر- به کمیت‌های الکتریکی تبدیل می‌شود سپس کمیت الکتریکی آنالوگ توسط مبدل آنالوگ به دیجیتال به داده‌های دیجیتال تبدیل شده و پردازنده می‌تواند داده‌ها را مورد بررسی قرار دهد و به عنوان مثال نتایج را روی نمایشگرها ارسال نماید.

در این بخش، نحوه‌ی تبدیل سیگنال آنالوگ به دیجیتال، استفاده از مبدل ADC داخلی ریزپردازنده، خواندن اطلاعات از سنسورهای آنالوگ و کار با مبدل دیجیتال به آنالوگ بررسی می‌شود.

7.1 سنسورهای برد آموزشی

در این مجموعه آموزشی سنسورهای ذیل در در بلوکی با عنوان Sensors قرار داده شده است.

- دو عدد سنسور دما (LM35 وNTC)
- یک عدد سنسور نور (CDS)
- یک عدد سنسور رطوبت (HIS06)
- یک عدد سنسور گاز شهری (TGS813)
- یک ولوم 10 کیلو اهم جهت شبیه سازی سنسور مولفه‌های محیطی دلخواه

7.2 برخی از خصوصیات اصلی ADC

یکی از اصلی‌ترین ویژگی‌های مبدل A/D دقت یا وضوح می‌باشد. دقت مبدل A/D بر حسب تعداد بیت بیان می‌شود که در ریزپردازنده Atmega16، 10 یا 8 بیت می‌باشد.

حداقل ولتاژی که برای A/D قابل شناسایی است را step size می‌نامند. هرچه تعداد بیت‌های مبدل دیجیتال بیشتر باشد، step size کمتر است.

[1] Analog-to-digital converter

$$step\ size = \frac{V_{ref}}{2^n}$$

V_{ref} یک ولتاژ مرجع است که A/D را قادر می‌سازد تا سیگنال‌های آنالوگ در محدوده‌ی بین 0 تا Vref را مانند شکل 7-2 اندازه بگیرد. V_{ref} می‌تواند مقادیر AVCC یعنی 5 ولت یا ولتاژ مرجع 2.56 ولت داخلی و یا پایه‌ی خارجی AREF را اتخاذ نماید.

در ریزپردازنده برای کاهش اثرات نویز از خط تغذیه(AVCC) و زمین (AGND) جداگانه‌ای استفاده می‌شود که AVCC نباید بیش از ±0.3V نسبت به VCC اختلاف داشته باشد.

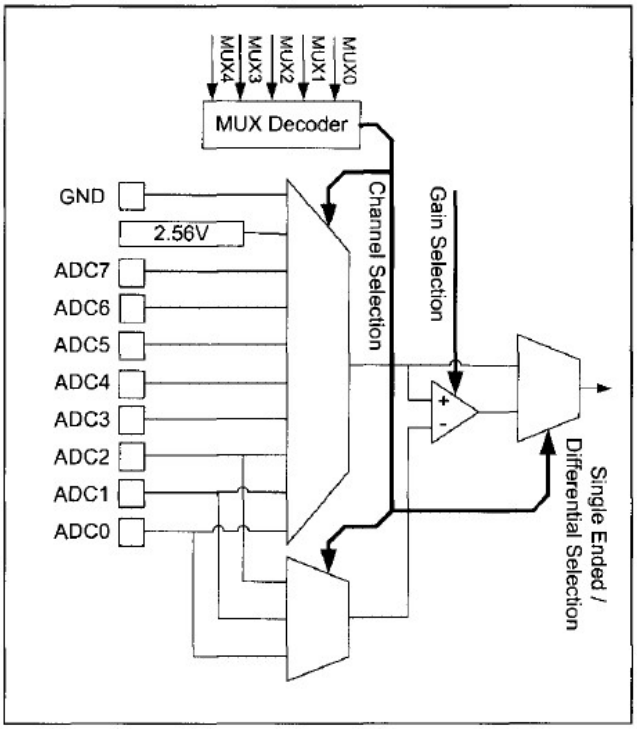
V_{ref} (V)	V_{in} Range (V)	Step Size (mV)
5.00	0 to 5	5/256 = 19.53
4.0	0 to 4	4/256 = 15.62
3.0	0 to 3	3/256 = 11.71
2.56	0 to 2.56	2.56/256 = 10
2.0	0 to 2	2/256 = 7.81
1.28	0 to 1.28	1.28/256 = 5
1	0 to 1	1/256 = 3.90

شکل 7-2 ارتباط میان ولتاژ مرجع، ولتاژ ورودی و step

حداکثر ولتاژ قابل اندازه‌گیری برابر با(Vref (=VCC است. در غیر این صورت، مبدل آنالوگ به دیجیتال آسیب می‌بیند. کمترین ولتاژ اعمالی نیز برابر با GND است. ADC به ازای ولتاژ 5 ولت در حالت 10 بیتی عدد 1023 (و در حالت 8 بیتی عدد 255) و به ازای صفر ولت عدد صفر را در ثبات مربوطه قرار می‌دهد.

در Atmega16 می‌توان 8 سیگنال آنالوگ ورودی را به دیجیتال مانند شکل 7-3 تبدیل کرد. هر یک از کانال‌های ورودی از طریق مالتی پلکسر انتخاب می‌شوند و پس از اعمال بهره مناسب، نمونه برداری شده و به صورت داده‌های دیجیتال ۸ بیتی و یا ۱۰ بیتی در ثبات مربوطه قرار می‌گیرد.

$$D_{out} = \frac{V_i}{V_{ref}} * 2^n$$



شکل 7-3 کانال‌های ورودی ADC - (درگاه A)

7.3 ثبات‌های مبدل آنالوگ به دیجیتال

7.3.1 ثبات کنترلی ADMUX

تنظیمات اولیه مبدل آنالوگ به دیجیتال در ثبات کنترلی **ADMUX** انجام می‌شود. این تنظیمات شامل انتخاب ولتاژ مرجع، ۸ یا ۱۰ بیتی بودن مبدل، انتخاب کانال ورودی و تنظیم حالت‌های عملکردی می‌باشد.

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

شکل 7-4- ثبات کنترلی ADMUX

REFS0,1: از این دو بیت برای انتخاب ولتاژ مرجع ADC استفاده می‌کنیم که دارای چهار حالت می‌باشد:

جدول 7-1: تعیین ولتاژهای مرجع مبدل آنالوگ به دیجیتال

REFS1	REFS0	Vref
0	0	AREF
0	1	AVCC
1	0	-
1	1	2.56

انتخاب ولتاژ مرجع دقیق در تبدیل کردن آنالوگ به دیجیتال نقش بسیار مهمی‌دارد. دقیق‌ترین ولتاژ مرجع همان 2.56 داخلی می‌باشد. البته می‌توان از تثبیت کننده‌های ولتاژ برای تولید ولتاژ مرجع دلخواه استفاده نمود و و به پایه‌ی AREF متصل نمود.

ADLAR: از این بیت برای تنظیم نحوه پر شدن ثبات ADC از چپ یا راست استفاده می‌شود. لذا اگر صفر باشد از چپ پرشده و اگر یک باشد از راست پر می‌شود.

MUX0-4: مقدار این بیت‌ها، مانند شکل 7-5 ترکیب ورودی‌های آنالوگ را برای مبدل تعیین می‌نماید. همچنین به ترکیب‌های تفاضلی بهره اختصاص می‌دهد.

$$D_{out} = A * \left(\frac{V_i^+ - V_i^-}{V_{ref}} \right) * 2^n$$

برای استفاده از حالت تفاضلی بایدPositive Differential Input را به ولتاژ ورودی آنالوگ و Negative Differential Input را به زمین وصل کنید. ترکیب تفاضلی فقط در پکیج‌های TQFP و MLF وجود دارد.

شکل 7-5 : تنظیمات قابل اعمال برای ADC با تغییر پارامترهای MUX4-0 در ثبات ADMUX

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			
01000	N/A	ADC0	ADC0	10x
01001		ADC1	ADC0	10x
01010		ADC0	ADC0	200x
01011		ADC1	ADC0	200x
01100		ADC2	ADC2	10x
01101		ADC3	ADC2	10x
01110		ADC2	ADC2	200x
01111		ADC3	ADC2	200x
10000		ADC0	ADC1	1x
10001		ADC1	ADC1	1x
10010		ADC2	ADC1	1x
10011		ADC3	ADC1	1x
10100		ADC4	ADC1	1x
10101		ADC5	ADC1	1x
10110		ADC6	ADC1	1x
10111		ADC7	ADC1	1x
11000		ADC0	ADC2	1x
11001		ADC1	ADC2	1x
11010		ADC2	ADC2	1x
11011		ADC3	ADC2	1x
11100		ADC4	ADC2	1x
11101		ADC5	ADC2	1x
11110	1.22V (V_{BG})	N/A		
11111	0V (GND)			

7.3.2 ثبات ADCSRA

ثبات وضعیت و کنترلی مبدل نحوه فعال شدن مبدل را تنظیم می‌نماید.

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

شکل 7-6- ثبات ADCSRA

ADEN: با یک کردن این بیت ADC فعال می‌شود.

ADSC: با نوشتن یک در این بیت، تبدیل شروع می‌شود.

ADATE: با یک کردن این بیت A/D می‌تواند به صورت اتوماتیک با لبه بالا رونده منبع تحریک کننده شروع به تبدیل کند. منبع تحریک توسط بیت‌های ADTS از رجیستر SFIOR انتخاب می شود.

ADIF: بعد از اتمام تبدیل یا به روز شدن ثبات داده ADC، برابر با یک می‌شود و به عبارتی یک شدن این بیت نشانه‌ی معتبر بودن داده‌های ثباتADC برای خوانده شدن است.

ADIE: با یک کردن این بیت پس از اتمام تبدیل، وقفه‌ای صادر می‌شود و در زیر روال وقفه، داده ثبات ADC خوانده می‌شود.

ADPS0-3: از این بیت‌ها مانند جدول 7-2 برای تعیین ضریب تقسیم پالس ساعت ریزپردازنده برای بخش مبدل آنالوگ به دیجیتال استفاده می‌شود.

جدول 7-2: جدول تنظیمات تقسیم پالس ساعت

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

7.3.3 ثبات داده ADC (ADCH,ADCL)

این ثبات شانزده بیتی، حاوی داده‌ی خروجی مبدل است و بنا به مقدار ADLAR، از چپ یا راست پر می‌گردد. در نتیجه در حالت عملکرد 8 بیتی مقدار ADCH خوانده می‌شود و در حالت عملکرد 10 بیتی ثبات ADC خوانده می‌شود.

7.3.4 ثبات SFIOR[1]

این ثبات که در شکل 7-7 نشان داده شده است، منبع تحریک مبدل و عملکرد سرعت بالا تنظیم می‌شود.

7	6	5	4	3	2	1	0	
ADTS2	ADTS1	ADTS0	ADHSM	ACME	PUD	PSR2	PSR10	SFIOR
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

شکل 7-7- ثبات SFIOR

[1]Special Function IO Register

از طریق بیت‌های ADTS0-2 می‌توان منبع تحریک مبدل برای شروع تبدیل را مانند جدول 7-3 تنظیم نمود.

جدول 7-3تعیین منابع تحریک ADC

منبع تحریک ADC	ADTS0	ADTS1	ADTS2
مدعملکردآزاد	0	0	0
مقایسه کننده آنالوگ	1	0	0
وقفه خارجی صفر	0	1	0
وقفه‌ی (Compare Match) تایمر صفر	1	1	0
سرریز تایمر صفر	0	0	1
وقفه‌ی Compare Match B	1	0	1
سرریز تایمر یک	0	1	1
ضبط رخداد تایمر یک	1	1	1

بیت ADHSM: با فعال شدن این بیت نمونه برداری مبدل با سرعت بیشتر و با مصرف انرژی بیشتر انجام می‌شود.

7.4 معرفی وقفه ADC

برای جلوگیری از اتلاف زمان ریزپردازنده، به جای بررسی مکرر بیت ADSC در ثبات ADCSRA بهتر است از وقفه استفاده شود (بیت ADIE از ثبات ADCSRA برابر با یک باشد). در این صورت به محض کامل شدن تبدیل، پرچم ADIF یک می‌شود و CPU به محل اجرای وقفه پرش کرده و از نتیجه ADC استفاده می‌نماید.

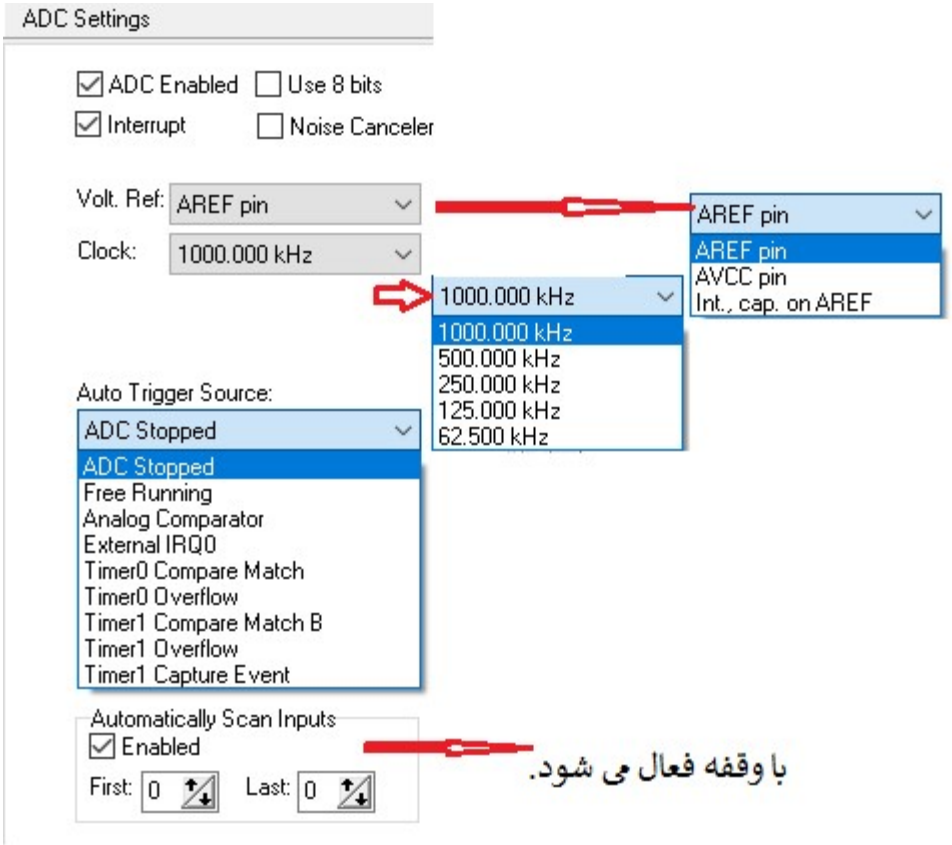
7.5 مراحل برنامه‌نویسی ADC

مراحل برنامه‌نویسی مبدل آنالوگ به دیجیتال در شکل 7-8 نشان داده شده است. بدین منظور لازم است پایه ورودی سیگنال آنالوگ تعریف شود و از منوی CodeWizard سایر پارامترها نیز تنظیم شود که به ترتیب شامل موارد ذیل می‌باشد.

1. فعال کردن ماژول ADC
2. انتخاب 10 یا هشت بیتی بودن تبدیل: در حالت کلی Atmega16 به صورت ده بیتی نمونه برداری می نماید ولی با انتخاب هشت بیتی فقط 8 بیت با ارزش‌تر را ارائه می‌دهد.
3. حذف نمودن نویز
4. تعیین کردن سرعت تبدیل: انتخاب سرعت تبدیل به فرکانس سیگنال آنالوگ ورودی بستگی دارد. اگر فرکانس بالا باشد بهتر است از نرخ های بالاتر برای تبدیل استفاده نمود و اگر تغییرات ورودی به کندی

انجام می‌شود می‌توان از نرخ‌های پایین‌تر تبدیل استفاده نمود تا مصرف انرژی ریزپردازنده کمتر شود و نویز کمتری وارد کل مدار به سبب ADC گردد که این مساله در مدارهای فرکانس بالا اهمیت خود را بیشتر نشان می‌دهد.

5. انتخاب ولتاژ مرجع : در مبدل های A/D برای بهبود دقت اندازه‌گیری از ولتاژ مرجع دقیق استفاده می‌شود که می توان در صورت نیاز با استفاده از تراشه‌های مربوطه این ولتاژ را تهیه نمود.
6. انتخاب منبع تحریک مبدل: در حالت عادی می‌توان از حالت free running استفاده کرد، همچنین امکان فعال نمودن وقفه در بازه زمانی مشخص هم مانند استفاده از وقفه‌های تایمر وجود دارد.
7. اگر وقفه فعال شده باشد می‌توان به طور خودکار تعدادی از کانال‌های ADC را اسکن نمود.



شکل 7-8 :برنامه نویسی مبدل آنالوگ به دیجیتال با CodeWizard

7.5.1 برنامه خواندن داده‌های مبدل دیجیتال

برای خواندن داده‌های مبدل A/D در حالت بدون وقفه از زیربرنامه ی read_adc مانند برنامه 7-1 می‌توان استفاده کرد. چنانچه وقفه فعال شده باشد، با روتین وقفه‌ی برنامه 7-2 مقدار هر یک از ورودی‌ها خوانده شده و در متغیری ذخیره می‌گردد. در این برنامه A/D به صورت ده بیتی فعال بوده و کل ثبات ADCW را می‌خواند. در صورتی که در حالت هشت بیتی فقط ADCH خوانده می‌شود.

برنامه 7-1

```
unsigned int read_adc(unsigned char adc_input)
{
ADMUX=adc_input | ADC_VREF_TYPE;
 //Delay needed for the stabilization of the ADC input voltage
delay_us(10);
 //Start the AD conversion
ADCSRA|=(1<<ADSC);
 //Wait for the AD conversion to complete
while ((ADCSRA & (1<<ADIF))==0);
ADCSRA|=(1<<ADIF);
return ADCW;
}
```

برنامه 7-2

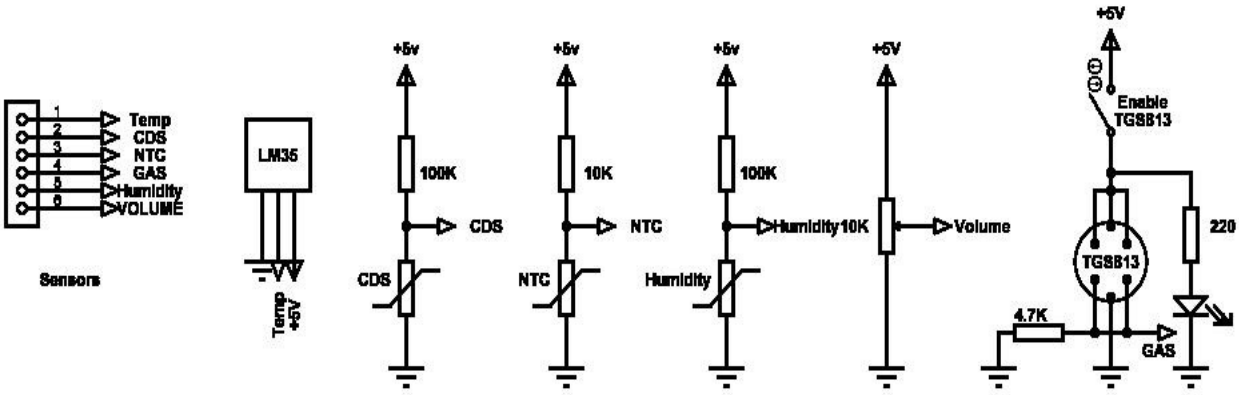
```
interrupt [ADC_INT] void adc_isr(void)
{
static unsigned char input_index=0;
// Read the AD conversion result
adc_data[input_index]=ADCW;
// Select next ADC input
if (++input_index > (LAST_ADC_INPUT-FIRST_ADC_INPUT))
    input_index=0;
ADMUX=(FIRST_ADC_INPUT | ADC_VREF_TYPE)+input_index;
// Delay needed for the stabilization of the ADC input voltage
delay_us(10);
// Start the AD conversion
ADCSRA|=(1<<ADSC);
}
```

7.6 معرفی چند سنسور

سنسورهای CDS، NTC و HIS06 به ترتیب سنسورهای دما، نور و رطوبت می‌باشند. این سه نوع سنسور با تغییر پارامترهای محیط تغییر مقاومت می‌دهند. از این رو به منظور ارتباط با ریزپردازنده طبق مدارهای شکل 7-9 تغییرات مقاومت به تغییرات ولتاژ تبدیل شده است.

سنسور TGS813 سنسور آشکارساز گاز شهری با خروجی ولتاژ می‌باشد که مستقیما به ریزپردازنده اعمال می‌شود. به دلیل افزایش حرارت بدنه سنسور و همچنین جریان کشی بالای این سنسور از منبع تغذیه مطابق شماتیک شکل

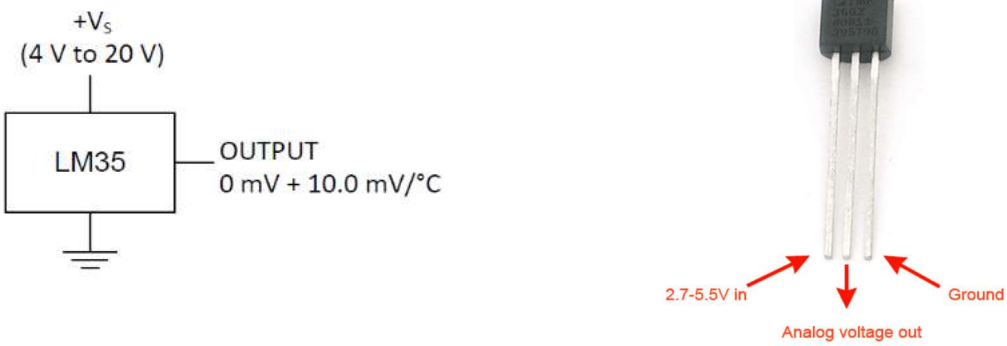
9-7 یک کلید کشویی برای قطع و وصل نمودن این سنسور در مسیر تغذیه سنسور تعبیه شده است. LED موجود در این بلوک نشانگر وصل یا قطع بودن این سنسور در مدار است. همچنین یک عدد ولوم 10 کیلو اهم به منظور تولید ولتاژ از سطح صفر ولت تا 5 ولت برای شبیه‌سازی سنسورهای پارامترهای محیط در این بلوک قرار داده شده است.



شکل 7-9- شماتیک مربوط به کلیه‌ی سنسورها بر روی برد آموزشی

7.7 اندازه‌گیری دما

برای اندازه‌گیری دما از سنسور دیگری به نام LM35 مانند شکل 7-10 نیز استفاده می‌شود. این سنسور دمای بین -55 تا 150 درجه‌ی سانتیگراد را اندازه می‌گیرد و به ازای هر 1 ± سانتی‌گراد ±10 mV ولتاژ خروجی را تغییر می‌دهد. یعنی به ازای دمای 1 درجه، ولتاژ خروجی سنسور 10mV و به ازای دمای 100 درجه، خروجی سنسور 1V می‌باشد.



شکل 7-10: سنسور اندازه گیری دما LM35

این سنسور دارای 3 پایه می‌باشد، در صورتی که سنسور روبروی شما باشد (بتوانید نوشته‌هایش را ببینید) اولین پایه سمت چپ VCC سنسور (متصل به 5 ولت می‌شود)، پایه وسط ولتاژ خروجی (به ریزپردازنده متصل می‌شود) و پایه سوم GND سنسور است.

خروجی آنالوگ این سنسور توسط مبدل‌های آنالوگ به دیجیتال موجود در تراشه Atmega16 به دیجیتال تبدیل شده و به عنوان مثال روی LCD نمایش داده می‌شود.

مثال: با استفاده از LM35، دما را در بازه‌ی 0 تا 50 درجه سانتیگراد اندازه‌گیری و روی LCD نمایش دهید.

برای اندازه گیری دما در بازه‌ی 0 تا 50 درجه روابط ذیل برقرار است:

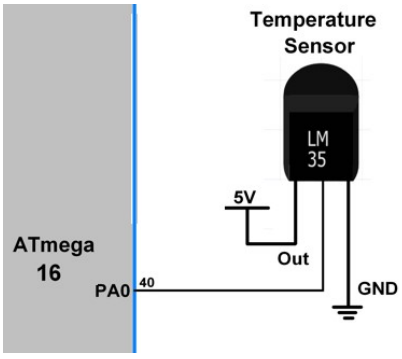
$$D_{OUT} = \frac{V_{LM35}}{V_{REF}} \times 2^{10}$$

$$V_{LM35} = 10mv \times T$$

اگر ولتاژ مرجع برابر با 5 ولت و مبدل آنالوگ به دیجیتال در حالت 10 بیتی کار کند دما برابر است با:

$$T = \frac{D_{OUT} \times 500}{2^{10}} \cong \frac{D_{OUT}}{2}$$

برای این منظور مانند شکل 7-11خروجی LM35 به پایه ADC0 و LCD به درگاه B متصل است.



شکل 7-11 نحوه اتصال LM35 به ریزپردازنده

سپس در قسمت CodeWizard تنظیمات مربوط به LCD و ADC انجام می‌شود و قطعه کد زیر به برنامه اضافه می‌گردد.

```
#include <mega16.h>

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

برنامه 3-7

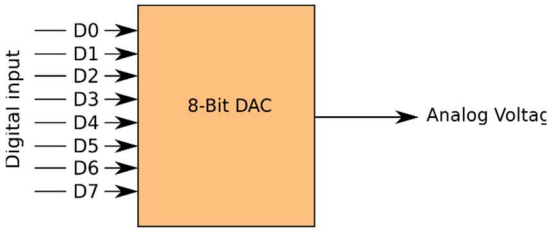
```
// Alphanumeric LCD functions
#include <alcd.h>

// Declare your global variables here

// Voltage Reference: AVCC pin
#define ADC_VREF_TYPE ((0<<REFS1) | (1<<REFS0) | (0<<ADLAR))

// Read the AD conversion result
unsigned int read_adc(unsigned char adc_input)
{
ADMUX=adc_input | ADC_VREF_TYPE;
// Delay needed for the stabilization of the ADC input voltage
delay_us(10);
// Start the AD conversion
ADCSRA|=(1<<ADSC);
// Wait for the AD conversion to complete
while ((ADCSRA & (1<<ADIF))==0);
ADCSRA|=(1<<ADIF);
return ADCW;
}

void main(void)
{
// Declare your local variables here
char str[20];
// Input/Output Ports initialization
// Port A initialization
// Function: Bit7=In Bit6=In Bit5=In Bit4=In Bit3=In Bit2=In Bit1=In
Bit0=In
DDRA=(0<<DDA7) | (0<<DDA6) | (0<<DDA5) | (0<<DDA4) | (0<<DDA3) |
(0<<DDA2) | (0<<DDA1) | (0<<DDA0);
// State: Bit7=T Bit6=T Bit5=T Bit4=T Bit3=T Bit2=T Bit1=T Bit0=T
PORTA=(0<<PORTA7) | (0<<PORTA6) | (0<<PORTA5) | (0<<PORTA4) |
(0<<PORTA3) | (0<<PORTA2) | (0<<PORTA1) | (0<<PORTA0);

// Port B initialization
// Function: Bit7=In Bit6=In Bit5=In Bit4=In Bit3=In Bit2=In Bit1=In
Bit0=In
DDRB=(0<<DDB7) | (0<<DDB6) | (0<<DDB5) | (0<<DDB4) | (0<<DDB3) |
(0<<DDB2) | (0<<DDB1) | (0<<DDB0);
// State: Bit7=T Bit6=T Bit5=T Bit4=T Bit3=T Bit2=T Bit1=T Bit0=T
PORTB=(0<<PORTB7) | (0<<PORTB6) | (0<<PORTB5) | (0<<PORTB4) |
(0<<PORTB3) | (0<<PORTB2) | (0<<PORTB1) | (0<<PORTB0);

// Port C initialization
// Function: Bit7=In Bit6=In Bit5=In Bit4=In Bit3=In Bit2=In Bit1=In
Bit0=In
DDRC=(0<<DDC7) | (0<<DDC6) | (0<<DDC5) | (0<<DDC4) | (0<<DDC3) |
(0<<DDC2) | (0<<DDC1) | (0<<DDC0);
// State: Bit7=T Bit6=T Bit5=T Bit4=T Bit3=T Bit2=T Bit1=T Bit0=T
```

```
PORTC=(0<<PORTC7) | (0<<PORTC6) | (0<<PORTC5) | (0<<PORTC4) |
(0<<PORTC3) | (0<<PORTC2) | (0<<PORTC1) | (0<<PORTC0);

// Port D initialization
// Function: Bit7=In Bit6=In Bit5=In Bit4=In Bit3=In Bit2=In Bit1=In
Bit0=In
DDRD=(0<<DDD7) | (0<<DDD6) | (0<<DDD5) | (0<<DDD4) | (0<<DDD3) |
(0<<DDD2) | (0<<DDD1) | (0<<DDD0);
// State: Bit7=T Bit6=T Bit5=T Bit4=T Bit3=T Bit2=T Bit1=T Bit0=T
PORTD=(0<<PORTD7) | (0<<PORTD6) | (0<<PORTD5) | (0<<PORTD4) |
(0<<PORTD3) | (0<<PORTD2) | (0<<PORTD1) | (0<<PORTD0);

// ADC initialization
// ADC Clock frequency: 1000.000 kHz
// ADC Voltage Reference: AVCC pin
// ADC Auto Trigger Source: Free Running
ADMUX=ADC_VREF_TYPE;
ADCSRA=(1<<ADEN) | (0<<ADSC) | (1<<ADATE) | (0<<ADIF) | (0<<ADIE) |
(0<<ADPS2) | (1<<ADPS1) | (1<<ADPS0);
SFIOR=(0<<ADTS2) | (0<<ADTS1) | (0<<ADTS0);

// Alphanumeric LCD initialization
// Connections are specified in the
// Project|Configure|C Compiler|Libraries|Alphanumeric LCD menu:
// RS - PORTB Bit 0
// RD - PORTB Bit 1
// EN - PORTB Bit 2
// D4 - PORTB Bit 4
// D5 - PORTB Bit 5
// D6 - PORTB Bit 6
// D7 - PORTB Bit 7
// Characters/line: 16
lcd_init(16);
while (1)
      {
 //       Place your code here
          read_adc(0);
          sprintf(str,"%d.%d",(ADCH*500)/1023,((
ADCH*500)/1023)%10);
          lcd_puts(str);
          delay_ms(1000);

      }
}
```

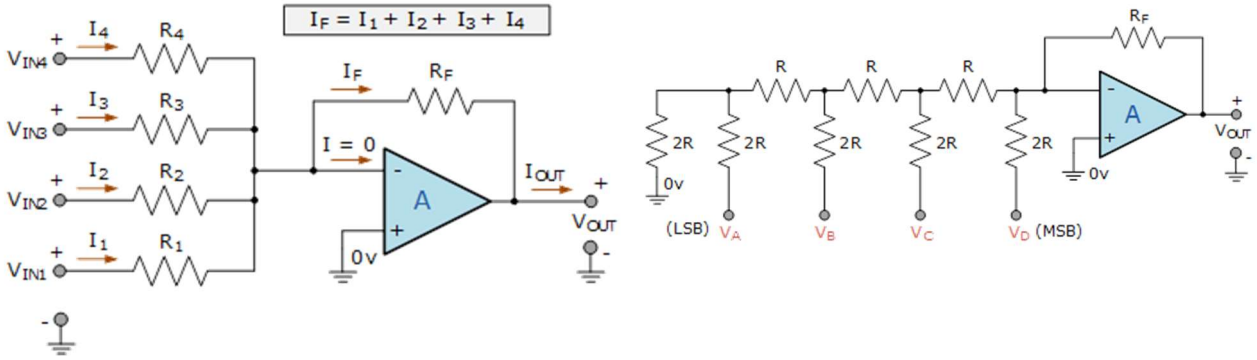
7.8 آشنایی با تبدیل سیگنال دیجیتال به آنالوگ

در تبدیل‌کننده‌های دیجیتال به آنالوگ مانند شکل 7-12 به ازای دریافت داده‌های دیجیتال، خروجی ولتاژی یا جریانی آنالوگ تولید می‌شود. مقدار دقت یا وضوح آن بر حسب تعداد ورودی‌ها تعیین می‌شود، اگر تعداد ورودی‌های

DAC به تعداد n باشد تعداد سطوح خروجی آن 2^n خواهد بود. پروسه‌ی تبدیل سیگنال دیجیتال به آنالوگ مانند شکل 7-13 به دو روش نردبانی[1] باینری وزن‌دار صورت می‌گیرد.

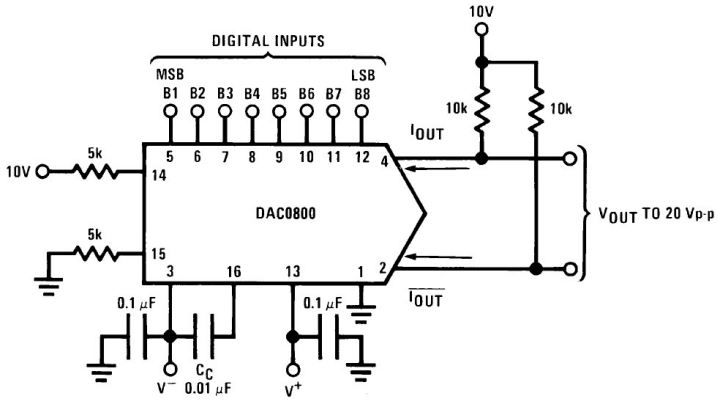


شکل 7-12: بلوک دیاگرام مبدل دیجیتال به آنالوگ



ب الف

شکل 7-13 : نمونه پروسه‌ی تبدیل سیگنال دیجیتال به آنالوگ، الف: نردبانی، ب: باینری وزن‌دار

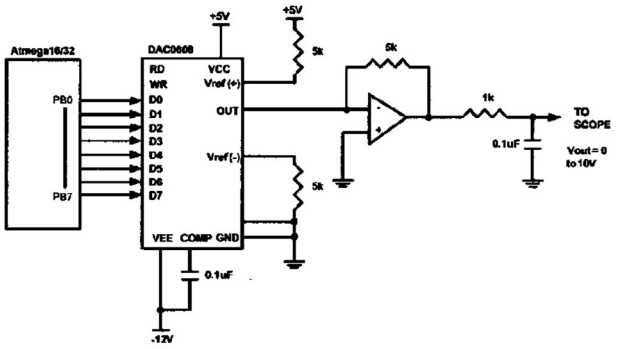


شکل 7-14 DAC0800

در برد آموزشی آزمایشگاه از ماژول DAC0800 به عنوان یک تبدیل‌کننده دیجیتال به آنالوگ هشت بیتی به روش باینری وزن‌دار استفاده شده است که دارای 256 سطح در خروجی می‌باشد. در شکل 7-14 نمایی از آن نشان داده شده است.

[1] ladder

D0 کم‌ارزش‌ترین بیت ورودی و V_{ref} ولتاژ ورودی است که باید به پایه 14 و 15 اعمال شود تا جریان مورد نیاز I_{ref} تامین گردد. بیشترین خروجی جریان I_{out} برابر 1.99mA است. مدار نحوه تبدیل جریان خروجی به ولتاژ مانند شکل 7-15 می‌باشد.



شکل 7-15: نحوه اتصالات به ریزپردازنده

جریان خروجی این تراشه از رابطه ذیل بدست می‌آید.

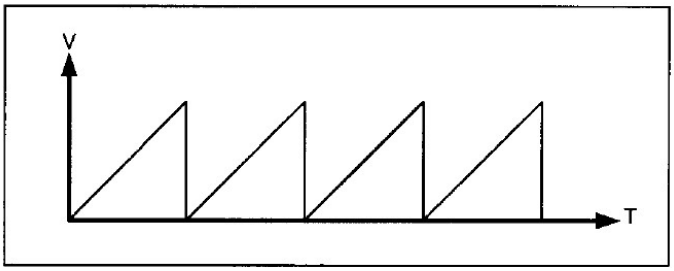
$$I_{out} = I_{ref}\left(\frac{D7}{2}+\frac{D6}{4}+\frac{D5}{8}+\frac{D4}{16}+\frac{D3}{32}+\frac{D2}{64}+\frac{D1}{128}+\frac{D0}{256}\right)$$

7.9 برنامه مبدل آنالوگ به دیجیتال

به عنوان نمونه مانند برنامه 7-4 می‌توان با استفاده از مبدل دیجیتال به آنالوگ یک موج دندانه اره‌ای مانند شکل 7-16 طراحی نمود.

برنامه 7-4

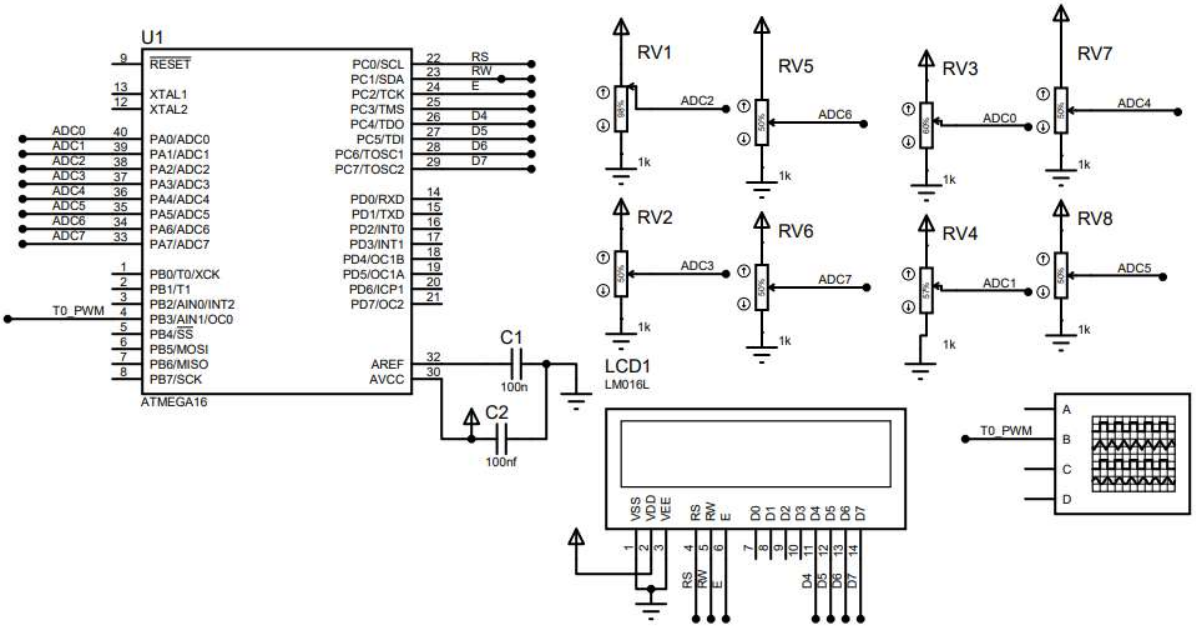
```
#include <mega16.h>
unsigned char i;        //define a counter
int main (void)
{
DDRB = 0xFF;
while (1)
{
PORTB = i;
i++;
}
return 0;
}
```



شکل 7-16: Step Ramp Output

7.10 برنامه‌های اجرایی مبدل‌های آنالوگ به دیجیتال

سخت افزار ذیل را در نظر بگیرید:



1- زیربرنامه‌ای بنویسید که ولتاژ هر یک از متغیرهای آنالوگ را بخواند و روی LCD بر حسب میلی‌ولت نشان دهد. (راهنمایی: تنظیمات CodeWizard را برای ADC بدون وقفه فعال نمایید در کدهای ایجاد شده برای خواندن مقدار دیجیتال هر یک از کانال‌ها از read_adc استفاده می‌شود. پس از اجرای صحیح این بند فراخوانی بخش ADC را از Main در قالب زیربرنامه جداگانه‌ای درآوردید. سپس زیربرنامه read_adc و تعاریف اولیه و زیربرنامه فراخوانی ADC را به فایل جداگانه منتقل نمایید. این فایل در بخش نهایی استفاده خواهد شد.)

2- زیر برنامه ای بنویسید که از طریق وقفه مقدار هر یک از متغیرهای آنالوگ را بخواند و در صورتی که تغییرات بیش از 5 درصد باشد روی LCD نشان دهد.(تنظیمات CodeWizard را برای ADC با وقفه فعال نمایید. پس از اجرای صحیح این بند فراخوانی بخش ADC را از Main در قالب زیربرنامه جداگانه‌ای درآوردید. سپس روال سرویس دهی وقفه و تعاریف اولیه و زیربرنامه فراخوانی ADC را به فایل جداگانه منتقل نمایید. این فایل در بخش نهایی استفاده خواهد شد. آخرین مقدار نشان داده شده روی LCD را با مقدار جدید هر ورودی آنالوگ مقایسه نمایید و چنانچه بیش از 5٪ تغییر داشت برای LCD ارسال نمایید بدین وسیله بار پردازشی ریزپردازنده کمتر شده و نوشته های روی LCD حالت چشمک زن نخواهد داشت.)

3- زیربرنامه‌ای بنویسید که با استفاده از تایمر صفر، یک پالس PWM تولید نماید و چرخه‌ی کار این پالس با سیگنال آنالوگ متصل به ADC0 تغییر نماید. راهنمایی: در این بخش مانند بند قبلی وقفه‌ی ADC فعال است.

4- بندهای فوق را در قالب یک پروژه در بیاورید. (راهنمایی: از فایل‌های جانبی تهیه شده در بندهای 1 و 2 استفاده نمایید. می‌توان ابتدا ADC را بدون وقفه فعال کرد و پس از نمایش ورودی‌های آنالوگ مجددا ADC را پیکربندی نمود و وقفه‌ی ADC را فعال کرد سپس بند 2 و 3 را به طور همزمان اجرا نموده به گونه‌ای که در حین اجرای برنامه تغییرات هر یک از ورودی‌ها نیز لحاظ گردد.)