**باسمه تعالی**

# صنایع رباتیک آراد

## بزرگترین خانواده رباتیک ایران

# فصل چهارم

**طراح و برنامه نویس : مهندس تالیا براری**

**نویسنده :مهندس فراز امیرغیاثوند**

**تاریخ انتشار آبان ۱۳۹۰**

کسی که می خواهد کاری را انجام دهد ، راهش را پیدا می کند و کسی که نمی خواهد ، بهانه اش را

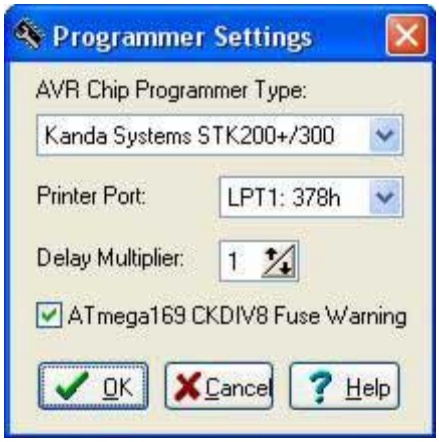
# فصل چهارم

## جلسه‌ی سی و یکم

تنظیم نرم افزاری پروگرامر درCodeVision ، ساختارهای کنترلی، ساختار if_else و....

این جلسه قراره در ابتدا نحوه‌ی تنظیم CodeVision را برای پروگرامری که در جلسه‌ی پیش ساختیم مطرح کنیم و بعد از اون، کمی هم آموزش زبان C را دنبال کنیم.

در جلسات پیش با بخش پروگرامر(Programmer) در CodeVision آشنا شدیم. برای استفاده از نرم افزار پروگرامر، ابتدا باید آن را با توجه به سخت‌افزاری که برای پروگرام کردن در اختیار داریم تنظیم کنیم. برای این کار CodeVision را باز کنید و از منوی بالا گزینه‌ی "Setting" را انتخاب کنید. سپس گزینه‌ی "Programmer" را انتخاب کنید. پنجره‌ای به شکل زیر باز می‌شود.




در این پنجره باید نوع سخت افزار پروگرامری را که در اختیار دارید را انتخاب کنید.

پروگرامری که ما جلسه‌ی پیش آموزش دادیم همین STK200+/300 است که معمولاً به صورت پیش فرض انتخاب شده است.

قسمت Delay Multiplier هم نیاز به تنظیم خاصی ندارد. با استفاده از این قسمت می‌توان کمی وقفه در عملیات پروگرام شدن به وجود آورد که این کار ممکن است برخی مشکلات احتمالی را در هنگام پروگرام کردن برطرف کند.

در این قسمت نیاز به تنظیم بخش دیگری نیست و پروگرامر آماده‌ی استفاده است.

تا به اینجا ما مقدمات کار با میکروکنترلرهای خانواده‌ی AVR را یاد گرفتیم و الان می‌تونیم در حد آماتور از میکروکنترلر در مدارهایی که طراحی می‌کنیم، استفاده کنیم.

اما کار با میکروکنترلرهای AVR به همین جا ختم نمی‌شود، از اینجا به بعد ما سعی می‌کنیم مبحث میکروکنترلر را به صورت حرفه‌ای تر دنبال کنیم.

در ادامه‌ی این جلسه برمی‌گردیم به بحث برنامه نویسی در زبان C تا کمی بیشتر با این زبان آشنا شویم.

یک نکته را باید قبل از شروع بحث متذکر شوم، در میکروکنترلر، همه‌ی ورودی‌ها **منطقی** می‌شوند. یعنی اگر سطح ولتاژ پایه‌ی ورودی (که مثلاً یک سنسور نوری به آن متصل شده است) بین ۰ تا ۲.۵ ولت باشد، آی‌سی آن را ۰منطقی در نظر می‌گیرد و اگر بین ۲.۵ تا ۵ ولت باشد، آن را ۱ منطقی در نظر می‌گیرد.

## ساختارهای کنترلی

در حالت عادی، دستورات داخل برنامه‌ی ما، از اولین دستور تا آخرین دستور به ترتیب اجرا می‌شوند. اما اگر بخواهیم بعضی از دستورات فقط تحت شرایط خاصی اجرا شوند یا مثلاً بخش‌های دیگری از برنامه چندین بار تکرار شوند، باید بتوانیم روند اجرای دستورات برنامه را کنترل کنیم. برای این منظور دستوراتی در زبان C وجود دارند که به آن‌ها دستورات یا ساختارهای کنترلی می‌گوییم.
اولین ساختار کنترلی که با آن آشنا می‌شویم دستور if است.

### if _ else:

به کمک این دستور، ما می‌توانیم برای اجرای هر بخش از برنامه شرط یا شروطی بگذاریم که اگر این شروط برقرار نباشند، میکروکنترلربدون اجرای آن دستورات از روی آن‌ها رد شود و روند اجرای برنامه به بعد از این دستورات منتقل شود. این دستور در برنامه به شکل زیر استفاده می‌شود.

```
if (شرط یا شروط)
  {
    دستوراتی که فقط اگر شروط بالا برقرار باشند اجرا می‌شوند
  }
else
  {
    دستوراتی که فقط اگر شروط بالا برقرار نباشند اجرا می‌شوند
  }
```

**نکات مهم در مورد این ساختار:**

۱- بعد از هر کدام از دستورات در داخل if و else باید حتماً ";" گذاشته شود.

۲- دقت کنید که بعد از خود if و else نیازی به ";" نیست.

۳- اگر بخواهیم چند شرط برای if بگذاریم ، به نحوی که فقط اگر **همه‌ی** شروط برقرار بودند دستورات اجرا شوند، باید به شکل زیر عمل کنیم:

if (شرط۱ && شرط۲ && شرط۳)

۴- اگر بخواهیم چند شرط برای if بگذاریم ، به نحوی اگر **هرکدام** ازشروط برقرار بودند، دستورات مربوطه اجرا شوند باید به شکل زیر عمل کنیم:

if ( شرط۱ || شرط۲ || شرط ۳)

۵- اگر بخواهیم تساوی ۲ عبارت، یا یک عبارت با یک مقدار را چک کنیم به شکل زیر عمل می کنیم:

```
if (a == b && f == 20)
```

این عبارت ۲ شرط دارد که اگر متغیر a برابر با متغیر b باشد **و** اگر متغیر f هم برابر با مقدار ۲۰ باشد، دستورات مربوط به if اجرا می‌شوند.

۶- اگر بخواهیم مقدار خروجی مثلاً یک سنسور نوری را چک کنیم که ۱ منطقی است یا نه، به صورت زیر عمل می‌کنیم:

```
if ( PORTB.1== 1 )
```
یا
```
if (PORTB.1 !== 0 )
```

این ۲ عبارت دقیقاً یک کار را انجام می‌دهند. همانطور که می‌بینید، "==!" به معنای عدم تساوی است.در حالت کلی "!"در این زبان به معنای نقیض است.

**جلسه‌ی سی و دوم**

تکمیلif-else ، ساختارwhile ، آشنایی با مبدل آنالوگ به دیجیتال (ADC)و...

در این جلسه هم بحث برنامه نویسی زبان C را دنبال می‌کنیم و در ادامه‌ی مبحث جلسه‌ی قبل، شما را با ۲ ساختار کنترلی دیگر آشنا می‌کنیم.

**ابتدا یک نکته‌ی دیگر در مورد ساختار if() و else:**

الزامی برای نوشتن قسمت else نیست، یعنی می‌توان فقط if را بدون داشتن else استفاده کرد. همانطور که می‌دانید، از else زمانی استفاده می‌کنیم که بخواهیم در صورت نادرست بودن شروط ، دستورات مشخصی اجرا شوند(به جلسه‌ی ۳۱ مراجعه شود)

## ساختار :while()

عملکرد این ساختار به این صورت است که ما شرط یا شروطی را برای آن تعریف می‌کنیم و **تا زمانیکه** این شرط یا شروط برقرار باشند، دستوراتی که تعیین می‌کنیم دائماً اجرا شوند و مکرراً تا زمانیکه شروط برقرار هستند این دستورات تکرار می‌شوند.

این ساختار به صورت زیر نوشته می‌شود.

```
while( شرط یا شروط )
{
دستور۱ ;
دستور۲ ;
دستور۳ ;
...
}
```

دستورات ۱ تا ۳ و کلاً هر دستوری که در قسمت مشخص شده نوشته شده باشد، مکرراً **تازمانیکه** شروط داخل پرانتز برقرار باشند اجرا می‌شوند.

## نکات مهم در مورد این ساختار:

تمام نکاتی که در مورد ساختار else-if در جلسه‌ی گذشته مطرح کردیم در مورد سختار while هم صادق هستند.

همانطور که قبلاً هم گفته شد، زبان C یک زبان "Case Sensitive" است، یعنی در این زبان بین حروف بزرگ و کوچک تفاوت است. while، if، else همگی با حروف کوچک نوشته می‌شوند و اگر با حروف بزرگ نوشته شوند کار نمی‌کنند.

همانطور که قبلاً گفته شد، دستور while(1) یک حلقه‌ی بی نهایت است و دستورات داخل آن تا زمانیکه میکروکنترلر روشن باشد مکرراً اجرا خواهند شد.

در ادامه‌ی این جلسه قصد داریم یکی از مهمترین و پرکاربردترین قابلیت‌های میکروکنترلر ATmega16 به نام ADC یا همان A to D را معرفی کنیم.

## ADCچیست؟

ADC مخفف "Analog-Digital Converter" و به معنای **مبدل آنالوگ به دیجیتال است**.

اگر بخواهیم این قابلیت را به صورت ساده توصیف کنیم، یک ولت متر دیجیتال است که بر روی پایه‌های میکروکنترلر نصب شده است و به وسیله‌ی آن می‌توان ولتاژ پایه‌های ورودی را با دقت مناسبی اندازه‌گیری کرد. همانطور که می‌دانید ما تا به حال در هیچ آی سی نمی‌توانستیم شدت ولتاژ ورودی را به دقت اندازه گیری کنیم و فقط می‌توانستیم بدانیم آیا ولتاژ ورودی بالای ۲.۵ ولت است یا زیر ۲.۵ ولت، و ورودی‌هایمان را به صورت ۰ و ۱ برسی می‌کردیم.(آی سی ها ورودی‌های خود را منطقی می‌کنند)

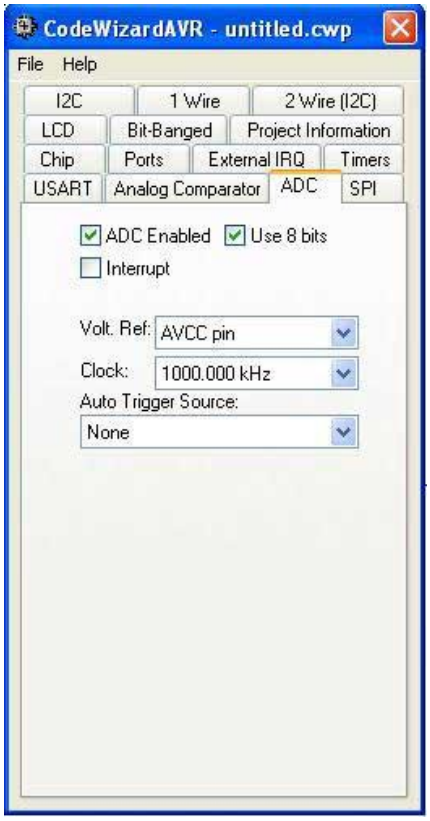
اما به کمک این قابلیتِ میکروکنترلرهایِ خانواده‌ی AVR، ما می‌توانیم با دقت مناسبی ولتاژ ورودی خود را بررسی کنیم.

یکی از کاربردهای مهم ADC می‌تواند در ربات‌های امدادگر دانش‌اموزی برای تشخیص رنگ باشد. همانطور که می‌دانید هر رنگ، میزان مشخصی از نور را بازتاب می‌دهد و بقیه را جذب می‌کند، ما با اندازه‌گیری مقدار نور بازتاب شده، میتوانیم رنگ را تشخیص دهیم. در اینجا ما برای اندازه‌گیری میزان شدت نور بازتاب شده، باید از ADC میکروکنترلر استفاده کنیم تا بتوانیم ولتاژ خروجی سنسور نوری خود را به دقت اندازه‌گیری کنیم.

ADC یکی از مهمترین و پر کاربردترین قابلیت‌های میکروکنترلرهای خانواده‌ی AVR است که به تدریج با کاربردهای فراوان آن در بخش‌های مختلف آشنا خواهید شد.

در این جلسه ما نحوه‌ی انجام تنظیمات اولیه‌ی CodeWizard برای راه‌اندازی ADC را آموزش می‌دهیم.

ابتدا CodeWizard را باز کرده و در آن، لبه‌ی ADC را انتخاب می‌کنیم، و در لبه‌ی ADC، گزینه‌ی "ADC Enabled" را تیک می‌زنیم و سپس مانند شکل زیر تنظیمات را انجام می‌دهید :




همانطور که می‌بینید، دومین قسمتی که تیک زده شده " Use ۸bits" است.در ادامه در مورد دلیل این کار توضیح داده شده است.

همانطور که گفته شد، وظیفه‌ی ADC، اندازه گیری ولتاژی است که بر روی پایه‌ی ورودی قرار گرفته است. اما ADC عددی را که مولتی متر به ما نشان می‌دهد گزارش نمی‌کند، بلکه متناسب با ولتاژ پایه‌ی ورودی، عددی را در بازه‌ی ۰ تا ۲۵۵ به ما گزارش می‌کند. یعنی عدد ۰ را به ولتاژ ۰ولت اختصاص می‌دهد، و ۲۵۵ را به ۵ ولت؛ و هر ولتاژی بین ۰ تا ۵ ولت را، متناسباً، با عددی بین ۰ تا ۲۵۵ گزارش می‌کند. در حقیقت برای سهولت در محاسبه، می توانید فرض کنید اندازه‌ی ولتاژ ورودی بر حسب ولت، ضربدر ۵۱، برابرست با عددی که ADC برای آن ولتاژ مشخص، به ما گزارش می‌کند.

به عنوان مثالی دیگر، اگر خروجی مدار یک سنسور نوری را به پایه‌ی ورودی ADC متصل کرده باشید، و خروجی مدار سنسور ۲ ولت باشد،ADC عدد ۱۰۲ را به ما گزارش می‌کند.

اگر تیکِ گزینه‌ی " Use ۸ bits" را برداریم، بازه‌ی ما به ۰ تا ۱۰۲۴، گشترش پیدا خواهد کرد و در حقیقت دقت اندازه‌گیری ما ۴ برابر خواهد شد. یعنی ۵ ولت ما، به جای ۲۵۵، با عدد ۱۰۲۴ گزارش خواهد شد، و ولتاژهای بین ۰ و ۵ ولت نیز متناسباً با عددی بین ۰ تا ۱۰۲۴ گزارش خواهند شد. اما در کارهای ما نیازی به این دقت بالا نیست و معمولاً " Use ۸ bits" را تیک می‌زنیم تا بازه‌ی ما بین ۰ تا ۲۵۵ باشد.

درباره‌ی تنظیمات ADC مطالب زیادی وجود دارد که ما از مطرح کردن تمام آن‌ها در این بخش می‌پرهیزیم، دوستانی که

علاقه‌مند هستند، می‌توانند از طریق کتاب‌های مرجع و دیگر مراجع موجود، مطلب را پی بگیرند، اما در همین حدی که مطالب در اینجا مطرح می‌شوند، برای کار ما تقریباً کافیست و الزامی در مطالعه‌ی منابع جانبی نیست.

در جلسه‌ی آینده در مورد نحوه‌ی دریافت این عددی که قرار است ADC به ما گزارش دهد توضیح خواهیم داد.

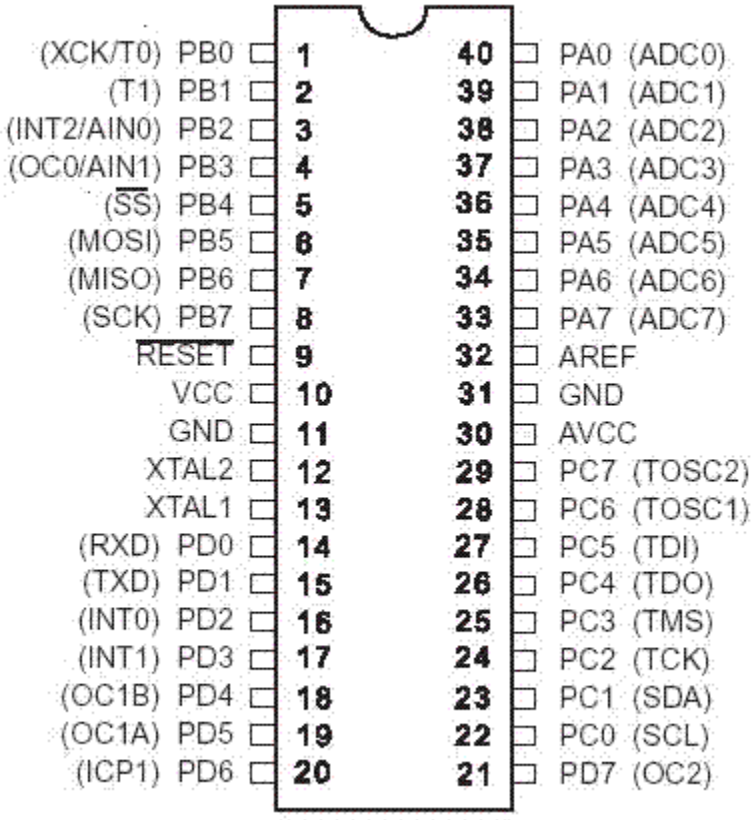
### جلسه‌ی سی و سوم

نحوه‌ی استفاده از ADC در برنامه نویسی...

در جلسه ی گذشته در مورد ساختار ADC یا همان مبدل آنالوگ به دیجیتال توضیحات مفصل داده شد. در این جلسه، نحوه ی استفاده از ADC در برنامه را توضیح خواهیم داد.
در میکرو کنترلر ATMEGA16، ۸ پایه ی به ADC اختصاص داده شده، یعنی شما می توانید به صورت همزمان، خروجیِ ۸ سنسور یا مدار جانبی را به میکروکنترلر خود وصل کنید و اطلاعات آن ها را به وسیله ی ADC دریافت کنید.

**اما این ۸ پایه کدام پایه ها هستند؟**

این ۸ پایه، پایه مربوط به پورت A هستند که با فعال کردن ADC در CodeWizard، این پایه ها در اختیار ADC قرار می گیرند. دقت کنید که برای استفده از ADC حتماً باید قبلاً تنظیمات را در CodeWizard انجام داده باشید.
این ۸ پایه طبق شکل زیر از ADC0 تا ADC7 نام گذاری شده اند.




## چگونه از ADC در برنامه استفاده کنیم؟

هنگامیکه شما تنظیمات اولیه را برای فعال سازی ADC در CodeWizard انجام می دهید، به شما این امکان داده می شود که در هر قسمت از برنامه، مقدار هر یک از ۸ ADC را بخوانید و از آن استفاده کنید.

عبارت " ADC (read_adc (شماره ی " در هر قسمت از برنامه، حاوی مقدار ADC مورد نظر شماست. مثلاً اگر بخواهیم مقدار ADC0 را بررسی کنیم که آیا از ۱۰۰ بزرگتر است یا خیر، به شکل زیر عمل می‌کنیم:

```
if(read_adc(0) > 100)
```

یا مثلاً اگر بخواهیم مقدار ADC۱ را در داخل یک متغیر نگه داری کنیم، به شکل زیر عمل می کنیم:

```
int a;
a= read_adc(1);
```

مبدل آنالوگ به دیجیتال، یکی از مهمترین قابلیت‌هایست که اکثر میکروکنترلرهای خانواده‌ی AVR آن را دارند. در میکروکنترلرهای قدیمی‌تر، برای این کار، باید یک آی سی به صورت جداگانه بر روی مدار تعبیه می‌شد.

در جلسه‌ی آینده مبحث بسیار مهم PWM را شروع خواهیم کرد. به وسیله‌ی این قابلیت بسیار مهم،ما می‌توانیم سرعت موتورها یا هر المان دیگری که به میکروکنترلر متصل می‌شود را کنترل کنیم. PWM نیز یکی از قابلیت‌های مهم خانواده‌ی AVR است که در جلسات آینده مفصلاً به آن خواهیم پرداخت.

### جلسه‌ی سی و چهارم

توضیحات در مورد بخش مکانیک ربات، موتورهای گیربکس دار، شاسی ربات و....

بنا به درخواست بسیاری از دوستان، ما قبل از شروع بحث PWM در AVR، ابتدا کمی به

## شاسی ربات

قسمت‌های مکانیکی ربات می‌پردازیم.

شاسی در ربات‌های مختلف بسته به کاربری ربات، ممکن است از جنس‌های مختلفی باشد که محدود به چند نوع فلز یا آلیاژ خاص نیست، اما به طور معمول در ربات‌های مسابقاتی دانش‌آموزی معمولاً شاسی ربات‌ها را از جنس‌های پلکسی گلاس، صفحه‌ی آلمینیوم و یا MDF می‌سازند.

جهت آشنایی دوستان با این انواع در مورد هر کدام توضیح مختصری بیان می‌شود.

## پلکسی گلاس:

نوعی پلاستیک فشرده است که نسبت به حجمش استحکام خوبی دارد. همچنین نسبت به فلزات وزن بسیار کمتری دارد. نوع بی رنگ آن کاملاً شبیه شیشه است، اما بسیار سبکتر از آن است. همچنین مانند شیشه در ضخامت‌های مختلفی در بازار موجود است. شکننده است و مانند فلزات انعطاف پذیری ندارد. تنها راه برای انعطاف دادن به آن اعمال حرارت بالا توسط آتش مستقیم یا ... است. برای بریدن آن می‌توان از اره مویی استفاده کرد، اما راه بهتر و راحت‌تر، استفاده از کاتِر مخصوص پلکسی گلاس است. از همان جایی که پلکسی گلاس را تهیه می‌کنید، می‌توانید کاتِر مخصوص آن را هم تهیه کنید. چگونگی استفاده از آن را هم از فروشنده سوال کنید.
پلکسی گلاس در ضخامت‌های مختلف موجود است. برای ربات مین یاب یا مسیریاب ضخامت ۴ یا ۵ میلیمتر مناسب است.
می‌توان از پلکسی گلاس مشکی یا دودی هم استفاده کرد که موجب زیبایی بیشتر ربات می‌شود.

## MDF

ممکن است بسیاری از دوستان با این نوع آشنایی داشته باشند،زیرا در تهیه‌ی کابینت ، کمد و بسیاری از اساس منزل استفاده می‌شود. این نوع، از ترکیب براده‌های چوب با نوعی چسب تولید می‌شود(مشابه نئوپان) و نسبت به چوب‌های معمولی استحکام بیشتری دارد. هیچگونه انعطافی ندارد، و می‌توان با اره برقی و معمولی آن را برید.

MDF در ضخامت‌های مختلفی وجود دارد که طبیعتاً هرچه ضخامت آن بیشتر باشد،استحکام و وزن آن نیز بالاتر می‌رود.

MDF 8 میلیمتری برای شاسی ربات‌های مسیریاب پیشرفته و آتش نشان مناسب است، زیرا برای ساخت این ربات‌ها محدودیت زیادی برای حجم نداریم، و استحکام بسیار خوبی هم دارد.

با اینکه تقریباً پلکسی گلاس از هر نظر از MDF مناسب‌تر است، اما کمی هم از MDF پرهزینه تر است و به همین خاطر MDF هنوز کاربرد زیادی در ساخت ربات‌ها دارد.

## صفحه‌ی آلمینیومی

در ربات‌های فوتبالیست دانش‌اموزی، به دلیل برخوردهای شدیدی که گاهاً ممکن است بین ربات‌ها پیش آید و فشاری که به بدنه‌ی ربات وارد می‌شود،معمولاً شاسی ربات را از جنس صفحه‌ی آلمینیوم ۲ یا ۳ میلیمتری می‌سازند، این امر موجب استحکام بسیار بالای بدنه می‌شود.MDF نیز استحکام مناسبی دارد، اما از لحاظ حجمی، حجم صفحه‌های فلزی بسیار کمتر از MDF است.تنها ایراد صفحه‌ی آلمینیوم، وزن زیاد آن است ممکن است کار را دچار مشکل کند، از این رو دوستان باید در استفاده از آن دقت لازم را داشته باشند.

## چگونه موتورها را به بدنه متصل کنیم؟

چند راه برای اتصال موتورها به بدنه‌ی ربات وجود دارد. یکی از ساده‌ترین و سریع‌ترین روش‌ها برای اتصال موتورهای گیربکس دار به بدنه ، استفاده از بَست دیوارکوب لوله‌ی آب است. به شکل زیر دقت کنید.


## گیربکس چیست و چه کاربردی در ساخت ربات دارد؟

برای سرعت موتور کمیتی به نام rpm یا "دور بر دقیقه" تعریف می‌شود که این کمیت، تعداد چرخش شفت موتور را در مدت یک دقیقه نشان می‌دهد.موتورهای عادی بدون گیربکس rpm بالا و قدرت کمی دارند. rpm بالا موجب بالا رفتن سرعت ربات می‌شود. قدرت کم و سرعت زیاد،در مجموع موجب غیر قابل کنترل شدن ربات می‌شود و هدایت ربات را دچار مشکل می‌کند.

rpm رسانده است.(در تصویر بالا گیربکس زیر بَست قرار گرفته است). در این موتور، سرعت شفت موتور قبل از اتصال به گیربکس ۵۲۸۰ rpm بوده است. این سرعت برای یک ربات مسیر یاب مقدماتی بسیار بالاست. سرعت موتور ربات‌های ما باید زیر ۲۰۰ rpm باشند.

تمام اطلاعات مربوط به موتور را معمولاً شرکت‌های معتبر موتورسازی بر روی بدنه‌ی موتور می‌نویسند.

انواع موتورهای گیربکس‌دار با سرعت‌ها و قدرت‌های مختلف در حال حاضر در بازار موجود است. یکی از راه‌های ارزان برای تهیه‌ی این موتورهای گیربکس دار،برای پروژه‌های ساده، استفاده از موتور ماشین اسباب‌بازی‌های ارزان قیمت ساخت چین است. قیمت این ماشین‌ها زیر ۲۰۰۰ تومان بوده و از هر اسباب بازی می‌توان یک موتور و گیربکس را به همراه چرخ آن استخراج کرد. به شکل نگاه کنید.

در جلسه‌ی آینده به مبحث میکروکنترلر برمی‌گردیم و نحوه‌ی کنترل سرعت موتورها را از طریق PWM در میکروکنترلرهای خانواده‌ی AVR بررسی می‌کنیم.جلسه‌ی آینده جمعه‌ی هفته‌ی آینده بر روی سایت قرار خواهد گرفت.

**جلسه‌ی سی و پنجم**

توضیح مقدماتی در موردPWM ، کاربرد و نحوه‌ی تولید آن برای ولتاژهای مختلف، latchکردن و...

در این جلسه به مبحث میکروکنترلر برمی‌گردیم و در مورد PWM و کاربردهای آن در ساخت ربات توضیح خواهیم داد.

## PWMچیست؟

در بسیاری از موارد، ما نیاز به کنترل ولتاژ بر روی پایه‌های خروجی میکروکنترلر را داریم. مثلاً اگر بخواهیم سرعت موتور را کنترل کنیم، باید ولتاژی که بر روی موتور اعمال می‌شود را کنترل کرد. در حقیقت سرعت موتور تقریباً تابع مستقیمی از ولتاژی است که بر روی آن اعمال می‌شود. یعنی اگر ولتاژ کاریِ موتوری (ولتاژ استاندارد برای فعال سازی موتور که بر روی بدنه‌ی آن نوشته می‌شود) ۱۲ ولت باشد، با اعمال ولتاژ ۶ ولت روی آن، می‌توانید سرعت چرخش آن(rpm) را حدوداً به نصف کاهش دهید.

کنترل **سرعت** ربات، در همه‌ی سطوح رباتیک اهمیت بسیار زیادی دارد، از ربات‌های مسیریاب ساده گرفته تا ربات‌های فوتبالیست. ما تا کنون یاد گرفته‌ایم که چگونه می‌توان به موتور دستور حرکت یا توقف داد، اما راهی برای کنترل سرعت موتور یاد نگرفته‌ایم.

**یادآوری**

همانطور که می‌دانید سطح ولتاژ پایه‌های خروجی میکروکنترلر منطقی است، یعنی یک پایه‌ای که برای کنترل موتور ربات استفاده می‌شود فقط می‌تواند 0 یا 1 باشد. ما ۲ پایه از میکروکنترلر را به حرکت ربات اختصاص می‌دهیم، برای صدور دستور حرکت، باید یک پایه را 0 و پایه‌ی دیگر را 1 کنیم، در این حالت بین ۲ پایه‌ی موتور اختلاف پتانسیل برقرار می‌شود و حرکت می‌کند. اگر هم بخواهیم موتور معکوس بچرخد، باید پایه‌ای که 1 بود 0 ، و پایه‌ای که 0 بود را 1 کنیم؛ و برای توقف موتور، باید هر دو پایه را 0 یا هر دو پایه را 1 کنیم (تا بین ۲ پایه‌ی موتور اختلاف پتانسیل ۰ ولت باشد). در نتیجه در حالت عادی ما فقط ۲ فرمان "حرکت" و "توقف" را می‌توانیم به موتورها بدهیم، و ما هیچ کنترلی بر روی **سرعت** موتور نداریم.

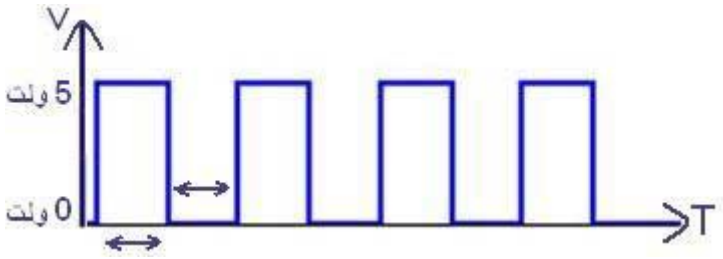
PWM تکنیکی است که به کمک آن می‌توانیم ولتاژ پایه‌های خروجی میکروکنترلر، و در نتیجه سرعت موتور یا سایر قطعات جانبی که به میکروکنترلر متصل می‌شود را کنترل کنیم.

## PWM

PWM مخفف واژه‌ی Pulse Width Modulation و به معنای "مدولاسیون پهنای پالس" است. همانطور که گفتیم PWM تکنیکی برای کنترل ولتاژِ پایه‌ی خروجی است. حال ببینیم چگونه با این تکنیک می‌توان ولتاژ خروجی را کنترل کرد.

می‌دانیم که ولتاژ در پایه‌های خروجی میکروکنترلر یا ۰ است یا ۵ ولت، اما برای کنترل **سرعت** موتور، باید بتوانیم حداقل ولتاژ یکی از پایه‌ها را بین ۰ تا ۵ تغییر دهیم. PWM روشی است تا ما بتوانیم با استفاده از همین پایه‌ی خروجی معمولی، به نوعی ولتاژ را بین ۰ تا ۵ ولت تغییر دهیم.

در این روش، ما با سرعت بالایی سطح ولتاژ خروجی را ۰ و بلافاصله ۱ می‌کنیم(مثلاً هزار بار در ثانیه)، نمودار ولتاژ خروجی بر حسب زمان به شکل زیر می‌شود.




نمودار بالا ولتاژ خروجی این پایه بر حسب زمان است.

در شکل بالا جمع ۲ بازه‌ای که با فلش‌های ۲طرفه نشان داده شده است، (به عنوان مثال) ۱۰ میکرو ثانیه است. که ۵میکرو ثانیه خروجی ۱ و سپس ۵میکرو ثانیه ۰ می‌شود. اما همانطور که گفته شد، این عمل هزاران بار در ثانیه تکرار می‌شود، اما آیا موتور نیز به همین تعداد در ثانیه روشن و خاموش می‌شود؟

جواب منفیست، اتفاقی که روی می‌دهد این است که موتور، این موج را در درون خود به نوعی میانگین گیری می‌کند و در حقیقت آنرا به شکل زیر می بیند:


یعنی در واقع موتور این موج را به صورت یک ولتاژ ۲.۵ ولت معمولی دریافت می‌کند.

به همین ترتیب می‌توان هر ولتاژی بین ۰ تا ۵ ولت را بر روی خروجی مورد نظر ایجاد کرد. اگر بخواهیم ولتاژی بالاتر از ۲.۵ ولت داشته باشیم، باید طول بازه‌های زمانی‌ای که خروجی ۱ است را نسبت به بازه‌هایی که خروجی ۰ است بیشتر کنیم.

به عنوان مثال برای ایجاد ولتاژ ۲.۵ ولت، باید ۵ میکرو ثانیه سطح ولتاژ خروجی ۱ باشد، سپس ۵ میکرو ثانیه سطح ولتاژ ۰ شود تا موجی به شکل بالا ایجاد شود.

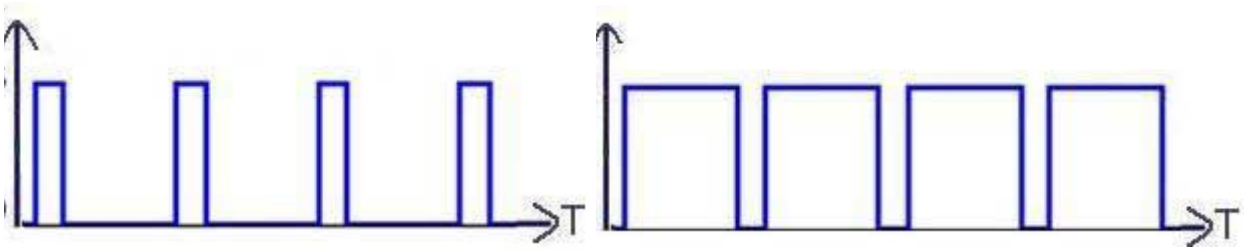
یا به عنوان مثالی دیگر، اگر بخواهیم در خروجی ولتاژ ۴ ولت داشته باشیم، باید باید ۸ میکرو ثانیه سطح ولتاژ خروجی ۱ باشد، سپس ۲ میکرو ثانیه سطح ولتاژ ۰ شود، تا ولتاژ پایه‌ی خروجی مورد نظر ۴ ولت باشد.

در حقیقت ولتاژ خروجی از رابطه‌ی ساده‌ی زیر به دست می‌اید:

**(طول کل بازه) / ( طول بازه‌ای که خروجی 1 است)**

پس طبق رابطه‌ی بالا،برای ایجاد ولتاژ ۴ ولت، می‌توان به جای استفاده از بازه‌های ۸ و ۲ میکرو ثانیه‌ای، از بازه‌های ۴ و ۱ میکرو ثانیه‌ای استفاده کرد. (یعنی ۴میکرو ثانیه ۵ولت، ۱ میکرو ثانیه ۰ ولت) زیرا: ۲÷ ۸ = ۱÷۴

نمودار ولتاژهای ۴ ولت و ۱ ولت در زیر نشان داده شده است:




نمودار نحوه‌ی تولید ولتاژ ۴ ولت با تکنیک.PWM  نمودار نحوه‌ی تولید ولتاژ ۱ ولت با تکنیک.PWM

حال ببینیم چگونه می‌توان برنامه‌ای نوشت تا بر روی پایه‌ای دلخواه از میکروکنترلر PWM ی برای ولتاژ ۴ ولت ایجاد کرد.
هر دستوری که بر روی خروجی‌های میکروکنترلر قرار می‌گیرد، تا زمانی که دستور بعدی، خروجی را تغییر ندهد، آن خروجی تغییری نخواهد کرد. یعنی مثلاً زمانی که پایه‌ای را ۱ می‌کنیم، تا زمانیکه با دستور دیگری آن پایه را ۰ کنیم، مقدار خروجی آن پایه ۱ خواهد ماند. به این عمل اصطلاحاً Latch کردن می‌گویند. میکروکنترلر همواره اطلاعاتی که بر روی خروجی قرار می‌دهد را Latch می‌کند و تا زمانیکه اطلاعات جدید بر روی پایه قرار نگیرد، اطلاعات قبلی را تغییر نمی‌دهد.
در نتیجه، مثلاً اگر می‌خواهیم پایه‌ای را ۵ میکروثانیه ۱ وسپس ۰ کنیم، کافیست پایه‌ی مورد نظر را ۱ کنیم و ۵میلی ثانیه در برنامه تاخیر ایجاد کنیم و سپس پایه‌ی مورد نظر را ۰ کنیم.
پس وقتی می خواهیم مثلاً بر روی پایه‌ی .۴B ، یک PWM برای ولتاژ ۲.۵ ولت ایجاد کنیم، باید به شکل زیر عمل کنیم.

```
while(1)
{
PORTB.4=1;
5 میکرو ثانیه تاخیر
PORTB.4=0;
5 میکرو ثانیه تاخیر
}
```

در بالا یک حلقه‌ی بی‌نهایت تعریف شده است که بر روی پایه‌ی.B.4، یک PWM برای ۲.۵ ولت ایجاد می‌کند.

در جلسه‌ی آینده با توابعی که برای ایجاد تاخیر (delay) در برنامه استفاده می‌شوند آشنا خواهید شد. همچنین می‌آموزید که

چگونه می‌توان از PWM میکروکنترلرهای خانواده‌ی AVR استفاده کنید.

اگر مطالب این جلسه کمی پیچیده به نظر می‌رسند جای نگرانی نیست، زیرا مبحث PWM مبحث گسترده‌ایست و کاربردهای زیادی در صنعت دارد و فقط محدود به میکروکنترلر هم نمی‌شود. پس طبیعیست که سطح مطالب کمی بالا باشد و دوستان نیز گاهاً در درک مفاهیم دچار مشکل شوند.

### جلسه‌ی سی و ششم

ادامه‌ی مبحثPWM ، توابع ایجاد وقفه(Delay) ، رجیسترهای OCRx و...
در این جلسه نیز مبحث PWM را دنبال می‌کنیم. عزیزان برای درک مطالب این جلسه حتماً باید جلسه‌ی قبل را مطالعه کرده باشند. در این جلسه به آی‌سی L298 نیز اشاراتی شده است که دوستان می‌توانند جهت یاداوری، بخش مربوط به L298 در جلسه‌ی ۱۷ را نیز مرور کنند.

ابتدا با توابعی که برای ایجاد وقفه در اجرای دستورات برنامه توسط CodeVision برای کاربران در نظر گرفته شده آشنا می‌شویم.
همان‌طور که در جلسه‌ی پیش دیدیم، در قسمت‌هایی از برنامه ممکن است نیاز پیدا کنیم تا برای لحظاتی روند اجرای دستورات را متوقف کنیم CodeVision .برای این کار توابعی را از پیش تنظیم کرده است. (در مورد مبحث« توابع »در زبان C در آینده مفصل توضیح خواهیم داد.)

## delay

برای ایجاد تاخیر در روند اجرای دستورات، CodeVision دو تابع زیر را در اختیار ما قرار داده است.

```
delay_ms( );
delay_us( );
```

تابع (delay_ms) برای ایجاد تاخیرهایی در حد میلی ثانیه به کار می‌رود. در داخل پرانتز، یک عدد صحیح مثبتمی‌نویسیم که نشان دهنده‌ی اندازه‌ی تاخیر مورد نیز ما بر حسب میلی ثانیه است. به بیان ساده‌تر، مثلاً اگر داخل پرانتز عدد 100 را بنویسیم، روند اجرای برنامه به اندازه‌ی 100 میلی ثانیه در همان خط متوقف خواهد شد.
تابع (delay_us) برای ایجاد تاخیرهایی در حد میکروثانیه به کار می‌رود. نحوه‌ی استفاده از آن دقیقاً مانند (delay_ms)است.
به عنوان یک مثال عملی،همان برنامه‌ی ایجاد PWM 2.5 ولت را با استفاده از توابع delay بازنویسی می‌کنیم.

```
while(1)
{
PORTB.4=1;
delay_ms(5);   // 5 milliseconds delay
PORTB.4=0;
delay_ms(5);    //5 milliseconds delay
}
```

تنها نکته‌ی بسیار مهم در استفاده از توابع delay اضافه کردن هِدِرفایل Header file باعنوان delay.h به برنامه است.( در مورد هدر فایل‌ها هم در آینده توضیح خواهیم داد، اما در این جلسه هیچ توضیحی در مورد آن نمی‌دهیم تا از بحث اصلی یعنی PWM منحرف نشویم.) برای این کار، جمله

```
#include<mega16.h>
```

که اولین جمله‌ی برنامه‌ی شما است را پیدا کنید. (این جمله را CodeWizard در برنامه‌ی شما نوشته است). حال کافیست این جمله را درست زیر ان تایپ کنید:

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

دقت کنید که این دستور نیازی به « ; » ندارد!!

با آموختن تابعdelay ، دیگر شما می‌توانید هر ولتاژی را که می‌خواهید بر روی پایه‌های خروجی ایجاد کنید. البته دوستان دقت داشته باشند که ولتاژی که با تکنیک PWM شبیه سازی می‌شوند، در حقیقت ولتاژ خاصی نیستند و فقط شبیه سازی شده‌ی ولتاژهای مختلف هستند. هرچند که در راه‌اندازی موتورها این تکنیک بسیار کارآمد است، اما باید دقت نظر لازم را در استفاده از این تکنیک در سایر موارد را داشته باشید.

همانطور که می‌دانید موتورهای متعارفی که برای ساخت ربات‌ها استفاده می‌شود،ممکن است ولتاژهای کاری مختلفی داشته باشند (مثلاً ۱۲ولت، ۲۴ ولت، ۶ ولت و ...) و برای راه‌اندازی آن‌ها باید از درایورهای موتور مثلL298 استفاده کنیم. سوالی که ممکن است پیش آید این است که وقتی ما میکروکنترلر را به درایورهای موتور(مثل L298) وصل می‌کنیم و از تکنیک PWM برای کنترل سرعت موتور استفاده می‌کنیم، چه وضعیتی پیش می‌آید؟ مثلاً وقتی ما PWMمربوط به ولتاژ 2.5 ولت را تولید می‌کنیم، درایور ما چه عکس العملی نشان می‌دهد؟ آیا ولتاژ ۲.۵ ولت بر روی پایه‌های موتور قرار می‌گیرد؟

برای پاسخ دادن به این سوال باید به ساختار PWM دقت کنیم، ما وقتی PWM مربوط به ۲.۵ ولت را تولید می‌کنیم، در حقیقت سطح ولتاژ خروجی را با فواصل زمانی برابر ۰ و ۱ می‌کنیم، پس اگر این خروجی را، به ورودی L298 وصل کنیم(مثلاً پایه‌ی ۷(، L298 نیز موتور را با همین الگو کنترل می‌کند و ولتاژی که به موتور می‌دهد را ۰ و ۱ می‌کند. و همانطور که می‌دانید، L298 هر ولتاژی که بر روی پایه‌ی شماره‌ی ۴ آن قرار گرفته باشد را بر روی موتور قرار می‌دهد(اگر ولتاژ کاری موتور

۱۲ ولت باشد، باید این پایه به ۱۲ ولت متصل شود) . پس جواب سوال بالا منفیست!!! وقتی ما PWM مربوط به ۲.۵ ولت را تولید می‌کنیم، در حقیقت سطح ولتاژ خروجی در ۵۰ درصد زمان ۱ و بقیه‌ی زمان ۰ است.

پس اگر همان طور که در بالا اشاره شد، این PWM به درایوری مثل L298 داده شود، و ولتاژ پایه‌ی ۴ ِ آن ۱۲ولت باشد، درایور، ولتاژ ۶ ولت را به موتور می‌دهد. در نتیجه اهمیتی ندارد چه ولتاژی بر روی پایه‌ی ۴ ِ L298 قرار گرفته باشد، وقتی که ما PWM مربوط به ۲.۵ ولت را تولید می‌کنیم، درایور ولتاژی که به موتور می‌دهد را ۵۰ درصد می‌کند. در نتیجه بهتر است از این به بعد به جای آن که بگوییم PWM مربوط به ۲.۵ ولت، بگوییمPWM ||| 50درصد .یا به جای PWM ِمربوط به ۱ ولت، بگوییمPWM ||| 20درصد

## PWMدر میکروکنترلرهایAVR

انجام تنظیمات اولیه برای استفاده از PWM برای راه اندازی موتور درمیکروکنترلرهای AVR کمی پیچیده است، اما در اینجا هم CodeWizard به کمک ما آمده است و کار را کمی ساده‌تر کرده است. ما در جلسه‌ی آینده بخشی از تنظیمات CodeWizard را بدون توضیح مطرح می‌نماییم، زیرا توضیح هر بخش از آن نیازمند مقدمات مفصلی است و تاثیر چندانی هم در روند کار ما ندارد، اما به دوستانی که می‌خواهند میکروکنترلر را کاملاً حرفه‌ای دنبال کنند، پیشنهاد می‌کنم از منابعی که قبلاً معرفی شده است، مطالب را تکمیل کنند .
به هر حال دوستان عزیز با انجام این تنظیمات اولیه‌ی مختصر درCodeWizard ، می‌توانند از الگویی به مراتب ساده‌تر از آنچه تا به حال آموخته‌ایم، برای ایجاد PWM برای هدایت موتورهای ربات استفاده نمایند.

در میکروکنترلرهای خانواده‌یAVR ، نیازی نیست در هربار استفاده از PWM ، چندین خط برنامه بنویسیم.
در ATmega16چهارپایه‌ی مشخص از آی سی به این موضوع اختصاص داده شده است. یعنی این چهارپایه علاوه بر کاربردهای معمولی خود، این قابلیت را دارند که در مواقع لزوم برای تولید PWM استفاده شوند.
حال سوال اینجاست که این چهارپایه چه تفاوتی با بقیه‌ی پایه‌های خروجی آی‌سی دارند که آن‌ها را از سایر پایه‌های خروجی میکروکنترلر متمایز می‌سازد؟
برای این چهارپایه نیازی به اجرای الگویی که تا به حال برای ایجاد PWM فراگرفته‌اید نیست. در این روش، فقط شما باید یک عدد صحیح بین ۰ تا ۲۵۵ انتخاب کنید، و طبق الگوی زیر آن را در برنامه‌ی خود بنویسید .

**یک عدد صحیح بین ۰ تا ۲۵۵ = نام رجیستر مربوطه;**

این عدد، بیانگر توان PWM شماست، و شما توان PWM ِ مورد نیاز خود را با این عدد مشخص می‌کنید. که ۲۵۵ بالاترین توان و مربوط بهPWM||| 100درصداست، و ۰ پایین‌ترین توان و مربوط بهPWM ||| 0درصد است .
به عنوان مثال اگر این عدد را ۱۲۸ قرار دهید، همانPWM ||| 50درصد را ایجاد کرده‌اید. یا مثلا اگر این عدد ۵۱ باشد،PWM||| 20درصد بر روی پایه قرار داده‌اید.

رجیسترهای مربوط به این ۴ پایه

همانطور که می‌دانید، برای پایه‌هایی که در CodeWizard به صورت خروجی تعریف شده‌اند، رجیستری به نام «PORTx»وجود دارد که هر مقداری در این رجیستر قرار داده شود، مقدار پایه‌های خروجی متناظر با آن رجیستر را مشخص می‌کند.(رجوع به جلسه‌ی ۲۴،تعریف رجیستر PORTx ).

در این جلسه با ۴ رجیستر دیگر آشنا می‌شویم، که وقتی تنظیمات مربوط به PWM موتور در CodeWizard را انجام دهیم، هر مقداری که در آن‌ها ریخته شود، توان PWM پایه‌ی متناظر را مشخص می‌کنند .

این رجیسترهاOCR0 ، OCR1AL، OCR1BLو OCR2 نام دارند که به ترتیب، متناظر پایه‌هایPB.3 ، PD.5، PD.4و PD.7هستند.

پس مثلاً اگر در بخشی از برنامه‌ی خود بنویسیم:

```
OCR0=127;
```

در حقیقت بر روی پایه‌ی PB.3 میکروکنترلر،PWM ااا-50درصد به وجود آورده‌ایم.

به مثال‌های دیگری توجه کنید: (توضیح هر دستور در جلوی دستور و بعد از // آورده شده است(

```
OCR1AL=51; // 20% Duty Cycle on PD.5
OCR1BL=255; //100% Duty Cycle on PD.4
OCR2=0;  //0% Duty Cycle on PD.7
```

در جلسه‌ی آینده، در مورد نحوه‌ی انجام تنظیمات اولیه جهت تولید PWM در CodeWizard را توضیح خواهیم داد.

**جلسه سی و هفتم**

در این جلسه، با توضیح در مورد نحوه‌ی انجام تنظیمات اولیه درCodeWizard ، مطلب راه اندازی موتورهای ربات به کمک PWMرا تکمیل خواهیم کرد...

در این جلسه، با توضیح در مورد نحوه‌ی انجام تنظیمات اولیه درCodeWizard ، مطلب راه اندازی موتورهای ربات به کمک PWM را تکمیل خواهیم کرد.

همانطور که در جلسه‌ی پیش هم متذکر شدیم، در اینجا مجال نیست تا تمام مباحث مربوط به PWM و تایمرها را باز کنیم و مفصل به آن‌ها بپردازیم، به همین خاطر در این جلسه قسمتی از تنظیمات در CodeWizard را بدون توضیح آموزش می‌دهیم.

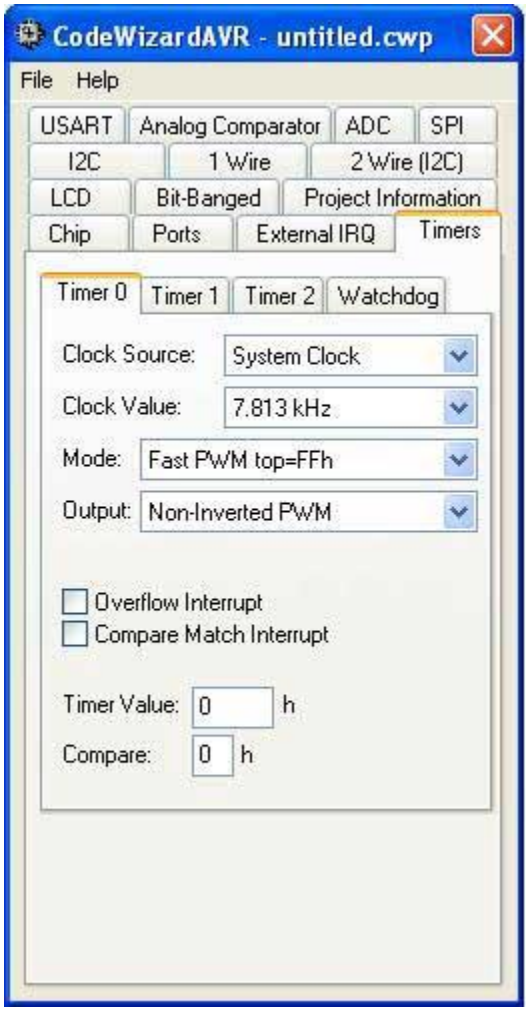
برای انجام تنظیمات به کمکCodeWizard ، ابتدا باید طبق روندی که در جلسه‌ی ۲۶ توضیح داده شد، یک پروژه‌ی جدید در CodeVision بسازید.

پس از انجام تنظیمات سایر لبه‌ها (مانندPorts ، Chipو .... ) درCodeWizard ، لبه‌ی Timers را باز کنید.

همانطور که می‌بینید میکروکنترلر ATmega16 دارای ۳ تایمر مجزا است و ما برای تولید PWM باید از این تایمرها استفاده کنیم. تایمرها کاربردهای متعددی دارند، و یکی از مهم‌ترین مباحث در میکروکنترلر هستند، ما هم در مورد تایمرها در جلسات آینده مفصل توضیح خواهیم داد. اما در این جلسه فقط استفاده از تایمرها را برای ایجادPWM برای کنترل موتورهای ربات استفاده می‌کنیم.

## Timer0

Timer0مربوط به رجیستر OCR0 است و باید به شکل زیر تنظیم شود.



نکته‌ای که در مورد تنظیم هر ۳ تایمر باید رعایت شود، این است که در بخش "Clock Value" باید پایین‌ترین فرکانس را انتخاب کنید. در این مورد توضیح مختصری می‌دهم، ولی اگر عزیزان این بند را متوجه نشوند اهمیت زیادی ندارد: اندازه‌ی فرکانسی که انتخاب می‌کنید در این بخش، در حکم اندازه‌ی همان Delay هایی است که برای تولید PWM به صورت عادی (که در ابتدای جلسه‌ی قبل توضیح دادیم) استفاده می‌کنیم. یعنی در حقیقت طول موج را در نمودار ولتاژ بر زمان تعیین می‌کند. هر چه فرکانس بالاتری را انتخاب کنید، طول موج کمتر می‌شود. در عمل دیده شده که هر چه فرکانس پایین‌تر باشد و در نتیجه طول موج بیشتر باشد، موتورها بهتر هدایت می‌شوند. به همین خاطر در بالا گفته شد که دوستان پایین‌ترین فرکانس را برای "Clock Value" انتخاب کنند.

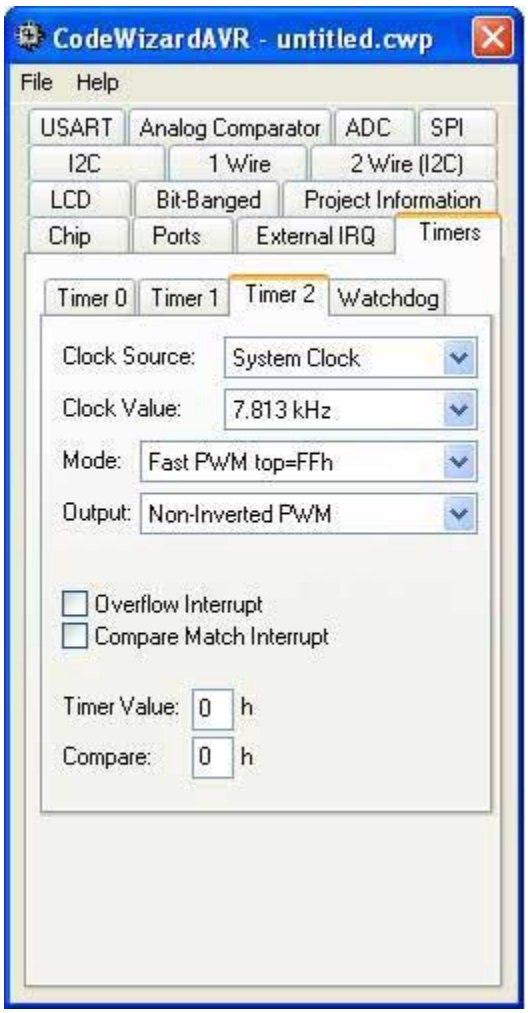
## Timer1

تایمر ۱ باید به شکل زیر تنظیم شود. دقت کنید که ممکن است در بخش Clock Value شما فرکانسی که در شکل زیر نمایش داده شده است را در گزینه‌ها نداشته باشید، ولی همانطور که گفته شد فقط مهم این است که شما پایین‌ترین فرکانس را انتخاب کنید.


همانطور که می‌بینید، تایمر۱ دارای دو خروجی مجزا است که رجیسترهای مربوط به آن‌ها OCR1AL وOCR1BL هستند.

## Timer2

تایمر۲ میز به شکل زیر تنظیم می‌شود و مانند تایمر۰ فقط یک خروجی دارد.




## Watchdog

یا سگ نگهبان نیز یکی از مباحث مربوط به تایمرهاست که در جلسات آینده به آن خواهیم پرداخت.

حال که همه‌ی تنظیمات لازم را در CodeWizard انجام داده‌اید، "Generate, Save and Exit" انتخاب کنید و وارد فضای برنامه نویسی شوید.

**نکته‌ی بسیار مهم:**

برای کنترل هر موتور، علاوه بر یک پایه‌یPWM ، یک خروجی معمولی نیز لازم داریم تا بتوانیم به وسیله‌ی این دو پایه و به کمک درایور موتور، اختلاف پتانسیل مورد نظر را بر روی دو پایه‌ی موتور برقرار کنیم. این ۲ پایه را به دو پایه‌ی ورودی L298 متصل می‌کنیم و دو پایه‌ی موتور را نیز، به دو

پایه‌ی خروجی L298 متصل می‌کنیم. حال می‌توانیم موتور را به وسیله‌ی میکروکنترلر با سرعت دلخواه کنترل کنیم. به عنوان مثال اگر بخواهیم موتور ما تقریباً با سرعت نصف بچرخد، و پایه‌های PD.6 و)PD.7 مربوط به رجیستر (OCR2 را به L298 متصل کرده باشیم،برنامه‌ی زیر را باید بنویسیم:

```
OCR2=127;
PORTD.6=0;
```

و اگر بخواهیم موتور ما با همین سرعت و در جهت معکوس بچرخد، می‌نویسیم:

```
OCR2=127;
PORTD.6=1;
```

برای درک این موضوع دقت کنید که در این حالت چه ولتاژی توسط L298 بر روی موتورها قرار داده می‌شود. همانطور که می‌دانید، سرعت و جهت چرخش موتور وابسته به اختلاف ولتاژی است که بر روی پایه‌های موتور قرار داده می‌شود.

تا به اینجا مباحث پایه‌ای در میکروکنترلرهای AVR مطرح شده است و همین آموخته‌های دوستان، نیازهای اولیه‌ی شما عزیزان را برای ساخت ربات‌های نسبتاً حرفه‌ای برطرف می‌سازد.
از این به بعد، مطالب به با دامنه‌ی گسترده‌تری پیرامون سایر مباحث مربوط به رباتیک،ارائه خواهد شد. در هر جلسه مطلب جدیدی را مطرح کرده و در مورد آن توضیح می‌دهیم و دیگر مطالب به شکل کنونی به صورت زنجیره‌وار و وابسته به هم، نخواهند بود.
طبیعتاً برای درک هر مطلبی نیاز به مقدمات و پیش‌نیازهایی است، که در ابتدای هر جلسه پیش نیازهای مطالبی که در ان جلسه قرار است مطرح شوند، ذکر خواهد شد، تا به این ترتیب پراکندگی مطالب، دوستان را دچار سر در گمی نکند.

### جلسه‌ی سی و هشتم

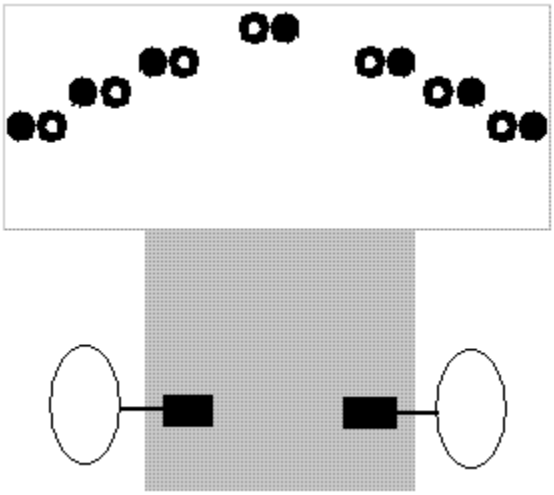
کاربرد تکنیک PWM در ساخت یک ربات مسیریاب پیشرفته...

در این جلسه به دلیل همزمانی با امتحانات پایان‌ترم عزیزان، سعی می‌کنیم مطالب کمی ساده‌تر و مختصر باشد. در این جلسه در مورد کاربرد PWM در ساخت یک ربات مسیریاب حرفه‌ای توضیح می‌دهیم.

در ربات‌های مسیریاب ساده که در جلسه ۱۹ نیز در مورد آن توضیح دادیم، هنوز میکروکنترلر وارد ربات ما نشده و تصمیصماتی که ربات ما برای یافتن خط می‌گیرید بسیار مقدماتی هستند و ربات نمی‌تواند مسیرهای پیچیده را با سرعت مناسبی دنبال کند.

یکی از مهمترین فواید استفاده از میکروکنترلر در ساخت ربات‌های مسیریاب، استفاده از قابلیت PWM برای هدایت موتورهای

ربات است. اما به چه صورت از PWM استفاده می‌کنیم؟



به شکل بالا نگاه کنید، در جلسه‌ی نوزدهم گفتیم ۳ سنسور هر طرف را با همدیگرAND منطقی می‌کنیم و اگر هر یک از این ۳ سنسور خط را تشخیص داد، موتور همان سمت را متوقف می‌کنیم تا به این ترتیب ربات خط را تعقیب کند.

اما در ربات‌های مسیریاب میکروکنترلر دار، ما می‌توانیم برای هر سنسور، به طور مجزا دستوری به موتور بدهیم. برای درک این موضوع مجدد به شکل بالا نگاه کنید، این نمای کلی یک ربات از زیر است. سنسورهای آن را به ترتیب از چپ به راست، از ۷ تا ۱ شماره گذاری می‌کنیم.
همانطور که به خاط دارید در ربات‌های بدون میکروکنترلر، تفاوتی نداشت که سنسور ۱ یا ۲ یا ۳ کدامیک خط را بیابند، هر کدام خط را تشخیص می‌داد، موتور سمت چپ خاموش می‌شد. اما در ربات‌های میکروکنترلر دار، ما می‌توانیم تعیین کنیم که مثلاً اگر سنسور شماره‌ی ۳ خط را دید، موتور سمت چپ به طور کامل متوقف نشود، بلکه سرعت آن به نصف کاهش پیدا کند. این کار به نظر هم منطقی می‌رسد، زیرا سنسور شماره‌ی ۳ و ۵ تا خط فاصله‌ی کمی دارند و نیاز نیست وقتی خط را تشخیص می‌دهند به طور کامل موتور متوقف شود، بلکه فقط کافیست سرعت موتور کمی کاهش پیدا کند تا ربات به تدریج به روی خط باز گردد. این عمل باعث می‌شود حرکت ربات نرم‌تر و دقیق‌تر بشود و در مجموع سرعت ربات بالاتر برود.
حال اگر سنسور شماره‌ی ۲ خط را ببیند، یعنی شرایط کمی خطرناک‌تر شده و ربات ممکن است از خط خارج شود، پس می‌توانیم در اینجا به موتور دستور توقف کامل را بدهیم تا ربات با سرعت بیشتری به مسیر مسابقه بازگردد. و در نهایت اگر سنسور شماره‌ی ۱ خط را ببیند، یعنی ربات در آستانه‌ی خروج از مسیر مسابقه قرار گرفته است و باید با حداکثر توان ربات را به مسیر مسابقه بازگرداند. برای این کار به موتور سمت چپ دستور باز گشت به عقب را می‌دهیم. این کار بیشترین سرعت ممکن برای چرخش ربات را فراهم می‌سازد و ربات با سرعت زیادی به زمینه مسابقه باز می‌گردد.

در زیر بخشی از برنامه‌ی یک ربات مسیریاب پیشرفته، که فقط برای سنسورهای سمت چپ و طبق توضیحات بالا نوشته شده است را می‌بینید. همانطور که می‌دانید ما نیاز به ۳ پایه به عنوان ورودی برای دریافت وضعیت سنسورهای سمت چپ، و یک پایه‌ی

خروجی و یک PWM برای کنترل موتور سمت چپ داریم که به ترتیب زیر هستند:

PA.0برای سنسور شماره‌ی ۱

PA.1برای سنسور شماره‌ی ۲

PA.2برای سنسور شماره‌ی ۳

PD.6و OCR2 برای کنترل موتور چپ

PD.3و OCR1BL برای کنترل موتور راست

حالا به برنامه دقت کنید:

```
if (PINA.0==0)
{
  PORTD.6=0;
  OCR2=127;

  PORTD.3=0;
   OCR1BL=255;
}

if (PINA.1==0)
{
  PORTD.6=0;
   OCR2=0;

 PORTD.3=0;
   OCR1BL=255;
}


if (PINA.2==0)
{
  PORTD.6=1;
   OCR2=0;

 PORTD.3=0;
 OCR1BL=255;
}
```

به همین منوال باید برای سنسورهای سمت راست هم برنامه را ادامه دهید. دقت کنید که باید حتماً قبل از نوشتن برنامه، از داخل CodeWizard، تنظیمات اولیه را انجام دهید.

در مورد سنسور وسط هم در جلسه‌ی بیستم توضیحاتی داده شد، اگر این سنسور خط را تشخیص دهد، بیانگر این است که ربات در وضعیت مناسبی نسبت به خط قرار دارد و هر ۲ موتور با تمام توان به سمت جلو حرکت می‌کنند. اگر پایه‌ی PA.۳ را نیز به سنسور وسط اختصاص دهیم، برای این سنسور نیز داریم:

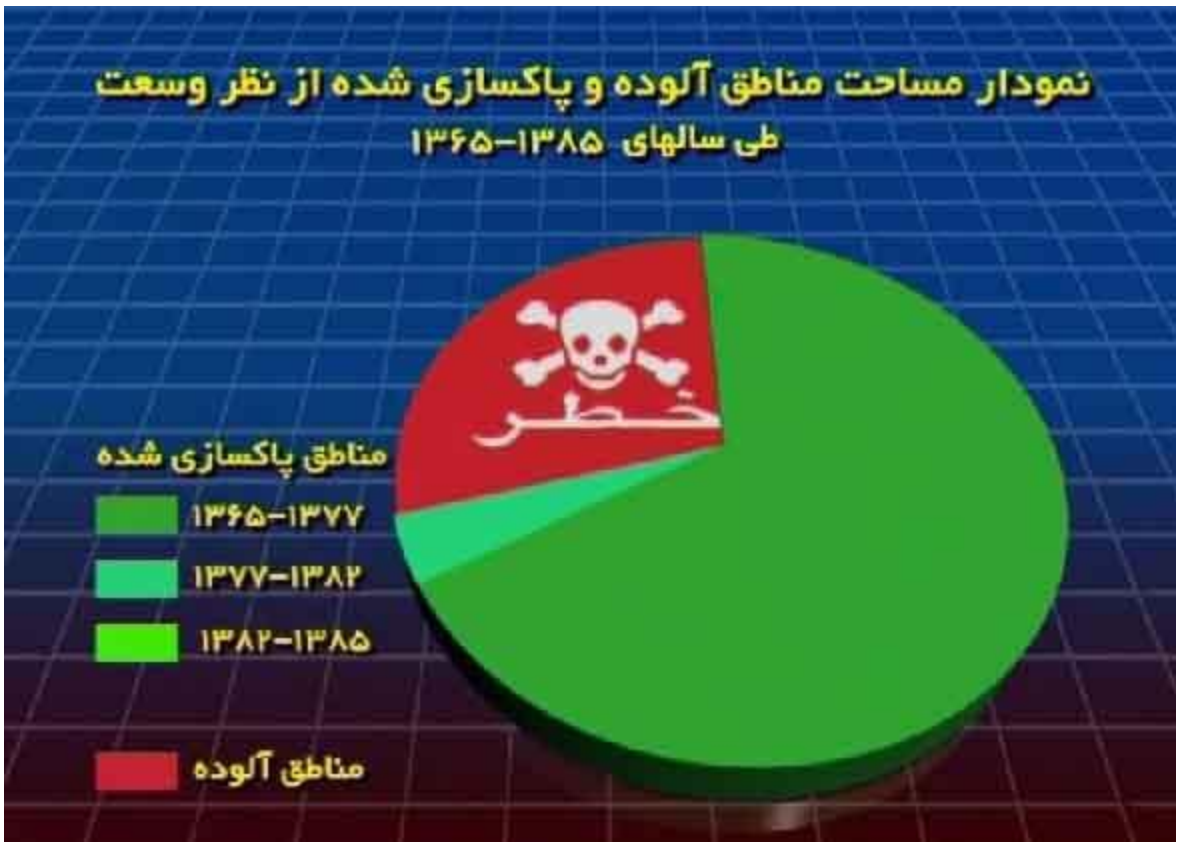
```
if (PINA.3==0)
{
PORTD.6=0;
 OCR2=255;

PORTD.3=0;
 OCR1BL=255;
}
```

**جلسه‌ی سی و نهم**

آشنایی با ربات های مین یاب خودکار...

این جلسه تصمیم داریم شما را با لیگ ربات‌های مین یاب یا "Deminer" آشنا کنیم. اما اهمیت این ربات‌ها چیست و اصولاً چرا چنین مسابقاتی برگزار می‌شود؟
پاراگراف زیر عیناً از دفترچه‌ی قوانین مسابقات کشوری ربات‌های مین یاب در تبیین اهداف ربات‌های مین‌یاب آورده شده است:

»بر اساس گزارشات موجود، ایران سومین کشور دنیا از لحاظ مین‌های دفن شده است. در طول ۸ سال جنگ تحمیلی عراق بر علیه ایران بالغ بر ۱۶ میلیون مین در سرزمین‌های مرزی غرب و جنوب کشور کاشته شده است که بعضاً به علت دورافتاده بودن و صعب‌العبور بودن مناطق خنثی نشده باقی مانده و همه ساله افراد زیادی از بین مردم عادی و نظامیان قربانی می‌کنند. استانهای آذربایجان غربی، کرمانشاه، کردستان، ایلام و خوزستان، هنوز در معرض خطر انفجار مین‌های باقی‌مانده از بیست سال پیش هستند. کشف و خنثی سازی مین‌ها کاری بسیار خطرناک و پرهزینه است. این امید وجود دارد که ربات‌های مین‌یاب بتوانند این خطر را رفع نموده و تلفات انسانی و هزینه مین‌یابی را کاهش دهند«.




این پاراگراف به خوبی هدف از ساخت چنین ربات‌هایی را تبیین می‌کند. اولین دوره‌ی مسابقات ربات‌های مین‌یاب در ایران، ۶ سال پیش و توسط دانشگاه امیرکبیر برگزار شد، اما آن مسابقات در سال‌های بعد تداوم نیافت. اما مجدداً بعد از چند سال لیگ ربات‌های مین‌یاب جایگاه خود را در بین لیگ‌های مسابقاتی رباتیک ایران باز کرد و توانست به یکی از پر طرفدارترین لیگ‌های رباتیک در ایران تبدیل شود. در حال حاضر مسابقات متعددی سالیانه در این لیگ برگزار می‌شود که معتبرترین آن هم مسابقات ملی رباتیک خوارزمی است که آبان ما سال گذشته دوره‌ی نخست آن برگزار شد.

هر چند که این لیگ توسط فداراسیون جهانی روبوکاپ به عنوان لیگ مسابقات جهانی شناخته نشده است، اما در تمامی مسابقات معتبری که هر سال در کشور ما برگزار می‌شود، نظیر خوارزمی، IranOpen، موش‌های هوشمند، دانشگاه نوشیروانی بابل و... مسابقات در این لیگ برگزار می‌شود و طرفداران زیادی هم دارد.این لیگ در ۳ زیر شاخه‌ی زیر برگزار می‌شود:

**۱ -ربات‌های مین‌یاب خودکار**

**۲ -ربات‌های مین‌یاب غیرخودکار(دستی)**

**۳ -رقابت فنی**

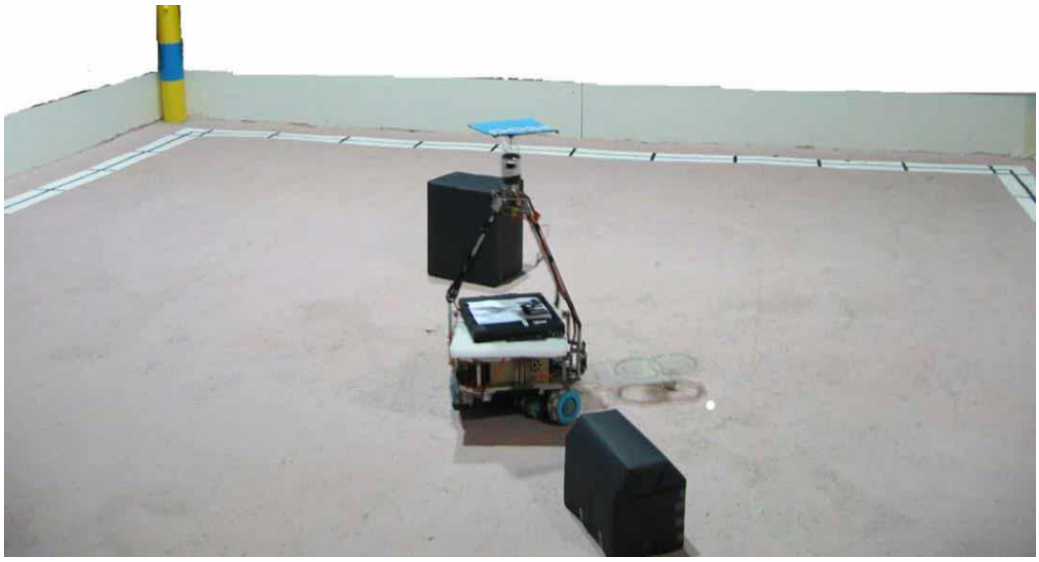
در این جلسه بحث ما فقط در مورد ربات‌های مین یاب خودکار خواهد بود، در جلسات آینده، در مورد ربات‌های مین یاب غیر خودکار و رقابت فنی لیگ هم توضیح خواهیم داد.

این لیگ جزو لیگ‌های دانش‌جویی دسته بندی می‌شود، اما به دلیل عدم پیچیدگی‌های فنی‌ای که نسبت به سایر لیگ‌های دانشجویی(مثل ربات‌های فوتبالیستو ...) دارد، باعث شده تا این لیگ در اکثر مسابقات داخلی  پرشرکت کننده‌ترین لیگ‌های دانشجویی باشد. حتی در

مسابقات امسال تیم‌های دانش‌آموزی حرفه‌ای هم در این مسابقات شرکت داشتند که اتفاقاً موفق به کسب جایگاه‌های مناسبی هم شدند.
در ادامه تشریح می‌کنیم که این ربات چه وظایفی بر عهده دارد و در پیست مسابقه باید چه عملی را انجام دهد.
یک ربات مین یاب، باید قادر باشد تمامی مین‌های کارگذاری شده در یک ناحیه‌ی مشخص را کشف و خنثی یا نابود سازد. اما شاید ساخت رباتی که بتواند مین‌ها را خنثی یا نابود سازد کار بسیار پیچیده‌ای باشد، زیرا این کار برای انسانها هم کار ساده‌ای نیست و نیاز به قابلیت‌های فیزیکی و هوشی بسیار بالایی دارد، و تا به حال رباتی با چنین قابلیت‌هایی ساخته نشده است. پس این بخش (یعنی خنثی سازی مین‌ها) از مسابقات حذف شده است و کار ربات مین یاب در زمین مسابقه به کشف مین‌ها و تهیه‌ی نقشه‌ی میدان مین بسنده شده است. ابتدا در مورد مشخصات زمین مسابقه و نحوه‌ی کارگذاری مین‌ها در این زمین و سپس در مورد نحوه‌ی کشف مین‌ها توضیح می‌دهیم و در نهایت هم به نقشه‌ای که باید از میدان مین تهیه می‌شود می‌پردازیم.

## زمین مسابقه

زمین مسابقه را هیئت داوری و کمیته‌ی برگزاری هر مسابقه با توجه به امکانات و شرایط برگزار کننده‌ها طراحی می‌کنند، پس طبیعی است که زمین مسابقات مختلف با هم تفاوت‌هایی داشته باشد، اما تشابهاتی بین همه‌ی آن‌ها وجود دارد که به آن‌ها خواهیم پرداخت.



شکل کلی زمین مسابقات مین‌یاب به شکل بالاست. ابعاد زمین معمولاً در حدود ۵×۵ متر است و معمولاً در داخل زمین موانعی مکعب شکل با ابعاد گوناگون بین (۲۰ تا ۵۰ سانتی متر) قرار دارد. جنس زمین از گچ و خاک است و سعی شده است تا جای امکان سطح آن مسطح و سفت باشد. مرزهای زمین با نوارهایی سفید رنگ (با پهنای حدوداً ۳۰ سانتی‌متر) مشخص شده و خارج از این مرزها مین کاشته نشده است. مین‌ها در زیر این زمین و با فاصله‌ی حدوداً ۱۰سانتی متر از سطح آن کاشته شده‌اند و جای آن‌ها را به جز تیم داوری هیچ فرد دیگری نمی‌داند و در واقع کار اصلی این ربات‌ها این است که جای مین‌ها را با علایمی مشخص نشان دهند. اما ربات‌ها چگونه‌در این زمین می‌توانند جای مین‌ها را پیدا کنند؟

**کشف مین**

همانطور که در عکس بالا می‌بینید، یکی از پرکاربردترین عناصر در مین‌ها و کلاً تسلیحات جنگی، فلزات هستند. این مسئله اساس کار

ربات‌های مین‌یاب است، ربات‌های مین یاب به یک عدد سنسور فلزیاب مجهز هستند که می‌تواند فلزات را تا فاصله‌ی معینی(حدوداً ۲۰ سانتی‌متر) در زیر خاک نیز تشخیص دهند. در مسابقات هم به جای مین واقعی، از یک عدد قوطی کنسرو ماهی استفاده شده است. کار اصلی ربات این است که این سنسور را در تمام نقاط زمین حرکت دهد و هرجا که توسط سنسور، جای مین (قوطی فلزی) را پیدا کرد، ربات بر روی آن نقطه ۵ ثانیه توقف کند و یک LED قرمز را روشن کند، تا به این وسیله به تیم داوری جای مین را اعلام کند.
زمین مسابقه به مربع‌های ۵۰×۵۰ سانتی‌متر تقسیم شده است، که مین‌ها در وسط بعضی از این مربع‌ها کارگذاری شده است. زمانیکه رباتی جای یک مین را اعلام می‌کند، داور با توجه به این که جای مین‌ها را می داند، مشخص می‌کند که ربات جای مین را درست تشخیص داده است یا خیر، و امتیاز هر مین را جداگانه محاسبه می‌کند. امتیاز نهایی هر تیم، بر حسب تعداد مین‌های درست کشف شده، زمان، و تعداد برخوردها با مانع و دیواره‌ها، طبق فرمول خاصی محاسبه‌می‌شود.در نهایت امتیاز تهیه‌ی نقشه‌ی میدان مین توسط ربات(که امتیاز زیادی هم هست) به این امتیازات اضافه می‌شود. اما نقشه چیست؟

## نقشه‌های مین‌ها

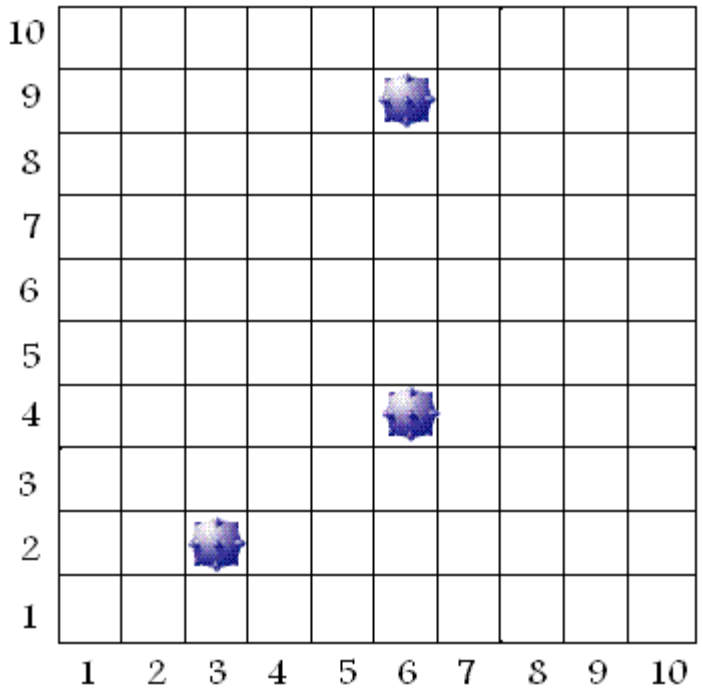
ربات‌ها باید امکانات مکان‌یابی‌ای در اختیار داشته باشند تا بتوانند به وسیله‌ی آن‌ها مختصات دقیق مین‌ها را مشخص کنند و در پایان مسابقه مختصات تمام مین‌های کشف شده را در اختیار کاربر قرار دهند. این بخش تقریباً پیچیده‌ترین بخش ساخت یک ربات مین‌یاب خودکار است، و تیم‌های محدودی این قابلیت را دارند که از محل مین‌ها نقشه‌ای تهیه کنند .
تهیه‌ی نقشه‌ی مین‌ها اجباری نیست، اما هر تیمی که بتواند این کار را انجام دهد امتیاز قابل توجهی دریافت خواهد کرد که شانس آن را برای موفقیت بسیار افزایش خواهد داد. در جلسه‌ی آینده در مورد تهیه‌ی نقشه‌ی مین‌ها بیشتر توضیح می‌دهیم، همچنین توضیحات مفصلی خواهیم داشت در باب الگوریتم‌هایی که در کوتاه‌ترین زمان ممکن ربات کل زمین را بتواند جستجو کند. همچنین در مورد ربات‌های مین یاب دستی(غیر خودکار) و رقابت فنی لیگ هم توضیحاتی خواهیم داد.

## جلسه‌ی چهلم

توضیح بیشتر در مورد ربات‌های مین‌یاب خودکار(autonomous) ، الگوریتم‌های متداول برای جستجوی زمین مسابقه و...

در این جلسه نیز در مورد ربات‌های مین‌یاب خودکار، توضیحات جلسات پیش را تکمیل خواهیم کرد و در مورد نحوه‌ی ارایه‌ی نقشه‌ی مین‌ها توضیح خواهیم داد، سپس الگوریتم‌های متداول در ساخت ربات‌های مین‌یاب را مورد بررسی قرار خواهیم داد.

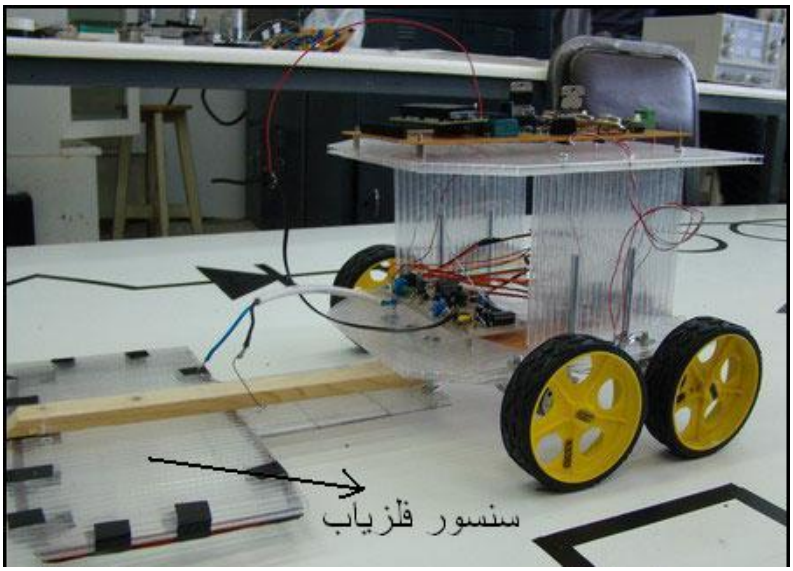
ربات‌های مین‌یاب خودکار باید به سیستم‌هایی مجهز باشند که وقتی مینی را کشف می‌کنند، بتوانند موقعیت(مختصات مکانی) آن مین را ثبت کنند، و در پایان زمان جستجو، مختصات تمام مین‌ها را در اختیار کاربر بگذارند تا کاربر بتواند با در اختیار داشتن این مختصات مکان مین‌ها را به راحتی پیدا کند. همان‌طور که در جلسه‌ی پیش هم گفته شد، زمین مسابقه به مربع‌های ۵۰×۵۰ سانتی‌متری فرضی تقسیم شده است و مین‌ها دقیقاً در وسط بعضی از این مربع‌های فرضی کاشته شده‌اند. اعلام مختصات مین‌ها نیز باید بر اساس همین تقسیم بندی‌های فرضی انجام شود. مثلاً اگر زمین مسابقه ۵×۵ متر باشد، این تقسیم‌بندی زمین مسابقه را به ۱۰۰ خانه تقسیم می‌کند که در وسط بعضی از آن‌ها (کمتر از ۱۰ خانه) مین کاشته شده است. ربات باید قادر باشد توسط ۲ عدد بین ۱ تا ۱۰ مختصات هر مین را اعلام کند. به عنوان مثال به شکل زیر نگاه کنید، این شکل مکان مین‌هایی که در زمین مسابقه دفن شده‌اند را نشان می‌دهد.




ربات در پایان زمان مسابقه، باید بتواند صورت خودکار مختصات خانه‌هایی را که در آن‌ها مین دفن شده است را به طریقی به کاربر (داور مسابقه) اعلام کند: مثلاً اگر زمین مسابقه مانند شکل بالا باشد، ربات باید ۳ جفت عدد روبه‌رو را اعلام کند. (۳،۲) و (۶،۴) و (۶،۹)

بعضی از ربات‌ها این اعداد را توسط نمایش‌گرهای کوچکی که بر روی آن تعبیه شده است نشان می‌دهند. این نمایش‌گرها مشابه نمایشگرهاییست که در ساخت ساعت‌های مچی دیجیتال استفاده می‌شوند. کار با این نمایش‌گرها را نیز در جلسات آینده توضیح خواهیم داد.

## الگوریتم‌های جستجوی زمین برای کشف مین




کار اصلی یک ربات مین‌یاب این است که سنسور فلزیاب را در تمام نقاط زمین حرکت دهد و در هر کجا که وجود فلز را احساس کرد، وجود مین فرضی را اعلام کند. اما ۲ نکته‌ی بسیار مهم را در ساخت یک ربات مین‌یاب باید مد نظر قرار دهیم تا ربات عمل‌کرد مطلوبی داشته باشد:

1-در جستجوی خود به دنبال مین کلیه‌ی نقاط زمین را پوشش دهد و هیچ نقطه‌ای از زمین نباشد که سنسور فلز یاب از روی آن گذر نکرده باشد.

2-عمل بالا را در کمترین زمان ممکن انجام دهد. به عنوان مثال اگر ربات یکی از نواحی زمین را چندین بار بگردد، ممکن است زمان زیادی تلف شود، پس باید الگوریتمی طراحی شود که هر نقطه از زمین بیش از یک بار هم جستجو نشود.

اما پیاده سازی این ۲ نکته کار ساده‌ای نیست و کمتر تیمی می‌تواند این ۲ نکته را به دقت اجرا کند، الگوریتم ساده‌تری هم برای جستجوی زمین وجود دارد که در این راه نیازی نیست ۲ نکته‌ی بالا رعایت شوند، اکثر تیم‌ها هم برای هدایت ربات‌های خود در زمین مسابقه از الگوریتم «تصادفی» (Random) استفاده می‌کنند. اما طبیعیست که عدم رعایت دو نکته‌ی بالا در طراحی ربات، موجب کاهش دقت و سرعت ربات خواهد شد.

## الگوریتم جستجوی تصادفی

در این روش، ربات زمین مسابقه را طبق هیچ الگوی خاصی جستجو نمی‌کند، و زمانی‌که به دیواره‌ها یا موانع برسد، فقط جهت خود را عوض می‌کند و راه را ادامه می‌دهد، هر زمانی هم که وجود مین را احساس کند، ۵ ثانیه بر روی آن متوقف می‌شود و مجدداً به راه خود ادامه می‌دهد. در این الگوریتم سازندگان ربات فقط ۲ مشکل اساسی در پیش رو دارند:

۱ -چگونه وجود مانع یا دیواره را در جلوی خود تشخیص دهند.
۲ -چگونه وقتی مانع را در جلوی خود تشخیص دادند، جهت خود را به گونه‌ای تغییر دهند که با مانع برخورد نکنند.

برای تشخیص موانع و دیواره‌های اطراف زمین، باید از سنسورهای فاصله‌یاب استفاده نمود. توسط این سنسورها می‌توان فاصله از مانعی که در روبروی ربات قرار دارد را تشخیص داد. در نتیجه زمانی که این فاصله کمتر از حد معینی شد می‌توان تشخیص داد که ربات به مانع نزدیک شده است و امکان برخورد با مانع وجود دارد. در مورد انواع سنسورهای فاصله یاب و نحوه‌ی کار با آن‌ها هم در جلسات آینده توضیح خواهیم داد.

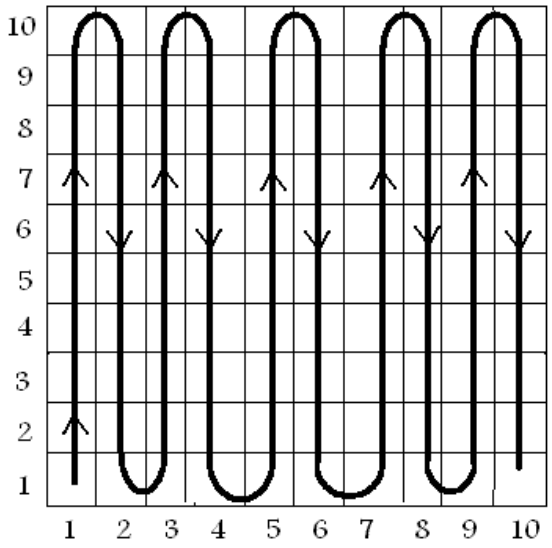
زمانی که ربات به مانع نزدیک شد، برای جلوگیری از برخورد ربات با موانع و دیواره‌ها باید ربات جهت حرکت خود را تغییر دهد. ساده‌ترین راه برای تغییر جهت این است که ربات مسیر حرکت خود را مثلاً ۹۰ درجه تغییر دهد، یعنی هرگاه مانعی را در جلوی خود احساس کرد، به اندازه‌ی ربع دایره ربات به دور خود بچرخد و به مسیر خود ادامه دهد.

این الگوریتم پر کاربردترین الگوریتم برای جستجوی زمین مسابقه است، زیرا پیاده‌سازی آن پیچیدگی فنی زیادی ندارد و به خاطر ساده‌تر بودن سیستم، احتمال بروز خطاهای پیش‌بینی نشده در آن کم‌تر است .

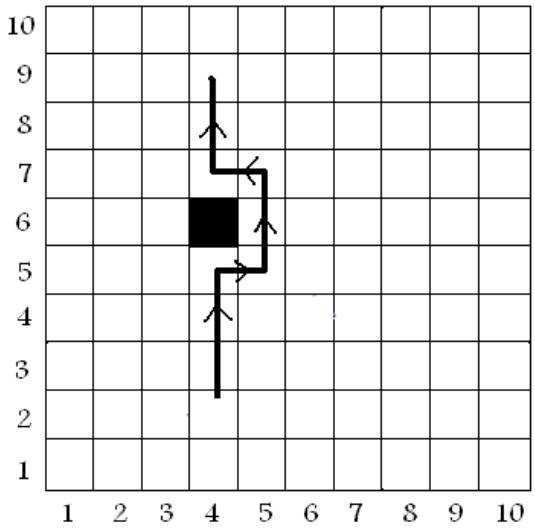
## الگوریتم‌های جستجوی منظم

در این الگوریتم، زمین مسابقه توسط الگوی مشخصی جستجو می‌شود که به واسطه‌ی آن، هیچ قسمتی از زمین جستجو نشده باقی نمی‌ماند و هیچ قسمتی هم چند بار جستجو نمی‌شود.

ساده‌ترین الگو برای جستجوی منظم زمین این است که ربات، زمین را به صورت ردیف به ردیف جستجو کند، شکل زیر نمای کلی حرکت ربات را توسط این الگوریتم نشان می‌دهد:



در شکل بالا، ربات حرکت خود را از خانه‌ی (۱و۱) شروع می‌کند، و در خانه‌ی (۱و۱۰) به پایان می‌رساند. در این روش اگر در زمین مسابقه مانعی وجود داشته باشد، کار کمی پیچیده‌تر می‌شود و ربات باید قادر باشد زمانیکه مانع را احساس می‌کند، به گونه‌ای از برخورد با مانع پیش‌گیری کند که از مسیر حرکت خود نیز منحرف نشود. به شکل زیر دقت کنید:




در شکل بالا، مانع در خانه‌ی شماره‌ی (۶و۴) قرار دارد. ایده‌ال‌ترین مسیر برای ردّ مانع در شکل بالا نشان داده شده است.

درجلسه‌ی آینده، به لیگ ربات‌های مین‌یاب غیر اتوماتیک(دستی) و همچنین رقابت فنی در این لیگ خواهیم پرداخت .

پایان فصل چهارم

گرد آورنده و طراح : مهندس تالیا براری