PWM چیست؟!

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) 12 ولت باشد، با اعمال ولتاژ 6 ولت روی آن، می‌توانید سرعت چرخش آن (rpm) را حدوداً به نصف کاهش دهید.

PWM چیست؟!

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) 12 ولت باشد، با اعمال ولتاژ 6 ولت روی آن، می‌توانید سرعت چرخش آن (rpm) را حدوداً به نصف کاهش دهید.

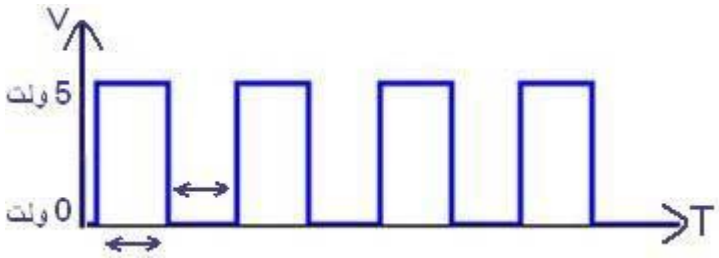
کنترل سرعت ربات، در همه‌ی سطوح رباتیک اهمیت بسیار زیادی دارد، از ربات‌های مسیریاب ساده گرفته تا ربات‌های فوتبالیست. ما تا کنون یاد گرفته‌ایم که چگونه می‌توان به موتور دستور حرکت یا توقف داد، اما راهی برای کنترل سرعت موتور یاد نگرفته‌ایم.

یادآوری

همانطور که می‌دانید سطح ولتاژ پایه‌های خروجی میکروکنترلر منطقی است، یعنی یک پایه‌ای که پایه از میکروکنترلر را به 2برای کنترل موتور ربات استفاده می‌شود فقط می‌تواند 0 یا 1 باشد. ما حرکت ربات اختصاص می‌دهیم، برای صدور دستور حرکت، باید یک پایه را 0 و پایه‌ی دیگر را 1 کنیم، در این حالت بین 2 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 2 پایه‌ی موتور اختلاف پتانسیل 0 ولت باشد). در نتیجه در حالت عادی ما فقط 2 فرمان "حرکت" و "توقف" را می‌توانیم به

.موتورها بدهیم، و ما هیچ کنترلی بر روی سرعت موتور نداریم
تکنیکی است که به کمک آن می‌توانیم ولتاژ پایه‌های خروجی میکروکنترلر، و در نتیجه PWM
.سرعت موتور یا سایر قطعات جانبی که به میکروکنترلر متصل می‌شود را کنترل کنیم

مدولاسیون پهنای پالس" "و به معنای Pulse Width Modulation مخفف واژه‌ی PWM
تکنیکی برای کنترل ولتاژِ پایه‌ی خروجی است. حال ببینیم PWM است. همانطور که گفتیم
.چگونه با این تکنیک می‌توان ولتاژ خروجی را کنترل کرد
می‌دانیم که ولتاژ در پایه‌های خروجی میکروکنترلر یا 0 است یا 5 ولت، اما برای کنترل سرعت
روشی است تا ما PWM .موتور، باید بتوانیم حداقل ولتاژ یکی از پایه‌ها را بین 0 تا 5 تغییر دهیم
.بتوانیم با استفاده از همین پایه‌ی خروجی معمولی، به نوعی ولتاژ را بین 0 تا 5 ولت تغییر دهیم
در این روش، ما با سرعت بالایی سطح ولتاژ خروجی را 0 و بلافاصله 1 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 2 بازه‌ای که با فلش‌های 2طرفه نشان داده شده است، (به عنوان مثال) 10
میکرو ثانیه است. که 5میکرو ثانیه خروجی 1 و سپس 5میکرو ثانیه 0 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

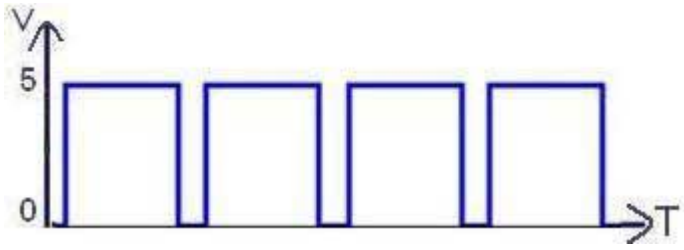
.یعنی در واقع موتور این موج را به صورت یک ولتاژ 2.5 ولت معمولی دریافت می‌کند
به همین ترتیب می‌توان هر ولتاژی بین 0 تا 5 ولت را بر روی خروجی مورد نظر ایجاد کرد. اگر بخواهیم ولتاژی بالاتر از 2.5 ولت داشته باشیم، باید طول بازه‌های زمانی‌ای که خروجی 1 است را نسبت به بازه‌هایی که خروجی 0 است بیشتر کنیم. به عنوان مثال برای ایجاد ولتاژ 2.5 ولت، باید 5 میکرو ثانیه سطح ولتاژ خروجی 1 باشد، سپس 5 میکرو ثانیه سطح ولتاژ 0 شود تا موجی به .شکل بالا ایجاد شود
یا به عنوان مثالی دیگر، اگر بخواهیم در خروجی ولتاژ 4 ولت داشته باشیم، باید باید 8 میکرو ثانیه سطح ولتاژ خروجی 1 باشد، سپس 2 میکرو ثانیه سطح ولتاژ 0 شود، تا ولتاژ پایه‌ی خروجی .ولت باشد 4مورد نظر
:در حقیقت ولتاژ خروجی از رابطه‌ی ساده‌ی زیر به دست می‌اید

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

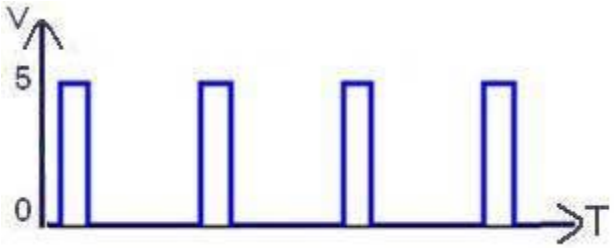
پس طبق رابطه‌ی بالا،برای ایجاد ولتاژ 4 ولت، می‌توان به جای استفاده از بازه‌های 8 و 2 میکرو ثانیه‌ای، از بازه‌های 4 و 1 میکرو ثانیه‌ای استفاده کرد. (یعنی 4میکرو ثانیه 5ولت، 1 میکرو ثانیه (0 ولت

4÷1 = 8 ÷زیرا: 2

نمودار ولتاژهای 4 ولت و 1 ولت در زیر نشان داده شده است:



نمودار نحوه‌ی تولید ولتاژ 4 ولت با تکنیک PWM.



نمودار نحوه‌ی تولید ولتاژ 1 ولت با تکنیک PWM.

حال ببینیم چگونه می‌توان برنامه‌ای نوشت تا بر روی پایه‌ای دلخواه از میکروکنترلر PWM ی برای ولتاژ 4 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 1 می‌کنیم، تا زمانیکه با دستور دیگری آن پایه را 0 کنیم، مقدار خروجی آن پایه 1 خواهد ماند. به این عمل کردن می‌گویند. میکروکنترلر همواره اطلاعاتی که بر روی خروجی قرار می‌دهد اصطلاحاً Latch می‌کند و تا زمانیکه اطلاعات جدید بر روی پایه قرار نگیرد، اطلاعات قبلی را تغییر Latch را نمی‌دهد.

در نتیجه، مثلاً اگر می‌خواهیم پایه‌ای را 5 میکروثانیه 1 وسپس 0 کنیم، کافیست پایه‌ی مورد نظر .کنیم 0را 1 کنیم و 5میلی ثانیه در برنامه تاخیر ایجاد کنیم و سپس پایه‌ی مورد نظر را
برای ولتاژ 2.5 ولت ایجاد کنیم، PWM ، یکB.4 پس وقتی می خواهیم مثلاَ بر روی پایه‌ی .باید به شکل زیر عمل کنیم

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

}
```

برای 2.5 ولت PWM ، یکB.4در بالا یک حلقه‌ی بی‌نهایت تعریف شده است که بر روی پایه‌ی .ایجاد می‌کند

PWM : مبحث کدویژن، وقفه و

برای کاربران CodeVision ابتدا با توابعی که برای ایجاد وقفه در اجرای دستورات برنامه توسط .در نظر گرفته شده آشنا می‌شویم
همان‌طور که در میحث پیش دیدیم، در قسمت‌هایی از برنامه ممکن است نیاز پیدا کنیم تا برای برای این کار توابعی را از پیش CodeVision .لحظاتی روند اجرای دستورات را متوقف کنیم

(.در آینده مفصل توضیح خواهیم داد C در مورد مبحث «توابع» در زبان) .تنظیم کرده است

delay :

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

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

برای ایجاد تاخیرهایی در حد میلی ثانیه به کار می‌رود. در داخل پرانتز، یک delay_ms() تابع عدد صحیح مثبت می‌نویسیم 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

برای ایجاد تاخیرهایی در حد میکروثانیه به کار می‌رود. نحوه‌ی استفاده از آن delay_us() تابع .است delay_ms()دقیقاً مانند

delay ولت را با استفاده از توابع PWM 2.5 به عنوان یک مثال عملی،همان برنامه‌ی ایجاد .بازنویسی می‌کنیم

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

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

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

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

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

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

، دیگر شما می‌توانید هر ولتاژی را که می‌خواهید بر روی پایه‌های خروجی delay با آموختن تابع شبیه سازی PWM ایجاد کنید. البته دوستان دقت داشته باشند که ولتاژی که با تکنیک 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 ولت، 24 ولت، 6 ولت و ...) و برای راه‌اندازی آن‌ها 12ولتاژهای کاری مختلفی داشته باشند (مثلاً استفاده کنیم. سوالی که ممکن است پیش آید این است که L298باید از درایورهای موتور مثل

PWM وصل می‌کنیم و از تکنیک (L298 وقتی ما میکروکنترلر را به درایورهای موتور(مثل PWM برای کنترل سرعت موتور استفاده می‌کنیم، چه وضعیتی پیش می‌آید؟ مثلاً وقتی ما مربوط به ولتاژ 2.5 ولت را تولید می‌کنیم، درایور ما چه عکس العملی نشان می‌دهد؟ آیا ولتاژ 2.5 ولت بر روی پایه‌های موتور قرار می‌گیرد؟
مربوط به 2.5 PWM دقت کنیم، ما وقتی PWM برای پاسخ دادن به این سوال باید به ساختار ولت را تولید می‌کنیم، در حقیقت سطح ولتاژ خروجی را با فواصل زمانی برابر 0 و 1 می‌کنیم، نیز موتور را با L298، (وصل کنیم(مثلاً پایه‌ی 7 L298 پس اگر این خروجی را، به ورودی و همانطور که می‌دانید، .همین الگو کنترل می‌کند و ولتاژی که به موتور می‌دهد را 0 و 1 می‌کند هر ولتاژی که بر روی پایه‌ی شماره‌ی 4 آن قرار گرفته باشد را بر روی موتور قرار L298 می‌دهد(اگر ولتاژ کاری موتور 12 ولت باشد، باید این پایه به 12 ولت متصل شود) . پس جواب مربوط به 2.5 ولت را تولید می‌کنیم، در حقیقت سطح PWM سوال بالا منفیست!!! وقتی ما ولتاژ خروجی در 50 درصد زمان 1 و بقیه‌ی زمان 0 است. پس اگر همان طور که در بالا اشاره داده شود، و ولتاژ پایه‌ی 4 ِ آن 12ولت باشد، درایور، L298 به درایوری مثل PWM شد، این قرار L298 ولتاژ 6 ولت را به موتور می‌دهد. در نتیجه اهمیتی ندارد چه ولتاژی بر روی پایه‌ی 4 ِ مربوط به 2.5 ولت را تولید می‌کنیم، درایور ولتاژی که به موتور PWMگرفته باشد، وقتی که ما PWM می‌دهد را 50 درصد می‌کند. در نتیجه بهتر است از این به بعد به جای آن که بگوییم ِمربوط به 1 ولت، بگوییمPWM ااا50 درصد. یا به جایPWM ولت، بگوییم 2.5مربوط به ااا20 درصدPWM

AVR : در میکروکنترلرهای PWM

AVR برای راه اندازی موتور درمیکروکنترلرهای PWM انجام تنظیمات اولیه برای استفاده از به کمک ما آمده است و کار را کمی ساده‌تر CodeWizard کمی پیچیده است، اما در اینجا هم را بدون توضیح مطرح می‌نماییم، زیرا CodeWizardکرده است. ما در ادامه مبحث، تنظیمات 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

، می‌توانند از CodeWizard به هر حال دوستان عزیز با انجام این تنظیمات اولیه‌ی مختصر در برای هدایت موتورهای PWMالگویی به مراتب ساده‌تر از آنچه تا به حال آموخته‌ایم، برای ایجاد ربات استفاده نمایند.

، چندین خط PWM ، نیازی نیست در هربار استفاده ازAVR در میکروکنترلرهای خانواده‌ی چهارپایه‌ی مشخص از آی سی به این موضوع اختصاص داده ATmega16 برنامه بنویسیم. در شده است. یعنی این چهارپایه علاوه بر کاربردهای معمولی خود، این قابلیت را دارند که در مواقع استفاده شوند PWM لزوم برای تولید.
حال سوال اینجاست که این چهارپایه چه تفاوتی با بقیه‌ی پایه‌های خروجی آی‌سی دارند که آن‌ها را از سایر پایه‌های خروجی میکروکنترلر متمایز می‌سازد؟
فراگرفته‌اید نیست. در PWM برای این چهارپایه نیازی به اجرای الگویی که تا به حال برای ایجاد این روش، فقط شما باید یک عدد صحیح بین 0 تا 255 انتخاب کنید، و طبق الگوی زیر آن را در برنامه‌ی خود بنویسید.

; یک عدد صحیح بین 0 تا 255 = نام رجیستر مربوطه

ِ مورد نیاز خود را با این عدد مشخص PWM شماست، و شما توان PWM این عدد، بیانگر توان 100٪ درصداست، و 0 پایین‌ترین توان و PWMمی‌کنید. که 255 بالاترین توان و مربوط به 0٪ درصد استPWM مربوط به.
50٪ درصد را ایجاد کرده‌اید. یا مثلا PWM به عنوان مثال اگر این عدد را 128 قرار دهید، همان 20٪ درصد بر روی پایه قرار داده‌ایدPWMاگر این عدد 51 باشد،.

: رجیسترهای مربوط به این 4 پایه
به صورت خروجی تعریف شده‌اند، CodeWizardهمانطور که می‌دانید، برای پایه‌هایی که در وجود دارد که هر مقداری در این رجیستر قرار داده شود، مقدار «PORTx» رجیستری به نام

.پایه‌های خروجی متناظر با آن رجیستر را مشخص می‌کند
ِ موتور درPWM در این مبحث با 4 رجیستر دیگر آشنا می‌شویم، که وقتی تنظیمات مربوط به پایه‌ی متناظر PWM را انجام دهیم، هر مقداری که در آن‌ها ریخته شود، توان CodeWizard .را مشخص می‌کنند
نام دارند که به ترتیب، متناظر OCR2و OCR1BL، OCR1AL، OCR0 این رجیسترها .هستند PD.7 و PD.4، PD.5، PB.3 پایه‌های
: پس مثلاً اگر در بخشی از برنامه‌ی خود بنویسیم

```
OCR0=127;
```

.۱۱۱50-درصد به وجود آورده‌ایمPWMمیکروکنترلر، PB.3 در حقیقت بر روی پایه‌ی

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

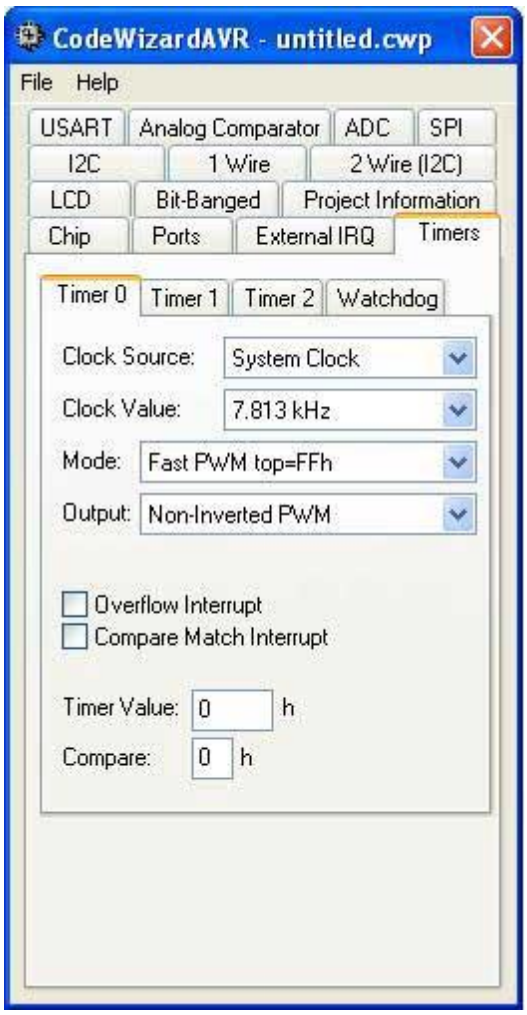
```
OCR1AL=51; // 20% Duty Cycle on PD.5
OCR1BL=255; //100% Duty Cycle on PD.4
OCR2=0; //0% Duty Cycle on PD.7
```

در PWM در ادامه این مبحث، در مورد نحوه‌ی انجام تنظیمات اولیه جهت تولید
....را توضیح خواهیم داد CodeWizard

همانطور که در مباحث قبلی هم متذکر شدیم، در اینجا مجال نیست تا تمام مباحث مربوط به و تایمرها را باز کنیم و مفصل به آن‌ها بپردازیم، به همین خاطر در این بخش قسمتی از PWM

.را بدون توضیح آموزش می‌دهیم CodeWizard تنظیمات در
، Ports ،پس از انجام تنظیمات سایر لبه‌ها (مانندCodeWizard برای انجام تنظیمات به کمک
.را باز کنید Timers ، لبه‌یCodeWizardو ....) در Chip
دارای 3 تایمر مجزا است و ما برای تولید ATmega16 همانطور که می‌بینید میکروکنترلر
باید از این تایمرها استفاده کنیم. تایمرها کاربردهای متعددی دارند، و یکی از مهم‌ترین PWM
مباحث در میکروکنترلر هستند، ما هم در مورد تایمرها در جلسات آینده مفصل توضیح خواهیم
برای کنترل موتورهای ربات PWM داد. اما در این مبحث فقط استفاده از تایمرها را برای ایجاد
.استفاده می‌کنیم

Timer0
:است و باید به شکل زیر تنظیم شود OCR0 مربوط به رجیستر Timer0



نکته‌ای که در مورد تنظیم هر 3 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 (که در به صورت عادی استفاده می‌کنیم. یعنی در حقیقت طول موج را در نمودار ولتاژ (ابتدای مبحث قبل توضیح دادیم هر چه فرکانس بالاتری را انتخاب کنید، طول موج کمتر می‌شود. در عمل .بر زمان تعیین می‌کند 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 انتخاب کنند "Clock Value".

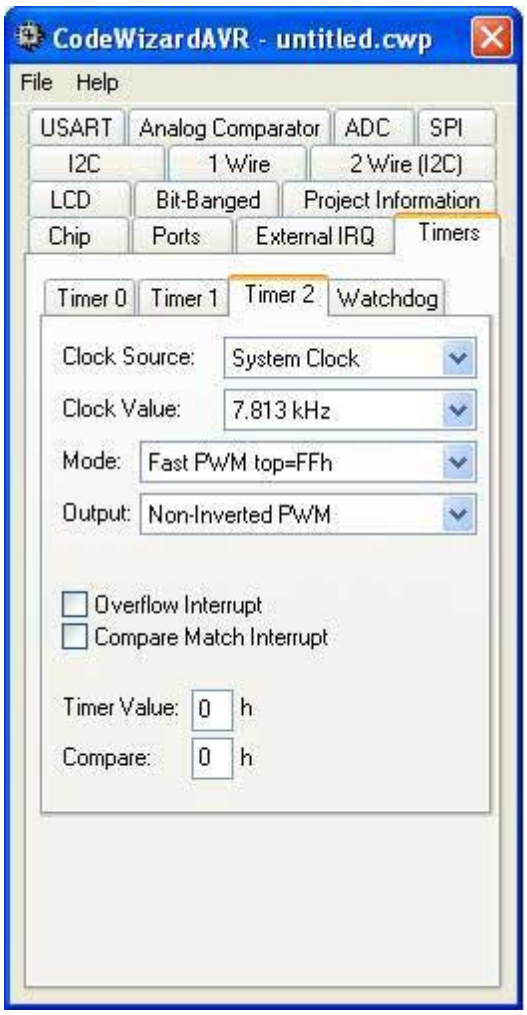
Timer1

شما Clock Value تایمر1 باید به شکل زیر تنظیم شود. دقت کنید که ممکن است در بخش 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، تایمر1 دارای دو خروجی مجزا است که رجیسترهای مربوط به آن‌ها هستند OCR1BL و OCR1AL.

Timer2

تایمر2 میز به شکل زیر تنظیم می‌شود و مانند تایمر0 فقط یک خروجی دارد.



Watchdog

نیز یکی از مباحث مربوط به تایمرهاست که در ادامه بحث به (Watch Dog) یا سگ نگهبان .آن خواهیم پرداخت

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

:نکته‌ی بسیار مهم

، یک خروجی معمولی نیز لازم داریم تا بتوانیم PWM برای کنترل هر موتور، علاوه بر یک پایه‌ی به وسیله‌ی این دو پایه و به کمک درایور موتور، اختلاف پتانسیل مورد نظر را بر روی دو پایه‌ی

متصل می‌کنیم و دو پایه‌ی موتور را L298 موتور برقرار کنیم. این 2 پایه را به دو پایه‌ی ورودی متصل می‌کنیم. حال می‌توانیم موتور را به وسیله‌ی میکروکنترلر L298 نیز، به دو پایه‌ی خروجی با سرعت دلخواه کنترل کنیم. به عنوان مثال اگر بخواهیم موتور ما تقریباً با سرعت نصف بچرخد، متصل کرده L298 را به (OCR2 مربوط به رجیستر) PD.7 و PD.6 و پایه‌های
:باشیم،برنامه‌ی زیر را باید بنویسیم

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

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

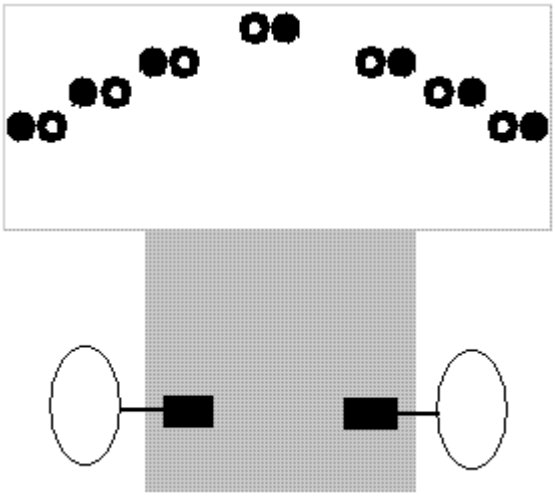
:و اگر بخواهیم موتور ما با همین سرعت و در جهت معکوس بچرخد، می‌نویسیم

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

مطرح شده است و همین آموخته‌های AVR تا به اینجا مباحث پایه‌ای در میکروکنترلرهای .دوستان، نیازهای اولیه‌ی شما عزیزان را برای ساخت ربات‌های نسبتاً حرفه‌ای برطرف می‌سازد

یکی از مهمترین فواید استفاده از میکروکنترلر در ساخت ربات‌های مسیریاب، استفاده از قابلیت

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



به شکل بالا نگاه کنید، در همانطور که می دانید در این ربات ها 3 سنسور هر طرف را با همدیگر منطقی می‌کنیم و اگر هر یک از این 3 سنسور خط را تشخیص داد، موتور همان سمت را AND 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 1 تا 7 شماره گذاری می‌کنیم.

همانطور که به خاط دارید در ربات‌های بدون میکروکنترلر، تفاوتی نداشت که سنسور 1 یا 2 یا 3 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 3 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 3 و 5 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 2 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 1 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 3 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:

برای سنسور شماره‌ی 1 PA.0

برای سنسور شماره‌ی 2 PA.1

برای سنسور شماره‌ی 3 PA.2

برای کنترل موتور چپ OCR2 و PD.6

برای کنترل موتور راست OCR1BL و PD.3

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 ; //end



{
```

به همین منوال باید برای سنسورهای سمت راست هم برنامه را ادامه دهید. دقت کنید که باید تنظیمات اولیه را انجام دهیدCodeWizard حتماً قبل از نوشتن برنامه، از داخل .،
اگر این سنسور خط را تشخیص دهد، بیانگر این است که ربات در وضعیت مناسبی نسبت به خط قرار دارد و هر 2 موتور با تمام توان به سمت جلو حرکت می‌کنند. اگر پایه‌ی PA.3 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;
}
```