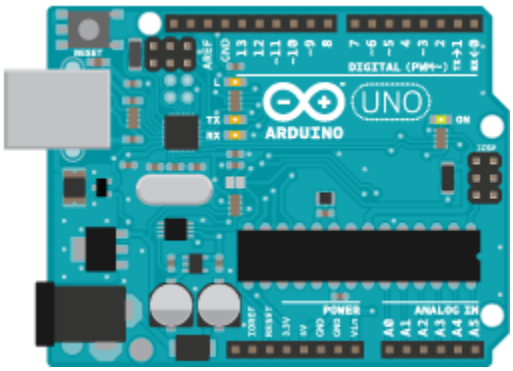


- مبدل آنالوگ به دیجیتال (ADC)
- آموزش ساخت ولت‌سنج با استفاده از مبدل آنالوگ در آردوینو
- آموزش ساخت دماسنج با استفاده از LM35 و مبدل آنالوگ در آردوینو
- تنظیم رزولوشن ADC (مخصوص آردوینو Due و Zero)
- مبدل دیجیتال به آنالوگ (PWM)
- آموزش کم‌وزیاد کردن نور LED با PWM
- آموزش تنظیم نور LED با ولوم و PWM
- تنظیم رزولوشن PWM (مخصوص آردوینو Due و Zero)
- دستورات تولید صدا با آردوینو
- اندازه‌گیری زمان تناوب (pulseIn)
- آموزش ساختن التراسونیک با ماژول `srf05` به کمک pulseIn

❖ مبدل آنالوگ به دیجیتال (ADC) :

مبدل سیگنال‌های آنالوگ به دیجیتال (Analog to Digital Converter=ADC)، مداری الکترونیکی که سیگنال‌های پیوسته آنالوگ را به داده‌های گسسته دیجیتالی یا رقمی تبدیل می‌کند.

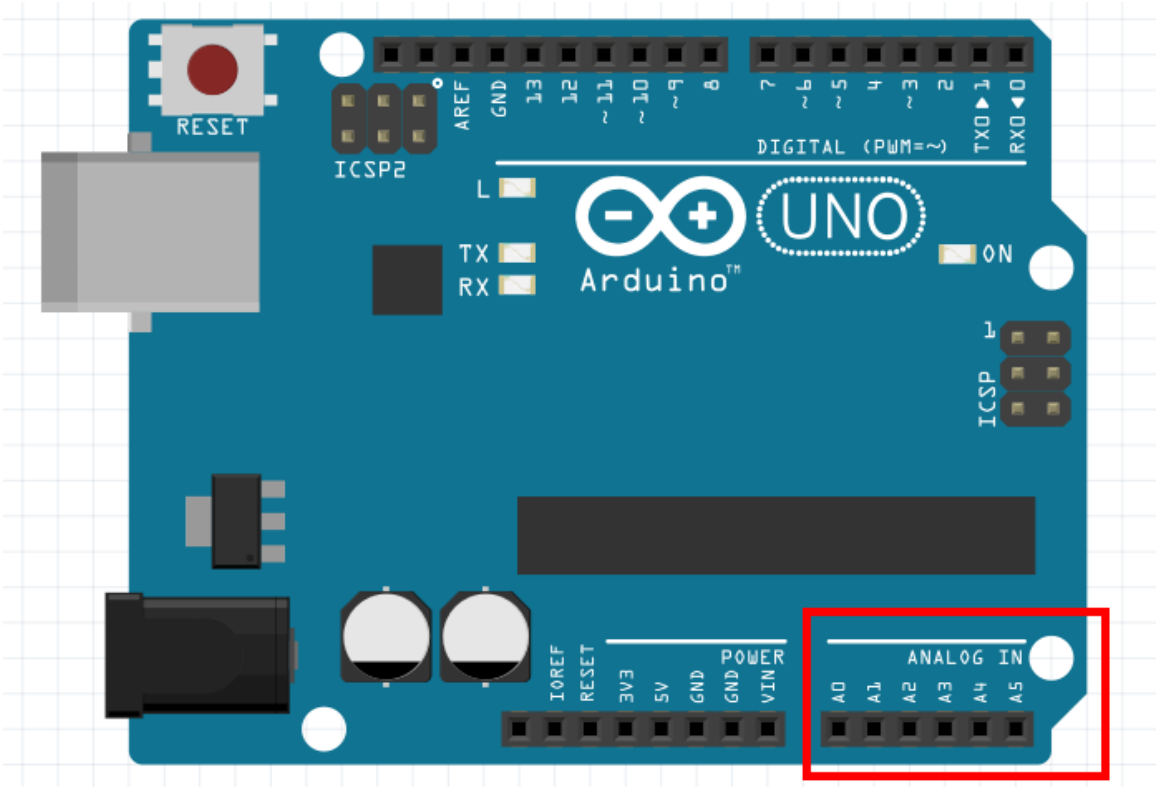
به زبان ساده‌تر ولتاژ و سیگنال‌های پیوسته آنالوگ هستند و داده (اعداد) های گسسته ، دیجیتال هستند . بعضی از سنسورها خروجی آنالوگ دارند به‌عنوان‌مثال سنسور دمای LM35 یک سنسور دمای آنالوگ است که خروجی آن ولتاژ است ، حالا آردوینو(میکرو) نمی تونه این ولتاژ رو درک کند . برای اینکه آردوینو(میکرو) بتونه با محیط بیرون ارتباط برقرار کنه ، روی میکروی آردوینو مدارهای مبدل آنالوگ تعبیه‌شده است که ولتاژ را به اعداد (دیجیتال) تبدیل می‌کند که این اعداد برای میکرو و برنامه‌نویسی قابل‌درک هستند عملکرد مبدل آنالوگ به شکل زیر هست :

به پایه‌ای که مبدل آنالوگ به دیجیتال است می‌توان 0 تا 5 ولت را بدهیم این ولتاژ توسط مبدل به اعداد 0 تا 1023 تبدیل می‌شود یعنی برای اعمال ولتاژ 0 عدد 0 و برای اعمال ولتاژ 5 عدد 1023 به ما داده می‌شود و اعداد بین این بازه هم به‌تناسب بازه از ولتاژ به عدد تبدیل می‌شوند مثلاً با اعمال ولتاژ 2.5 عدد 511 به ما داده می‌شود . حالا ما می‌توانیم روی این اعداد کارهای خاص را انجام دهیم و آن را در فرمول‌های خاص قرار دهیم

در بیشتر آردوینو ها تعداد ۶ مبدل آنالوگ داریم و در تعدادی مثل آردوینو نانو و آردوینو مینی تعداد ۸ مبدل داریم و در بعضی آردوینو ها هم مثل آردوینو مگا ۱۶ عدد مبدل آنالوگ داریم .

در آردوینو 100 میکروثانیه (0.0001 s) طول می‌کشد تا مبدل آنالوگ اطلاعات‌پایه آنالوگ را چک کند و تبدیل را انجام دهید به‌عبارت‌دیگر در هر ثانیه 10.000 بار پایه را چک و تبدیل را انجام می‌دهد

در شکل زیر پایه‌های مبدل آنالوگ برد آردوینو uno رو می‌بینید ، این برد ۵ مبدل داره که با علامت قرمز آن را جدا کرده‌ایم .



اکنون وقت آن رسیده است تا برنامه‌نویسی و استفاده از مبدل آنالوگ را توضیح دهیم

۱ – دستور خواندن مقدار آنالوگ

کاربرد دستور : با استفاده از این تابع دو عدد را مقایسه می‌کنیم و کدام کوچک‌تر (مینیمم) بود ، انتخاب می‌شود

شکل کلی دستور :

```
var = analogRead(pin)
```

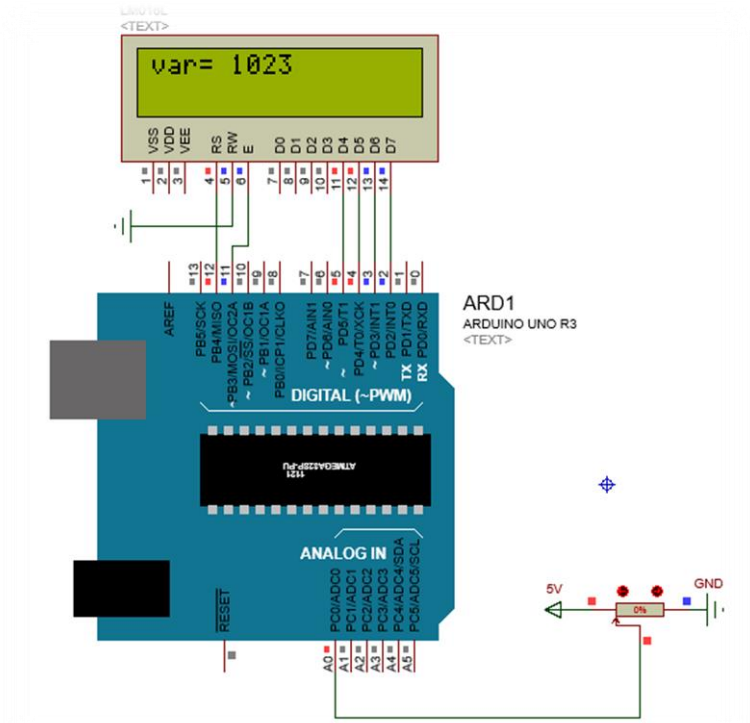
پارامترهای دستور :

pin : شماره پایه آنالوگ (0 تا 5 در اکثر بردها ، 0 تا 7 در آردوینو نانو و مینی و 0 تا 15 در آردوینو مگا

Var : متغیر برای ذخیره نتیجه (که اعدادی بین ۰ تا ۱۰۲۳ است)

مثال ۱ : یک پتانسیومتر را به پایه A0 آردوینو UNO متصل کنید و نتیجه را روی نمایشگر ۴ در ۱۶ نمایش دهید

شبیه‌ساز (Proteus 8) :



برنامه (Arduino 1.6) :

```
#include <LiquidCrystal.h>
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
////پیکربندی نمایشگر
int var ;
/////  تعریف متغیر
void setup() {
lcd.begin(16, 2);
}
void loop() {
var = analogRead(0);
/////خواندن مقدار آنالوگ از پایه A0 آردوینو
lcd.setCursor(0, 0);
lcd.print("var=");
/////نمایش متن  var= روی نمایشگر
lcd.setCursor(5, 0);
lcd.print(var);
/////  نمایش مقدار دیجیتال
delay(100);
lcd.clear();
///// توقف و پاک‌سازی نمایشگر
}
```

توجه : درصورتی‌که پایه A0 اردوینو آزاد باشد مبدل دائماً نوسان می‌کند و اعداد تصادفی را نمایش می‌دهد

مثال ۱-۱ : با استفاده از مبدل آنالوگ یک ولت‌سنج درست کنید که تا ۵ ولت DC را اندازه‌گیری کند

توضیح :

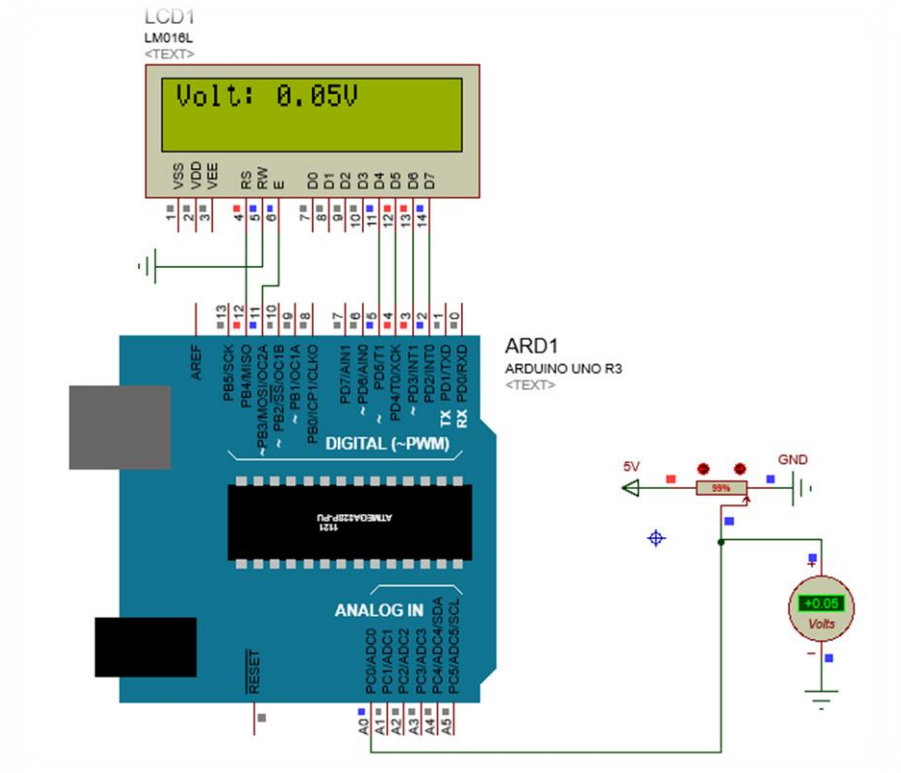
ولت‌سنج یا ولت‌متر دستگاهی است که برای اندازه‌گیری اختلاف‌پتانسیل الکتریکی در دو سر یک مدار الکتریکی بکار می‌رود. در اینجا چون ما می‌خواهیم یک ولت‌سنج را درست کنیم که تا ۵ ولت را اندازه‌گیری کند ، پس یابد بیشترین مقدار دیجیتال که 1023 را بر 6 . 204 تقسیم کنیم تا ۵ ولت به دست بیاید ، پس این شد یک قاعده کلی اگر مقدار آنالوگ خوانده‌شده بر 6 . 204 تقسیم شود ولتاژ به دست می‌آید

سؤال : چطوری فهمیدم باید تقسیم‌بر 6 . 204 بشه ؟

اومدم 1023 را تقسیم‌بر ۵ کردم چون می‌دانستیم با اعمال ۵ ولت عدد 1023 تشکیل‌شده است پس عدد 6 . 204 به دست آمد ، با تقسیم 1023 بر 6 . 204 مقدار عددی 5 به دست می‌آید که بیانگر ولتاژ است به‌این‌ترتیب اعداد بین 0 تا 1023 با تقسیم‌بر 6 . 204 به ولتاژ تبدیل می‌شوند .

توجه : با تقسیم 1023 بر 6 . 204 مقدار اختلاف‌پتانسیل برحسب ولت به دست می‌آید و با تقسیم‌بر 2046 . 0 اختلاف‌پتانسیل برحسب میلی ولت به دست می‌آید

شبیه‌ساز (Proteus 8) :



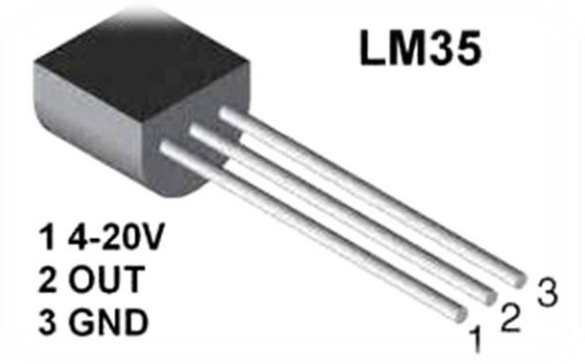
برنامه (Arduino 1.6) :

```
#include <LiquidCrystal.h>
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
///// پیکربندی نمایشگر
 float var ;
 /////  تعریف متغیر از جنس اعشاری
void setup() {
lcd.begin(16, 2);
}
void loop() {
var = analogRead(0);
/////var خواندن مقدار آنالوگ و ریختن آن در متغیر
var = var/204.6;
/////تقسیم کردن متغیر  var  بر عدد 204.6 تا ولتاژ برحسب ولت به دست بیاید
lcd.setCursor(0, 0);
lcd.print("Volt:");
///// نمایش متن
lcd.setCursor(6, 0);
lcd.print(var);
///// نمایش ولتاژ
lcd.setCursor(10, 0);
lcd.print("V");
///// نمایش متن
delay(100);
lcd.clear();}
```

مثال ۱-۲ : با استفاده از مبدل آنالوگ سنسور دمای LM35 را راه‌اندازی کنید

توضیح :

سنسور دمای استفاده‌شده در این پروژه LM35 هست .



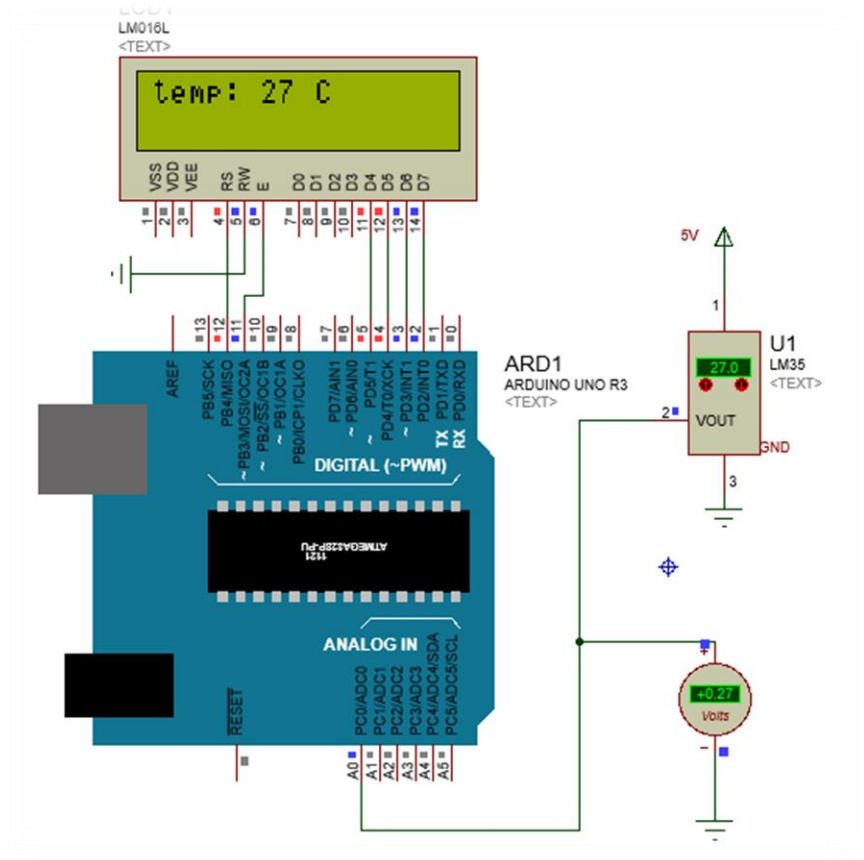
خروجی این سنسور از نوع آنالوگ هست و به ازای افزایش هر درجه سانتی‌گراد، خروجی را ۱۰ میلی ولت افزایش می‌دهد. برای سنجش مقدار دما برحسب درجه سانتی‌گراد ، باید خروجی این سنسور را برحسب میلی ولت اندازه‌گیری کرده و این مقدار را بر ۱۰ تقسیم نماییم.

به‌عنوان‌مثال : اگر خروجی lm35 را اندازه‌گیری کنیم و ۳۵۰ میلی ولت باشد آن را تقسیم‌بر ۱۰ می‌کنیم که می‌شود ۳۵ درجه سانتی‌گراد .

نکته : برای درست کردن دماسنج با استفاده از سنسور LM35 درواقع باید ابتدا یک ولت‌سنج درست کنیم که اختلاف‌پتانسیل رو برحسب میلی ولت نشان بده بعد اون رو بر ۱۰ تقسیم می‌کنیم و نتیجه به درجه سانتی‌گراد تبدیل میشه

توجه : در توضیحات برنامه ، از توضیح خطوطی که قبلاً توضیح داده‌شده‌اند خودداری می‌کنیم

شبیه‌ساز (Proteus 8) :




```
#include <LiquidCrystal.h>
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
/////
int var ;
/////
void setup() {
lcd.begin(16, 2);
}
void loop() {
var = analogRead(0);
/////  خواندن اطلاعات از پایه آنالوگ
var = var/0.2046;
///// تبدیل کردن مقدار دیجیتال به ولتاژ برحسب میلی ولت
var = var/10;
///// تقسیم کردن ولتاژ بر ۱۰ و به دست آوردن دما
lcd.setCursor(0, 0);
lcd.print("temp:");
/////
lcd.setCursor(6, 0);
lcd.print(var);
/////
lcd.setCursor(9, 0);
lcd.print("C");
/////
delay(100);
lcd.clear();
}
```

۲ – تنظیم کردن ولتاژ مرجع مبدل آنالوگ

به‌صورت پیش‌فرض ما میتونیم 0 تا 5 ولت را به پایه‌های مبدل آنالوگ بدیم ، ما با استفاده از ولتاژ مرجع می‌توانیم این دامنه را محدودتری کنیم ، مثلاً مبدل فقط 0 تا 3 ولت را پردازش کند .

کاربرد دستور : فعال‌سازی ولتاژ مرجع برای پایه‌های آنالوگ

شکل کلی دستور :

```
analogReference(type)
```

پارامترهای دستور :

type : می‌تواند یکی از این موارد باشد

(DEFAULT, INTERNAL, INTERNAL1V1, INTERNAL2V56, or EXTERNAL)

DEFAULT : همون حالت پیش‌فرض هست (دربردهای ۵ ولتی تا ۵ ولت و دربردهای ۳,۳ ولتی ۳,۳ ولت است)

INTERNAL: رفرنس داخلی که در ATmega168 یا ATmega328 1.1 ولت است و در ATmega8 2.56 ولت است (این گزینه در سری مگا کار نمی کنه)

INTERNAL1V1: رفرنس 1.1 ولت داخلی است (تنها در سری مگا کار می‌کند)

INTERNAL2V56: رفرنس 2.56 ولت داخلی است (تنها در سری مگا کار می‌کند)

EXTERNAL: رفرنس خارجی هست که 0 تا 5 ولت را باید به پایه AREF اعمال کنیم .

با توجه به ساده بودن و کاربرد کم این دستور برای آن مثال نمی‌آوریم

❖ تنظیم رزولوشن ADC (مخصوص آردوینو Due و Zero)

رزولوشن (Resolution) به معنی وضوع است ، حتماً درباره رزولوشن صفحه‌نمایش گوشی ، تلویزیون و ... شنیده‌اید ، هرچه رزولوشن بالاتر باشد تعداد نقاط تصویر (پیکسل) بیشتر خواهند بود و درنتیجه کیفیت تصویر بالاتر خواهد بود ، در مورد ADC هم به همین شیوه است ، هر چه رزولوشن مبدل آنالوگ بیشتر باشد می‌تواند سیگنال ورودی را به بخش‌های ریزتری تقسیم کند و درنتیجه اطلاعات دقیق‌تری به ما بدهد . آردوینو هایی که بر پایه AVR هستند دارای رزولوشن ثابت 10 بیت هستند ولی در آردوینو هایی مثل Due و Zero که بر پایه میکرو پرقدرت ARM هستند این امکان را به ما می‌دهند که بتوانیم رزولوشن ADC را تنظیم و تغییر دهیم

در زیر دامنه چند رزولوشن پرکاربرد را بیان می‌کنیم :

چون هر بیت دارای ۲ ارزش 0 و 1 است پس وقتی گفته میشه رزولوشن X بیت یعنی 2 به توان X و ولتاژ ۵ ولتی که به پایه آنالوگ داده می‌شود به 2^X قسمت تقسیم می‌شود

8 بیت : ولتاژ 0 تا 5 ولت ورودی به 0 تا 256 قسمت تقسیم می‌شود

10 بیت : ولتاژ 0 تا 5 ولت ورودی به 0 تا 1024 قسمت تقسیم می‌شود

12 بیت : ولتاژ 0 تا 5 ولت ورودی به 0 تا 4096 قسمت تقسیم می‌شود

16 بیت : ولتاژ 0 تا 5 ولت ورودی به 0 تا 65536 قسمت تقسیم می‌شود

تنظیم رزولوشن ADC (مخصوص آردوینو Due و Zero)

آردوینو Due و Zero دارای رزولوشن داخلی با دقت 12 بیت هستند ، ولی با استفاده از دستور زیر می‌توان رزولوشن را روی ۱ تا ۳۲ تنظیم کرد . به این نکته دقت کنید که تا رزولوشن 12 بیت دارای دقت واقعی است و از 12 بیت به بالا رزولوشن تقریبی است

کاربرد دستور : تغییر رزولوشن (مخصوص آردوینو Due و Zero)

شکل کلی دستور :

```
analogReadResolution(bits)
```

پارامترهای دستور :

bits : تعیین دقت رزولوشن بر پایه بیت که می‌تواند 1 تا 32 باشد ، این دستور همراه دستور analogRead کاربرد دارد و روزولیشن اطلاعات ورودی آن را تعیین می‌کند

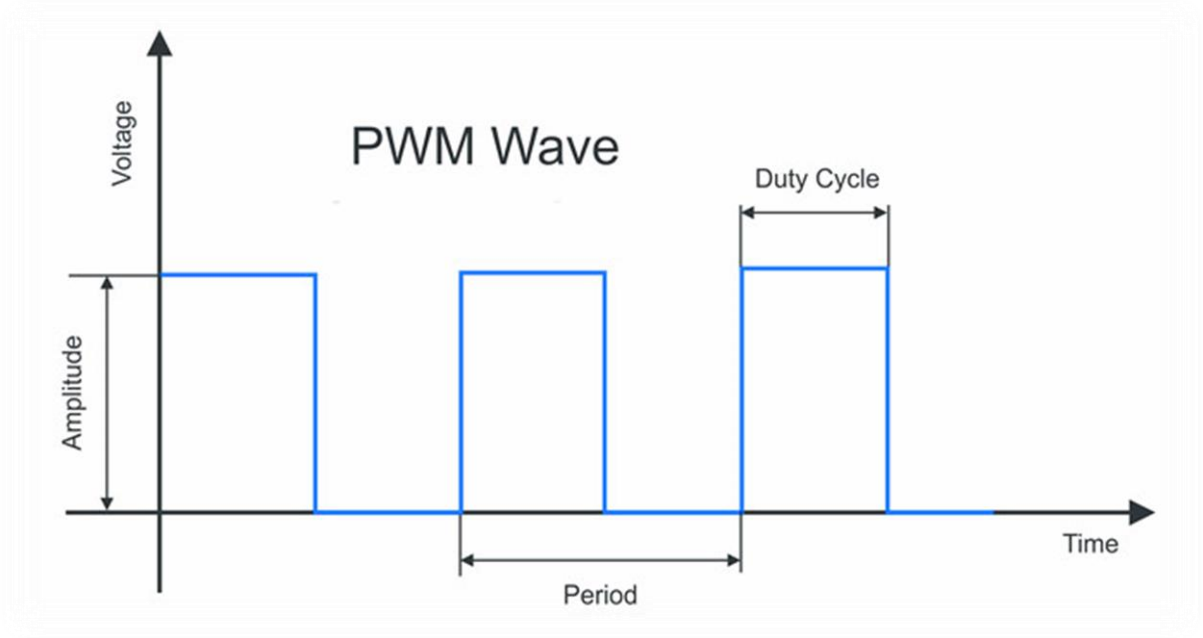
چون این بردها در پروتیوس قابلیت شبیه‌سازی ندارند و ما هم به این بردها دسترسی نداریم مجبوریم تنها به نحوه استفاده از آن بپردازیم و نمی‌توانیم مثالی عملی بیاوریم .

مثال : این مثال ، مثال ۱ ابتدای آموزش است که رزولوشن آن ۱۰ بیت است ، در اینجا آن را به ۱۶ بیت تبدیل می‌کنیم

```
#include <LiquidCrystal.h>
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
/////
int var ;
/////
void setup() {
lcd.begin(16, 2);
}
void loop() {
analogReadResolution(16);
//// تبدیل زرولیشن به ۱۶ بیت
var = analogRead(0);
///// خواندن اطلاعات از مبدل ۱۶ بیتی
lcd.setCursor(0, 0);
lcd.print("var=");
/////
lcd.setCursor(5, 0);
lcd.print(var);
/////
delay(100);
lcd.clear();
/////
}
```

❖ مبدل دیجیتال به آنالوگ (PWM)

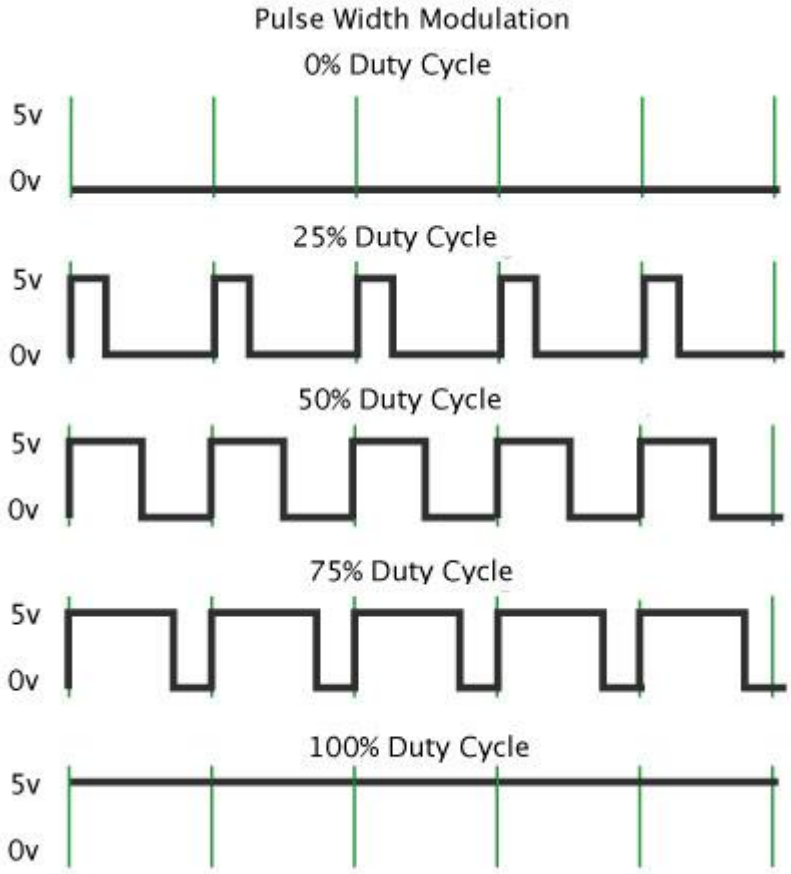
PWM مخفف Pulse Width Modulation به معنای مدولاسیون عرض پالس است . PWM کاربردهای زیادی دارد به‌عنوان‌مثال کم‌وزیاد کردن نور LED ها ، کم‌وزیاد کردن سرعت موتور ، راه‌اندازی موتور براشلس و سرو موتور و بسیاری کار دیگر .



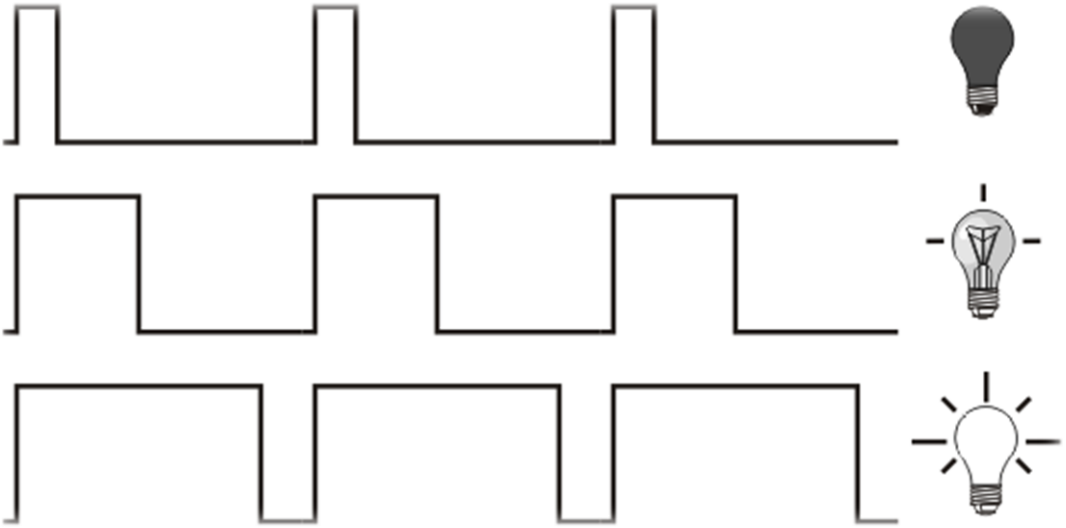
PWM موجی مربعی است که در برخی زمان‌ها 0 و برخی زمان‌ها 1 است و این 0 و 1 شدن‌ها با فرکانس مرتبی تکرار می‌شود (0 را معادل 0 ولت و 1 را معادل ۵ ولت در نظر بگیرید).

PWM مانند همه موج‌ها ، دارای دامنه (Amplitude) ، دور تناوب (Period) فرکانس است. عبارت دیگری که در PWM مورداستفاده قرار می‌گیرد Duty Cycle است . دیوتی سایکل مدت‌زمان 1 بودن به مدت‌زمان کل دور تناوب (Period) در هر نوسان کامل موج است که معمولاً برحسب درصد (٪) نمایش داده می‌شود. به‌عنوان‌مثال اگر Duty Cycle یک موج PWM برابر با 40٪ باشد به این معنی است که در هر نوسان کامل٪۴۰ ولتاژ برابر با 1 (VCC) و در ٪۶۰ اوقات ولتاژ برابر 0 (GND) است. همان‌گونه که می‌دانید در چنین حالتی ولتاژ مؤثر یا Vrms برابر با ٪۴۰ VCC خواهد بود.

مثال : اگر با یک میکرو با تغذیه ۵ ولت ، موج PWM با دیوتی سایکل ٪50 ایجاد نمایید ولتاژ مؤثر (RMS) شما برابر ٪50 ۵ ولت یا به عبارتی ۲,۵ ولت خواهد بود . در شکل زیر تعدادی موج PWM با فرکانس ثابت و دیوتی سایکل متفاوت نمایش داده‌شده است.



شکل زیر توضیح ساده‌ای از PWM ارائه میده



حالا که با PWM آشنا شدید و تئوری کارکرد آن را بررسی کردیم به سراغ استفاده از آن در آردوینو می‌رویم

توضیح : در آردوینو تعدادی از پایه‌ها را می‌توان به‌عنوان PWM تنظیم کرد ، روی آردوینو بردهای آردوینو پایه‌هایی که PWM دارند با علامت مشخص می‌شوند ، به‌عنوان‌مثال درهایی که با میکرو Atmega168 یا Atmega328 درست می‌شوند پایه‌های 3, 5, 6, 9, 10 و 11 را می‌توان به‌عنوان pwm تنظیم کرد و در آردوینو مگا 2 تا 13 و 44 تا 46 و در آردوینو های قدیمی که با اتمگا 8 درست می‌شوند پایه 10 , 9 و 11 را می‌توان به‌عنوان pwm تنظیم کرد

کاربرد دستور : تولید موج pwm توسط آردوینو

شکل کلی دستور :

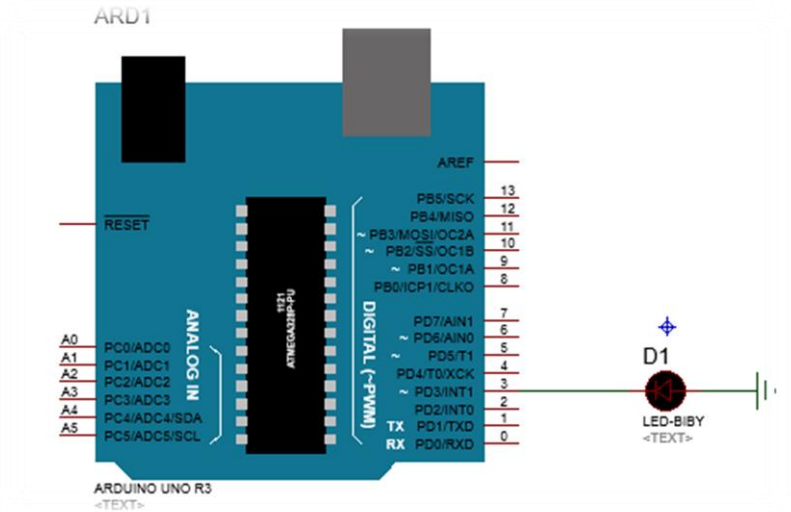
```
analogWrite(pin, value)
```

پارامترهای دستور :

pin: شماره پایه آردوینو که قرار است به‌عنوان خروجی موج تنظیم شود

value : میزان دیوتی سایکل که عددی بین 0 تا 255 است ، (در صورت قرار دادن 0 خروجی 0 ولت می‌شود و در صورت قرار دادن عدد 255 خروجی 5 ولت می‌شود)

مثال۲ – ۱ : یک LED را به پایه 3 آردوینو UNO وصل کنید و نور آن را از کم به زیاد تغییر دهید

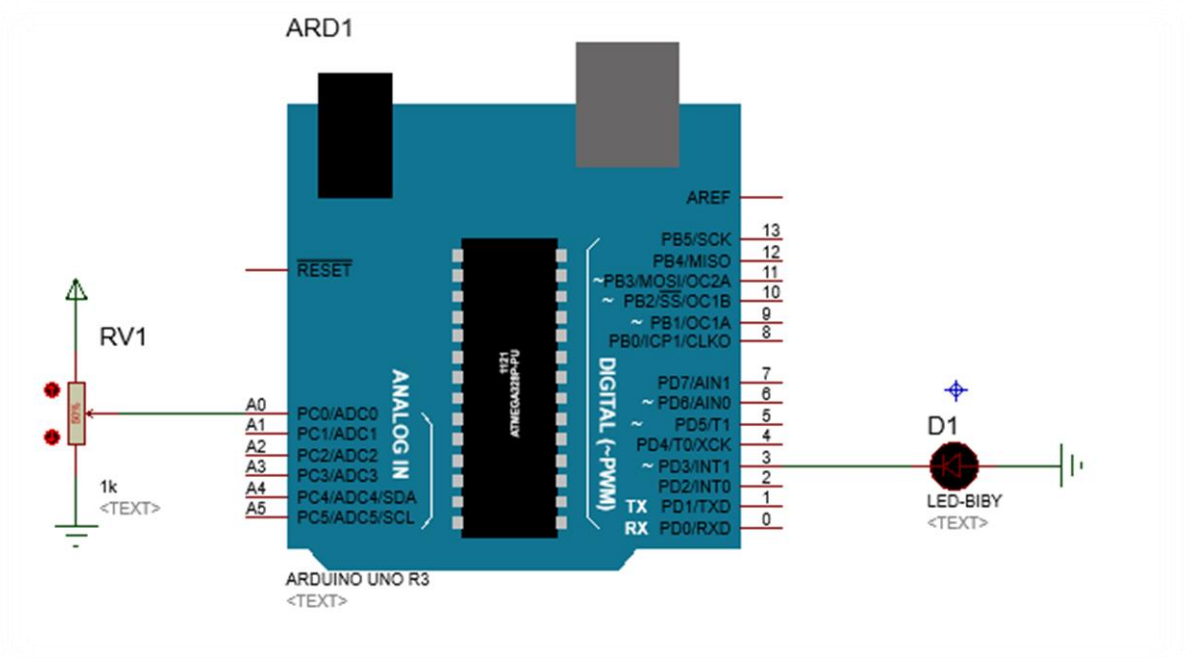


```
byte a = 0;
/// تعریف متغیر از جنس بایست که تا ۲۵۵ را در خود نگه می‌دارد
/// مقدار اولیه این متغیر برابر با صفر است
void setup()
{

}
void loop()
{
analogWrite(3, a);
///مقدار عددی که در a قرار دارد را به‌عنوان موج pwm به پایه 3 آردوینو می‌دهد
a++;
delay(100);
////هر ۱۰۰میلی‌ثانیه یک واحد به a افزوده می‌شود و وقتی a به ۲۵۵رسید دوباره از ۰ شروع می‌شود
}
```

نکته : پروتیوس قابلیت شبیه‌سازی موج pwm را ندارد ، برای تست این مثال باید عملاً مدار را پیاده کنید

مثال ۲ –۲ : یک ولوم را به آردوینو وصل کنید سپس با استفاده از pwm نور یک LED را کم‌وزیاد کنید



```
int a ;
/// تعریف متغیر
void setup()
{
}
void loop()
{
a = analogRead(0);
/// خواندن مقدار آنالوگ که عددی بین ۰ تا ۱۰۲۳ است
a = a/4 ;
/// اطلاعات دریافتی را بر ۴ تقسیم می‌کنیم تا بین ۰ تا ۲۵۵ شود
analogWrite(3, a);
/// ایجاد سیگنال روی پایه شماره ۳ آردوینو

}
```

نکته : پروتیوس قابلیت شبیه‌سازی موج pwm را ندارد ، برای تست این مثال باید عملاً مدار را پیاده کنید

تنظیم رزولوشن PWM (مخصوص آردوینو Due و Zero)

قبلاً در مورد رزولوشن ADC توضیح دادیم ، رزولوشن PWM هم به همین شکل است ، در اینجا عددی که قرار است به ولتاژ تبدیل کنیم هرچه بزرگ‌تر باشد دقت PWM بیشتر است .

آردوینو Due و Zero دارای رزولوشن PWM با دقت 8 تا 12 بیت هستند ، ولی با استفاده از دستور زیر می‌توان رزولوشن را روی ۱ تا ۳۲ تنظیم کرد . به این نکته دقت کنید که تا رزولوشن 8 تا 12 بیت دارای دقت واقعی است و از 12 بیت به بالا و 8 به پایین رزولوشن تقریبی است

کاربرد دستور : تغییر رزولوشن PWM (مخصوص آردوینو Due و Zero)

شکل کلی دستور :

```
analogWriteResolution(bits)
```

پارامترهای دستور :

bits : تعیین دقت رزولوشن بر پایه بیت که می‌تواند 1 تا 32 باشد ، این دستور همراه دستور analogWrite کاربرد دارد و روزولیشن اطلاعات خروجی آن را تعیین می‌کند

چون این بردها در پروتیوس قابلیت شبیه‌سازی ندارند و ما هم به این بردها دسترسی نداریم مجبوریم تنها به نحوه استفاده از آن بپردازیم و نمی‌توانیم مثالی عملی بیاوریم .

مثال : این مثال ، (مثال۲ – ۱) است که رزولوشن آن ۸ بیت است ، در اینجا آن را به ۱۲ بیت تبدیل می‌کنیم

```
byte a = 0;
/// تعریف متغیر از جنس بایست که تا ۲۵۵ را در خود نگه می‌دارد
/// مقدار اولیه این متغیر برابر با صفر است
void setup()
{

}
void loop()
{
analogWriteResolution(12);
/// تغییر رزولوشن خروجی و تنظیم روی ۱۲ بیت
analogWrite(3, a);
///مقدار عددی که در a قرار دارد را به‌عنوان موج pwm به پایه 3 آردوینو می‌دهد
a++;
delay(100);
////هر ۱۰۰میلی‌ثانیه یک واحد به a افزوده می‌شود و وقتی a به ۲۵۵رسید دوباره از ۰ شروع می‌شود
}
```

❖ دستورات تولید صدا با آردوینو

آردوینو به ما این امکان را می‌دهد که تن‌های صوتی را روی پین‌های آردوینو تولید کنیم ، تن‌های صوتی در درواقع موج‌های مربعی با دیوتی سایکل 50% هستند ، برای تولید این تن‌ها باید به آردوینو یک بلندگو و بازر وصل شود ، حداکثر و حداقل فرکانسی که بردهای آردوینو می‌توانند تولید کنند کمی باهم متفاوت است که در جدول زیر می‌بینید

برد آردوینو	حداقل فرکانس (Hz)	حداکثر فرکانس (Hz)
Uno, Mega, Leonardo & other board	31	65535
Gemma	عدم پشتیبانی	عدم پشتیبانی
Zero	41	275000
Due	عدم پشتیبانی	عدم پشتیبانی

اکنون به برنامه‌نویسی آن در آردوینو می‌پردازیم

کاربرد دستور : تولید تن‌های صوتی توسط آردوینو

شکل کلی دستور :

```
tone(pin, frequency)
tone(pin, frequency, duration)
```

پارامترهای دستور :

pin : پایه‌ای که می‌خواهیم روی آن تن صوتی تولید کنیم

Frequency : فرکانس تن صدایی که تولید می‌شود در واحد هرتز (Hz) (نوع متغیر: unsigned int)

Duration : مدت‌زمان دوره تن برحسب میلی‌ثانیه (اختیاری) (نوع متغیر: unsigned long)

نکته : با اجرای دستور بالا تولید صدا شروع می‌شود ولی متوقف نمی‌شود ، برای توقف تولید تن باید از دستور زیر استفاده کرد

کاربرد دستور : توقف تولید تن‌های صوتی توسط آردوینو

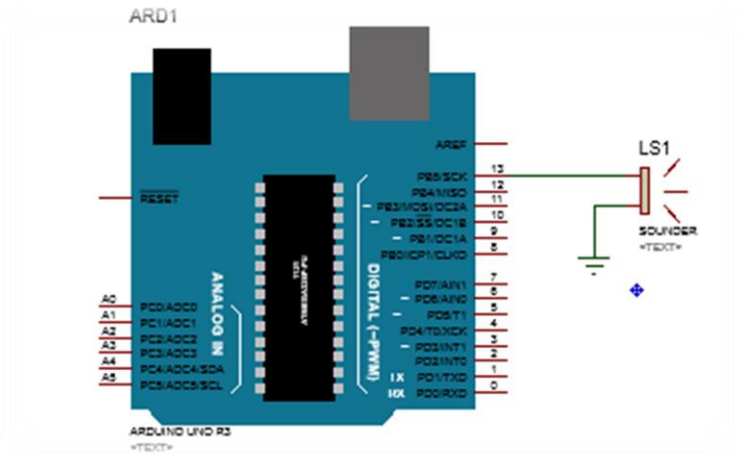
شکل کلی دستور :

```
noTone(pin)
```

پارامترهای دستور :

pin : پایه‌ای که باید تن صدای آن را قطع کنیم

مثال۳ - ۱ : با استفاده از آردوینو روی پایه ۱۳ تن صوتی ایجاد کنید



```
unsigned int a = 310 ;
/// تعریف متغیر و قرار دادن مقدار پیش‌فرض ۳۱۰
void setup()
{
}
void loop()
{
tone(13, a) ;
/// ایجاد تن روی پایه ۱۳ و با فرکانس ۳۱۰ هرتز
}
```

مثال۳ – ۲ : با استفاده از ولوم صدای تن را تغییر دهید

```
int a ;
/// تعریف متغیر
void setup()
{
}
void loop()
{
a = analogRead(0);
/// خواندن اطلاعات از پایه آنالوگ
tone(13, a);
/// تولید تن روی پایه ۱۳
delay(100);

}
```

❖ اندازه‌گیری زمان تناوب (pulseIn)

تابع pulseIn برای اندازه‌گیری زمان پالس اعمالی به پایه مدنظر در آردوینو به کار می‌رود ، زمانی که حساب می‌شود برحسب میکروثانیه (microseconds) است ، این تابع می‌تواند طول پالس از 10 میکروثانیه تا 3 دقیقه را اندازه بگیرد ، اکنون به توضیح توابع و متغیرهای آن در آردوینو می‌پردازیم و سپس توضیحات تکمیلی را ارائه می‌دهیم

کاربرد تابع : اندازه‌گیری زمان پالس اعمالی

شکل کلی تابع :

```
pulseIn(pin, value)
pulseIn(pin, value, timeout)
```

پارامترهای تابع :

pin : پایه‌ای که می‌خواهیم زمان پالس اعمالی به آن را اندازه بگیریم

value : نوع پالس بری شروع اندازه‌گیری زمان (می‌تواند HIGH یا LOW باشد)

timeout: مدت‌زمان (برحسب میکروثانیه) که منتظر می‌شود پالس تکمیل شود اگر پالس تکمیل نشد تابع عدد 0 را برمی‌گرداند ، در حالت پیش‌فرض این زمان 1 ثانیه است ، این قسمت اختیاری است و می‌توانید آن را حذف کنید که در این صورت این زمان 1 ثانیه حساب می‌شود

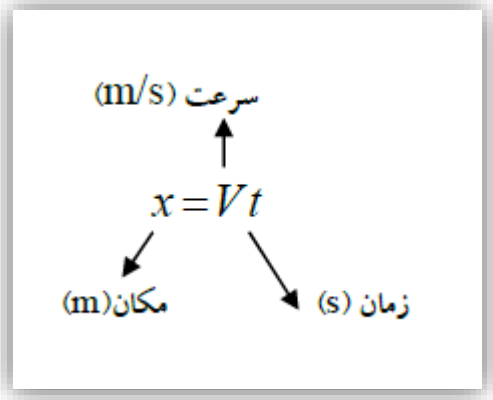
نکته : به‌عنوان‌مثال اگر تابع HIGH باشد و پایه 3 را برای ورودی پالس در نظر بگیریم مانند زیر:

```
pulseIn(3, HIGH)
```

تابع pulseIn منتظر می‌شود تا پایه شماره 3 فعال (HIGH) شود ، وقتی این پایه HIGH شد ، شمارش زمان شروع می‌شود و شمارش زمان تا وقتی ادامه پیدا می‌کند که پایه شماره 3 غیرفعال (LOW) شود ، وقتی این پایه LOW شد ، شمارش زمان متوقف می‌شود و زمانی که حساب‌شده به ما داده می‌شود .

مثال۴ – ۱ : با استفاده از دستور pulseIn و سنسور SRF05 یک‌فاصله سنج درست کنید

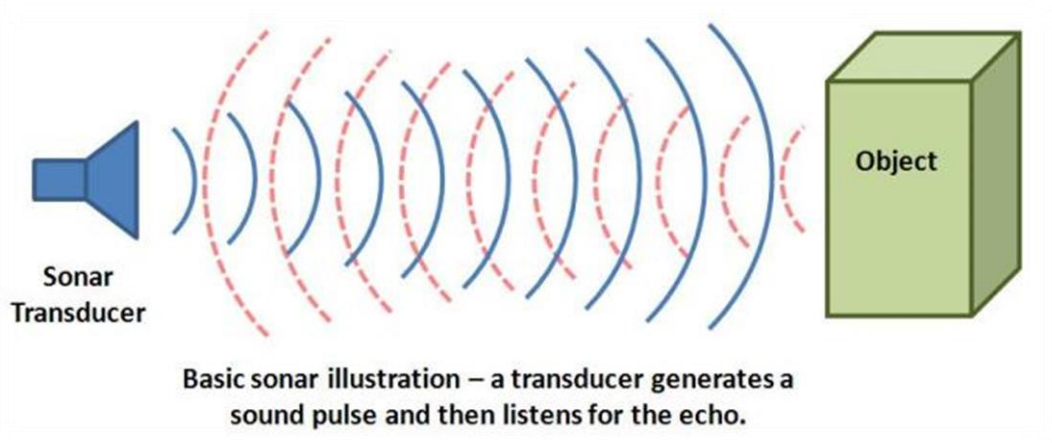
توضیح : برای اندازه‌گیری فاصله با استفاده از امواج التراسونیک به پارامترهای سرعت و زمان صوت نیاز داریم سپس با جایگذاری این اطلاعات در فرمول زیر به‌راحتی فاصله به دست می‌آید



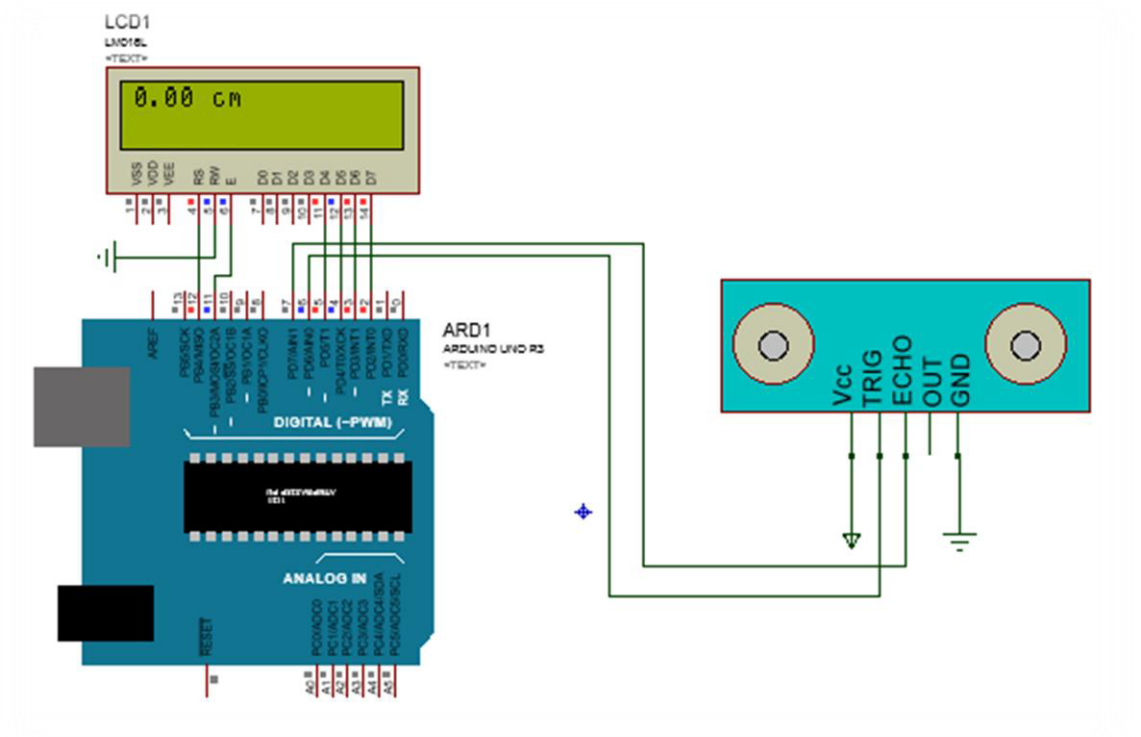
چون ما با استفاده از امواج صوتی مسافت را به دست می‌آوریم پس به مؤلفه V یعنی سرعت دسترسی داریم و می‌دانیم که سرعت صوت در هوا برابر با 340 متر بر ثانیه است ، تنها کافی است زمان رفت و برگشت صوت را به دست بیاوریم و در فرمول روبه‌رو جایگذاری کنیم

ما از ماژول التراسونیک SRF05 استفاده می‌کنیم که دارای پایه‌های GND ، VSS ، OUT ، ECHO و TRIG است

عملکرد این ماژول به این شکل است که ما یک پالس به پایه TRIG می‌دهیم با این کار ماژول یک صوت با فرکانس 40KHz ارسال می‌کند ، هر وقت این صوت به مانع برخورد و برگشت ماژول آن را آشکار می‌کند و پایه ECHO را فعال می‌کند به‌این‌ترتیب ما با اندازه‌گیری فاصله زمانی بیت روشن شدن پایه TRIG و پایه ECHO می‌توانیم فاصله رفت‌وبرگشت صوت را اندازه بگیریم



Basic sonar illustration – a transducer generates a sound pulse and then listens for the echo.



در شکل بالا شماتیک مدار را می‌بینید ، حالا به برنامه‌نویسی آن می‌پردازیم

```
float x;
float v =0.034;
float t;
```
در اینجا متغیرهای لازم برای فرمول x=vt را تعریف می‌کنیم////

```
#include <LiquidCrystal.h>
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
```
////نمایشگر را کانفیگ می‌کنیم
```
void setup()
{

  pinMode(6, OUTPUT);//TRIG
```
/// پایه ۶ آردوینو را به‌عنوان خروجی تعریف می‌کنیم
```
  pinMode(7, INPUT);//ECHO
```
/// پایه ۷ میکرو را به‌عنوان ورودی تعریف می‌کنیم
```
}

void loop()
{
digitalWrite(6, HIGH);
delayMicroseconds(20);
digitalWrite(6, LOW);
```
در اینجا به مدت ۲۰ میکروثانیه پایه ۶ آردوینو را فعال می‌کنیم تا ماژول التراسونیک فعال شورد یک‌تن صوتی بفرستد
```
t = pulseIn(7, HIGH);
```
همزمان با فعال کردن پایه ۶ آردوینو شروع به شمارش زمان می‌کنیم تا وقتی پایه ۷ فعال شود و شمارش متوقف شود و زمان به ما داده شود
```
t=t/2;
```
زمانی که به‌دست‌آمده است مال رفت‌وبرگشت صوت است درصورتی‌که ما فقط زمان رسیدن صدا به جسم را می‌خواهیم پس باید زمان را بر ۲ تقسیم کنیم
```
x=v*t;
```

```
حالا فرمول ما محاسبات را انجام می‌دهد ، مقدار v که همان سرعت صوت در هوا بود را در ابتدا تعریف کردیم و
زمان هم که با دستور بالا دریافت شد ، حالا اطلاعات جایگذاری و مقدار نهایی در متغیر x قرار می‌گیرد
lcd.begin(16, 2);
lcd.setCursor(0, 0);
lcd.print("d:");
lcd.print(x);
lcd.print(" cm");
////اطلاعات را روی نمایش نشان می‌دهیم

;(delay(1000
lcd.clear)(;

//// یک ثانیه توقف همراه با پاک کردن نمایشگر
}
```

نکات مهم : در فرمول $x=vt$ سرعت برحسب m/s است و زمان (t) برحسب ثانیه (s) و درنهایت فاصله برحسب متر به دست می‌آید ، الآن چند مسئله وجود دارد که باید حل شوند ، اولاً ما می‌خواهیم فاصله برحسب سانتی‌متر به دست آید ، دوما زمانی که از تابع `pulseIn` به دست می‌آید برحسب میکروثانیه است و نمی‌توان از آن به‌صورت مستقیم در فرمول استفاده کرد

الآن دو راه وجود دارد ، یا باید زمانی که از تابع به دست می‌آید را به ثانیه تبدیل کنیم ، یا باید m/s که در فرمول وجود دارد به cm/us تبدیل کنیم ، خب ما راه‌حل دوم را انتخاب می‌کنم . پس الآن ما باید کار زیر را انجام دهیم

سرعت صوت در هوا که $340m/s$ است را به cm/us (سانتی‌متر بر میکروثانیه) تبدیل می‌کنیم با این کار اولاً مسافت برحسب سانتی‌متر به دست می‌آید دوما می‌توانیم مستقیماً میکروثانیه را در فرمول به کار ببریم .

$340m/s = 0.034cm/us$

پس الآن متوجه شدید چرا مقدار متغیر t در ابتدای برنامه برابر 0.034 تعریف‌شده بود

حالا اگر در فرمول جایگذاری کنم می‌بینید به شکل زیر خواهد بود:

$X(m)=V(m/s)*T(s)$

$X(cm) = V(cm/us)*T(us)$