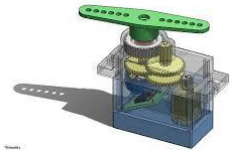
مفهوم PWM ها و کاربرد در servo motor ها :

سروو موتور ها، موتور های DC هستند که از جاروبک و یک برد کنترلی و یک پتانسیومتر (برای موقعیت خروجی از برد) و یک جعبه دنده تشکیل شده اس.

موج مربعی و سیکل کاری برای موتور های سروو:

موتور های سروو برای کنترل یک سیم بیشتر ندارند. و باید در همان یک سیم هر دیتایی که نیاز است را منتقل کنیم. در میکروکنتلر ها برای هر پین دیجیتال که به عنوان دیجیتال رایت روی آن داده ها را می نویسیم می توانیم تعیین کنیم که high باشد یا low باشد. اگر بخواهیم عددی را با آن نشان دهیم نمی توان آن را با زاویه جاروبک مشخص کرد در نتیجه دیتا را بر اساس 0 و 1 (0v,5v) در محور زمانی مشخص می کنند به این صورت که در محور افقی بر اسا فرکانسی که از قبل مشخص کرده ایم مدت زمانی که در موج مربعی سیگنال 0 یا 1 مانده است نماینده یک عدد خواهد بود. (به این مدت زمان دوره کاری، microcycle یا سیکل کاری می گویند.)

سروو موتور بر اساس موج مربعی که می گیرد متوجه می شود که در کجا بایستد. دوره تناوب و سیکل کاری این موج ها فاکتور های اصلی برای مقدار دهی دیتا هستند.



کاربردها و موارد استفاده از PWM در servo موتور ها :

1. کنترل کردن سروو موتور
2. انتقال داده ها روی 2 سیم برای(یک پایه منفی و زمین است تا سطح منطقی 0 باشد و پایه دیگر حاوی داده)
3. کنترل سرعت (می توان گفت چه مقدار از زمان را روشن باشد و چه مقدار را خاموش و به دلیل اینرسی سرعت های مختلف ایجاد می کند)
4. کنترل نور چراغ (ایجاد طیف های مختلف رنگ یا نیمه روشن خاموش کردن LED ها)
5. بازر ها و بوق ها و ایجاد فرکانس های مختلف صوتی (active buzer در صورت مقدار 1 بوق زده و در صورت مقدار 0 خاموش می ماند. نوع دیگری از بازر ها هستند به نام پسیو منفعل که اگه 1 یا 0 بدهیم بوق نمی زند بلکه به تغییرات اثر بوق را نشان می دهد)
6. ..

PWM ها در برد Arduino :

پایه ها:

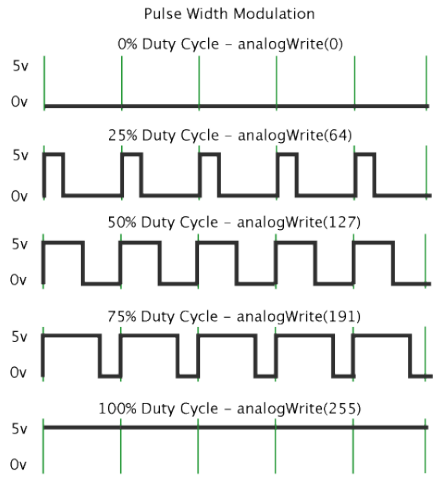
در سمت پین های دیجیتال برد آردواینو در کنار بعضی پین ها pwm نوشته شده است یا علامتی از موج وجود دارد و به این معنی است که خروج را گرفته و به طور سخت افزاری موج ساخته شده را ساپورت می کند. از شماره 14 به بعد در برد pwm را ساپورت نمی کند. و 2 تا 13 برای pwm است.

Analog write and read :

دستوری که می توان موج را روی پین pwm انداخت دستور آنالوگ رایت است. (به معنی آنکه باید از پین های آنولگ استفاده کنیم **نیست** و منظور از آنالوگ pwm است). این تابع شماره پین(9 یا 10) و duty cycle را می گیرد(0 تا 255) و از لحظه اجرا شدن این موج دائما روی این پین ارسال می شود تا زمانی که عوض شود یا متوقف شود. مقدار مورد نظر را مانند پین های دیجیتال با analog read و تعیین آن در setup به عنوان خروجی می توان خواند.

دوره تناوب :

اگر به سایت برد های آردواینو مراجعه کنیم خواهیم دید که جدولی از برد ها و پین های pwm آنها نوشته شده و رو به روی هر کدام فرکانس مربوطه نیز وجود دارد و با استفاده از اسیلوسکوپ قابل مشاهده است.
دوره تناوب برای هر پین مکن است متفاوت باشد و وابسته به نیازمان می توانیم استفاده کنیم.



توضیح در مورد ورودی آنالوگ و تحلیل آن در آردواینو و تابع مورد استفاده (analogRead()) :

Analog digital converter (ADC) مبدل آنالوگ به دیجیتال:

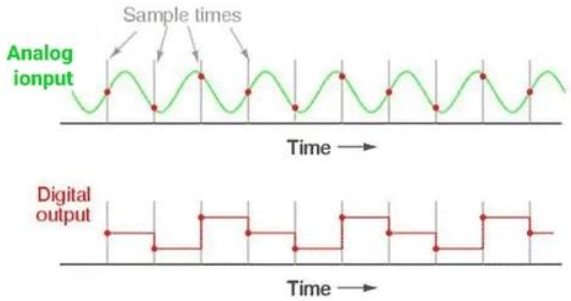
برا اسا قضیه ریمان عمل کرده و یک موج پیوسته را به مقادیر بالا و پایین آن ست می کنیم(گرد می کنیم) و تشکیل که موج گسسته پلکانی را می دهیم که سیگنال آنالیز شده و دیجیتال است.(در غیر این صورت ولتاژ ها میانی را به 0 و 1 بدل کرده و خوانده نمی شود.) این کار را به وسیله قطعه ای به نام ADC انجام می دهیم و می توانیم با آن سطح ولتاژ متغییر را بخوانیم. به دقتی محور عمودی که با آن گرد می کند رزولوشن می گویند.

analogRead() و پین های ورودی خروجی :

با استفاده از پتانسیومتر(مقاومت متغییر) که به خروجی میکرو وصل می کنیم می توانیم موقعیت سیگنال را پیدا کنیم (با تبدیل کردن آن به عددی بین 1000 تا 1024) و می توانیم از آن عدد به عنوان ورودی استفاده کنیم. مثلا سیگنالی که بین 0 تا 5 ولت است در اینجا تبدیل به عددی بین 0 تا 124 می شود و می توان از آن عدد به عنوان ورودی استفاده کرد.
با ورودی که از یک پین آنالوگ به دست آورده ایم در analog read می توان خروجی آن را به عددی به میکرو ثانیه مپ کرد. تقسیم بر 1024 نسبتش می دهد و در 5 ضرب شود ولتاژ آن بدست می آید و ولتاژ خروجی پتانسیو متر می شود. برای تبدیل کردن به 0 تا 180 ابتدا تقسیم بر 1023 کرده و در 180 ضرب می کنیم تا زاویه مورد نظر را بدهد.
پایه های متعلق به analogRead() باید پایه های هدر باشد(کنار پاور) و با حروف A0 تا A11شماره گذاری شده است.(پین های آنالوگ آردایینو می توانند دیجیتال نیز باشند). در جدول مربوطه در سایت آردایینو رزولوشن هر پین جلوی آن نوشته شده است (مثلا 10 bit یعنی 2 به توان 10 حالت می تواند سیگنال را مپ کند)

فرکانس نمونه برداری :

به فاصله میان نمونه ها گفته می شود. به دلیل اینکه بافری وجود ندارد که سیگنال را سیو کند و هر نقطه که آنالوگ رید را فراخوانی کنیم همان نقطه که قرار دارد را بر می گرداند. به لحاظ تئوری با کلاک mhz16 در آردایینو در صورتی که پشت هم فراخوانی اتفاق بیفتد نمی تواند فرکانس نمونه برداری از 16 مگاهرتز بیشتر شود. در عمل تاخیر های داخلی فرکانسی حدود 9615 هرتز را به عنوان ماکسیم فرکانس نمونه بردرای باز می گرداند(15 بار در ثانیه).



رگولاتور :

موتور های سروو توان به خصوصی دارند و در ابعاد عملی با وصل شدن به برد نمی توانند جریان بکشند و ماکسیمم جریان برد - 20 30mA است و برای موتور کم است. پس لازم است که مثبت سروو را که منبع توان آن است از یک منبع خارجی بگیریم ولی منفی آن را به دلیل سیگنال با منفی برد مشترک می کنیم تا سطح منطقی 0 آنها یکسان بشود. در غیر این صورت قطعه رگولاتور میکرو میسوزد. کار رگولاتور آن است که برق ورودی به برد را روی 5 ولت فیکس می کند و پین 5v برد خروجی رگولاتور است. بنا برا این در هر صورت برق نباید از رگولاتور میکرو کشیده شود.

رگولاتور ها یک پایه مثبت خروجی، یک پایه منفی ورودی و یک پایه مثبت ورودی دارد یا در واقع منقی خروجی و ورودی یکسان است. رگولاتور ها کاهنده و افزاینده اند و در اکثر رگولاتور ها مانند7805 نوع آنها کاهنده است. دسته بندی دیگری وجود دارد بر اساس ولتاژ ثابت و جریان ثابت و تبدیل آنها به یک ولتاژ یا جریان مشخص و گاها دارای بردی هستند که با پیچی می توان تنظیم کرد که چه مقداری را داشته باشند.

کاربرد آن در تغذیه مجزا، ثابت نگه داشتن و شارژ کردن باتری ها است.

توابع کتابخانه servo.h:

Servo-attach() :

پین servo pwm رو به پین های مورد نظر متصل می کند. در ورژن های جدید این کتابخانه تنها پین 9 و 10 را ساپورت می کند.

پهنای باند به اندازه درجه موتور را نیز می تواند ساپورت کند.

Syntax

```
servo.attach(pin)
servo.attach(pin, min, max)
```

Servo-write() :

ورودی تابع مقداری بین 0 تا 360 را سایورت می کند که بر اساس زاویه است. گاها این مقدار را به عنوان سرعت نیز در نظر می گیرد(تمام سرعت) و باید دقت شود که استاندارد 180 درجه است یا کانتینیوز 360 تا به موتور آسیب نرساند.

Syntax

```
servo.write(angle)
```

Servo-read() :

خروجی write را عینا دریافت می کند و نشان می دهد سروو در چه زاویه ای قرار دارد.

Syntax

```
servo.read()
```

Servo-writeMicrosecond() :

به جای زاویه یک عدد در واحد میکرو ثانیه می گیرد که استاندارد آن 1500 است که به معنی وسط و 2000 به معنی ساعت گرد کامل و 1000 به معنی پاد ساعت گرد کامل است.

درواقه دقت 1000 واحد است. دارای اندکی خطاست

Syntax

```
servo.writeMicroseconds(uS)
```

Servo-readMicrosecond() :

مانند همان ریدمیکروسکند است و واحد آن نیز میکرو ثانیه است و از آن موقعیت جاروب را می خواند.

Syntax

```
servo.readMicroseconds(uS)
```
