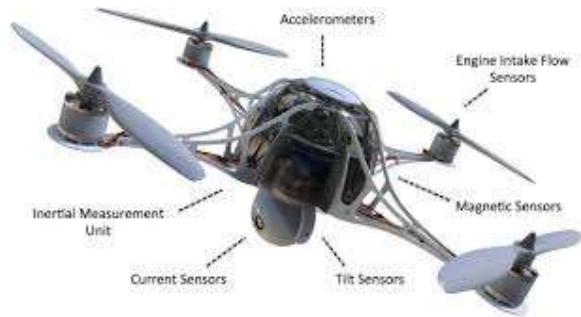


معرفی پروژه

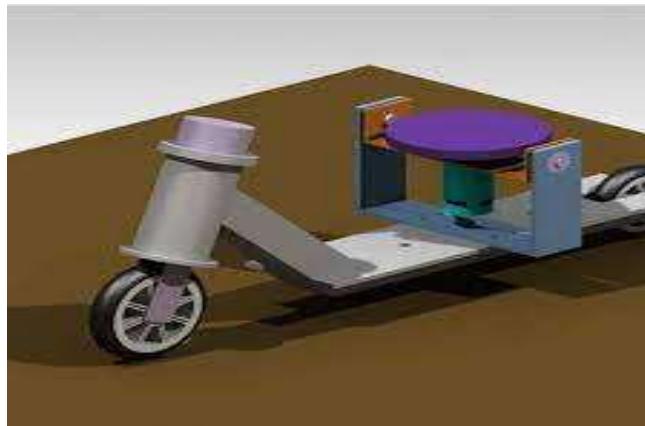
حفظ تعادل مسئله حائز اهمیتی در صنایع نظامی و صنعتی و امروزه حتی تفریحی در جهان محسوب میشود، از ماشین ها گرفته برای حفظ تعادل گیربکس و چرخ ها و یا حتی ربات های پرنده مانند کوادکوپتر ها و جت های *RC* پروازی و یا حتی موشک ها داشتن تعادل در حرکت و پرواز بسیار حائز اهمیت است.



مانند بسیاری از پروژه ها ربات خود تعادلی نیز در پروسه ساخت میتواند از چندین مدل ساخت و نوع قطعات بهره ببرد که به اختصار تصاویر پایین نمایش داده شده است، فرق مهم این پروژه ها استفاده از موتور های مختلف است که در ادامه در بخش موتور ها تفاوت مهم آن هارو شرح خواهیم داد.



ربات تعادلی یک ربات با قابلیت حفظ تعادل روی دو چرخ است که متفاوت با ربات های دارای ژیروسکوپ داخلی و محور چرخش داخلی که با چرخش یک صفحه تعادل خود را حفظ میکنند میباشد.



ربات حفظ تعادل با صفحه چرخنده

ربات صرفا دارای دو موتور باشد که برای حفظ تعادل به صورت لحظه ای از کنترل کننده مدار که در اینجا یک برد توسعه آردوبینو میباشد بهره میبرد. در کل کار ربات به این شکل است که ابتدا توسط سنسور ژیروسکوپ اطلاعاتی مانند موقعیت زاویه ای خودرا دریافت کرده و با استفاده از توابع کنترلی PID به موتور ها دستور عقب یا جلو رفتن میدهد.

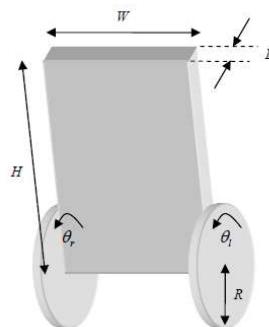
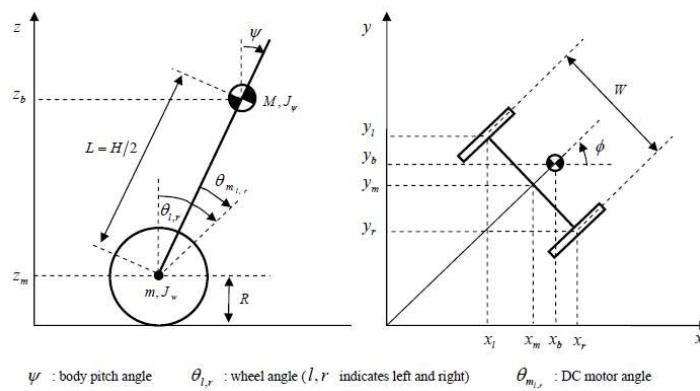
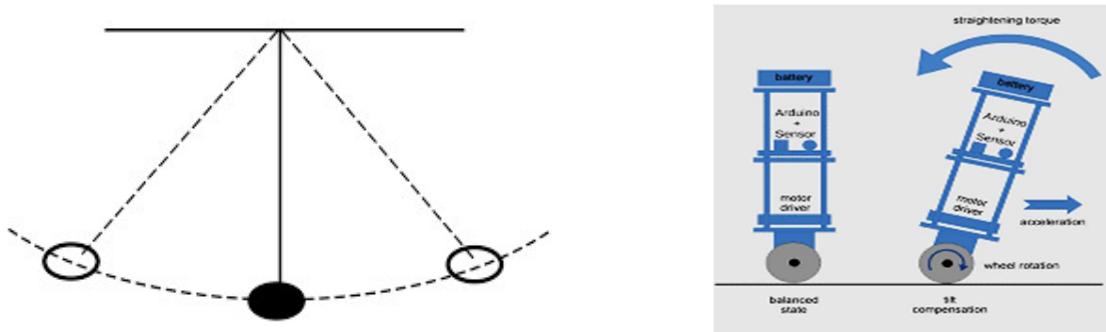


Figure 3-1 Two-wheeled inverted pendulum

Figure 3-2 shows side view and plane view of the two wheeled inverted pendulum. The coordinate system used in 3.2 Motion Equations of Two-Wheeled Inverted Pendulum is described in Figure 3-2.



اگر ربات به سمت جلو مایل شود با توجه به ضرایب ربات به سرعت و آنی شروع به جلو بردن چرخ ها میکند و اگر ربات به سمت عقب مایل شود نیز چرخ ها به سرعت به عقب حرکت کرده و زاویه را ثابت نگه میدارند، که اگر ژیروسکوپ را یک میخ و بدن ربات را یک طناب با یک وزنه تشبيه کنیم مانند این است که سعی بر ثابت ماندن و در یک راستا قرار داشتن با مرکز تعادل میکند.



بلوک دیاگرام کلی ربات با توجه به اینکه در آینده بخش هایی مانند سنسور مجاورت و بلوتوث کنترل اضافه خواهد شد مختصرا به صورت کلی به این شکل میباشد.

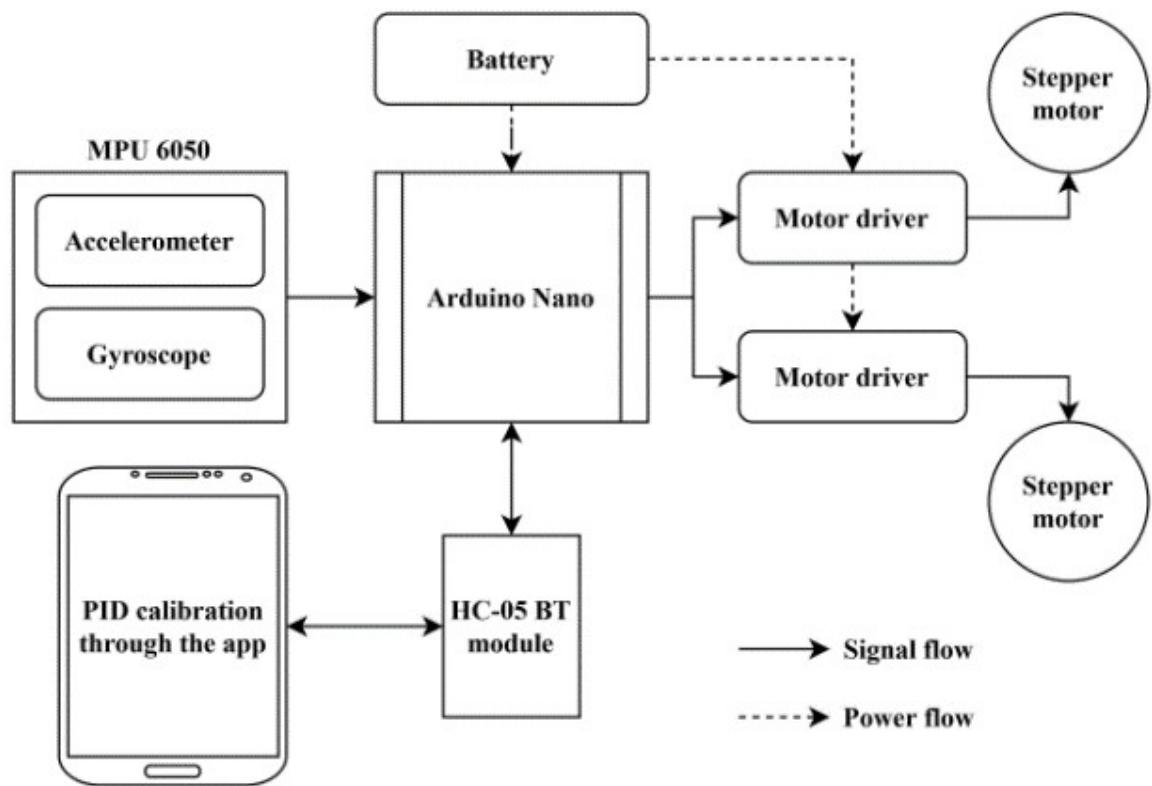


Fig. 1. Block diagram of the self-balancing robot.

برد توسعه آردوینو



معرفی

برد آردوینو Uno یک میکروکنترلر بر پایه Atmega328 می باشد ([datasheet](#)). این برد 14 پین ورودی و خروجی دیجیتال (که 6 تای آن می تواند به عنوان خروجی PWM استفاده گردد)، 6 ورودی آنالوگ، یک تشدیدگر سرامیکی 16مگاهرتز (Ceramic Resonator)، یک پورت USB، یک پاور جک (ورودی منبع تغذیه)، یک ICSP header و یک دکمه ریست دارد. برد Uno شامل کلیه امکانات مورد نیاز جهت بکارگیری میکروکنترلر موجود بر روی برد می باشد. برای شروع تنها با یک کابل USB، به سادگی برد را به کامپیوترتان متصل کنید و یا آن را با یک آداپتور AC-To-DC و یا باتری راه اندازی نمایید.

Uno با برد های پیشین متفاوت است؛ زیرا در آن از تراشه FTDI USB-to-serial است و به جای آن از یک Atmega16U2 (تاریخی از R2) که به عنوان مبدل USB-to-serial برنامه ریزی شده، استفاده گردیده است. نسخه R2 برد Uno دارای یک مقاومت جهت پولینگ اتصال HWB میکروکنترلر به زمین می باشد که تغییر حالت به DFU را آسان تر می کند. نسخه R3 برد، ویژگی های جدیدی دارد که در ادامه آمده است:

- 1.0 pinout : پین های SCL و SDA نزدیک پین AREF و 2 پین جدید دیگر در نزدیکی پین Reset اضافه شده اند. یکی از این پین های جدید IOREF می باشد که اجازه می دهد شیلد ها، خود را با ولتاژ خروجی برد تطبیق دهند. در آینده شیلد ها قادر خواهند بود خود را با برد هایی که از AVR با ولتاژ 5 ولت استفاده می کنند و همچنین برد های آردوینو Due که از ولتاژ 3/3 ولت استفاده می کنند، تطبیق دهند. دو مین پین برای تغییرات آتی رزرو شده است و در حال حاضر هیچ کاربردی ندارد.
- مدار ریست قوی تر
- Atmega16U2 جایگزین U28 شده است.

"Uno" مخفف واژگان one in Italian می باشد که برای مشخص کردن نسخه 0/1 نرم افزار آردوینو انتخاب شده است. Uno آخرین سری از برد های USB دار آردوینو و همچنین مدل مرجع پلتفرم های آردوینو می باشد.

قابلیت ها

میکروکنترلر	Atmega328
ولتاژ عملیاتی	5 ولت
ولتاژ رودی (پیشنهادی)	12-7 ولت
ولتاژ رودی (محده)	20-6 ولت
پین های دیجیتال ورودی/خروجی	14 (6 تای آن به عنوان خروجی PWM استفاده می شود.)
پین های ورودی آنالوگ	6 عدد
جریان DC هر پین ورودی و خروجی	40 میلی آمپر
جریان DC جهت پین V3.3	50 میلی آمپر
حافظه فلاش	32 کیلوبایت (Atmega328) که 0.5 کیلوبایت از آن مورد استفاده Boot-Loader قرار می گیرد.
SRAM	2 کیلوبایت (Atmega328)
EEPROM	1 کیلوبایت (Atmega328)
سرعت کلک	16 مگاهرتز

تغذیه

ولتاژ مورد نیاز آردوینو Uno می تواند از طریق اتصال USB و یا یک منبع تغذیه خارجی مثل باتری یا آداپتور AC-to-DC تأمین گردد. منبع تغذیه به صورت خودکار انتخاب می شود. آداپتور (که سوکت آن از نوع center-positive با قطر 1/2 میلی متری میباشد) می تواند به پاورجک موجود بر روی برد متصل گردد و سیمهای باتری می توانند مستقیماً وارد پین های GND و Vin شوند.

برد می تواند با منبع تغذیه خارجی 6 تا 20 ولت کار کند. اگر ولتاژ منبع تغذیه پایین تر از 7 ولت باشد، روی ولتاژ پین ها نیز اثر خواهد گذاشت و ممکن است ولتاژ خروجی آنها کمتر از 5 ولت شود، و باعث نوسان گردد. ولتاژ بیش از 12 ولت نیز، می تواند موجب افزایش دمای رگولاتور و در نتیجه آسیب برد گردد. ولتاژ پیشنهادی مناسب بین 7 تا 12 ولت می باشد.
پین های مربوط به Power (منبع تغذیه) به شرح زیر است:

VIN: این پین، پین ورودی ولتاژ آردوینو است که در موقع استفاده از منبع تغذیه خارجی (به جای منبع تغذیه تنظیم شده یا اتصال 5 USB ولتی) از آن استفاده می شود و چنانچه برد از طریق پاورجک به منبع تغذیه وصل شده باشد، می توانید از طریق این پین (به عنوان خروجی) به ولتاژ منبع تغذیه دسترسی داشته باشید.

V5: این پین یک ولتاژ تنظیم شده 5 ولت را از طریق رگولاتور موجود بر روی برد فراهم می کند. برد می تواند از طریق پاورجک (12-7 ولت) DC، پورت(5 ولت) USB و یا پین VIN برد (12-7 ولت)، تغذیه گردد. ولتاژ پین های 5 ولت و 3/3 ولت از رگولاتور عبور می نماید و استفاده از ولتاژ این پین ها ممکن است باعث صدمه دیدن برد شود. از همین رو استفاده از این پین ها توصیه نمی گردد.

V3.3: یک ولتاژ 3/3 ولتی، بوسیله ی رگولاتور روی برد فراهم می گردد که حداقل جریان آن 50 میلی آمپر می باشد.

GND: پین های زمین.

IOREF: این پین میزان ولتاژ مرجعی را که میکروکنترلر با آن کار می کند، مشخص می نماید. یک شیلد که به درستی تنظیم شده باشد، می تواند مقدار ولتاژ را از پین IOREF خوانده و منبع تغذیه مناسب خود را انتخاب نماید و یا اینکه مبدل های ولتاژ را برای کار کردن با ولتاژ های 5 ولت یا 3/3 ولت، بر روی خروجی ها فعال نماید.

حافظه

Boot-Loader دارای 32 کیلو بایت حافظه است (نیم کیلوبایت از آن برای Atmega328 استفاده می شود).

(همچنین دارای 2 کیلو بایت حافظه SRAM و 1 کیلو بایت حافظه EEPROM می باشد)(که با تابع کتابخانه ای [EEPROM](#) قابلیت خواندن و نوشتن را دارد)

پین های ورودی-خروجی

هریک از 14 پین دیجیتال Uno می تواند با استفاده از توابع `digitalRead()`, `digitalWrite()`, `pinMode()` به عنوان ورودی یا خروجی استفاده شود. ولتاژ پین ها 5 ولت بوده و ظرفیت جریان جهت هر پین حداکثر 40 میلی آمپر می باشد. همچنین هر یک از این پین ها دارای یک مقاومت داخلی (20-50 کیلواهم) جهت Pull-Up می باشد (که به صورت پیش فرض غیرفعال است). بعلاوه بعضی از پین ها دارای عملکردهای منحصر به فردی می باشند که شرح آن در ذیل آمده است:

- Serial 0 – RX (TX) : پین RX برای دریافت و TX جهت انتقال اطلاعات به صورت سریال و با پروتکل TTL استفاده می شود. این پین ها به پین های مرتبط USB-to-Atmega8U2 TTL متصل هستند.
- External interrupts (وقفه های خارجی) – 2 و 3: این پین ها می توانند طوری تنظیم شوند که یک وقفه را براساس اندکی افزایش یا کاهش لب، و یا هر نوع تغییر در مقدار، ایجاد نمایند. برای جزئیات بیشتر، تابع `attachInterrupt()` را مشاهده نمایید.
- PWM – 3, 5, 6, 9, 10, 11: امکان دسترسی به یک خروجی PWM هشت بیتی را با استفاده از تابع `analogWrite()` فراهم می کنند.
- SPI – SPI کتابخانه ای SPI این پین ها می توانند یک ارتباط `SPI library` ایجاد نمایند.
- LED – 13: یک LED آماده، به پین دیجیتال 13 متصل شده است. هنگامی که پین در حالت HIGH قرار دارد، LED روشن و زمانی که پین در حالت LOW قرار دارد، خاموش می شود.

6 ورودی آنالوگ دارد که از A0 تا A5 نامگذاری شده اند. میزان تفکیک پذیری هر یک از پین ها تا 10 بیت می باشد (به عنوان 1024 مقدار مختلف). به صورت پیش فرض این پین ها می توانند ولتاژ بین ولتاژ پایه (Ground) تا حداکثر 5 ولت را اندازه گیری نمایند. ولی با استفاده از پین AREF و تابع `analogReference()` تغییر حد بالای میزان تفکیک پذیری امکان پذیر می باشد. همچنین بعضی از پین ها دارای عملکردهای منحصر به فردی می باشند که شرح آن در ذیل آمده است:

- TWI : پین A4 یا SDA و A5 یا SCL: این پین ها امکان ایجاد یک ارتباط TWI را با استفاده از توابع کتابخانه ای Wire مقدور می سازند.

سایر پین ها:

- AREF: ولتاژ مرجع برای ورودی های آنالوگ، از طریق این پین و با استفاده از تابع `analogReference()` تأمین می گردد.
- Reset: وضعیت لاین مرتبط را برای ریست میکروکنترلر در حالت LOW قرار می دهد، معمولاً زمانی از این پین استفاده می شود که بخواهید بر روی شیلدتان دکمه ریست قرار دهید. زیرا استفاده از شیلدها از دکمه ریست موجود بر روی برد آردوینو جلوگیری می کند.

نقشه اتصال بین پین های آردوینو به پورتهای Atmega328 را مشاهده نمایید. نحوه اتصال پین ها برای Atmega168، Atmega8 و Atmega328 یکسان می باشد.

ارتباطات

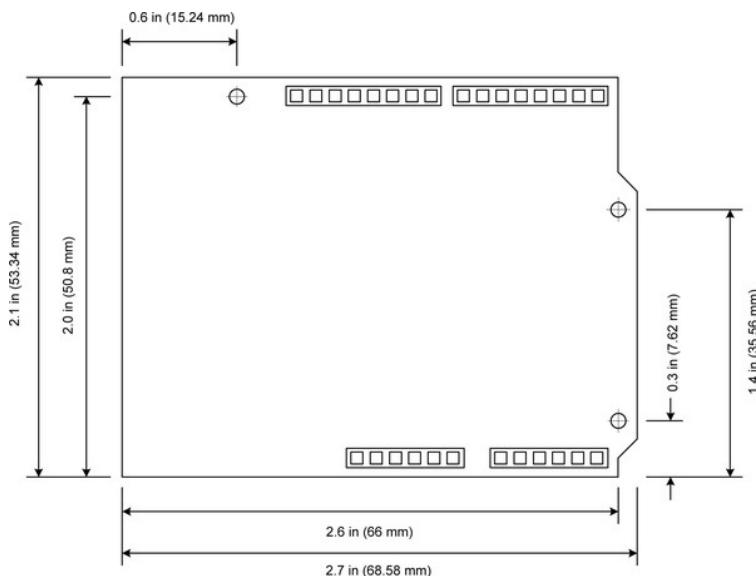
آردوینو Uno شامل امکاناتی می باشد که ارتباط با کامپیوتر، سایر بردهای آردوینو، و یا دیگر میکروکنترلرها را مقدور می سازد. The Atmega328 امکان ارتباط سریالی(5ولت) UART را از طریق پین های(TX)1 ، 0(RX) می نماید. Atmega16U2 فراهم می نماید. Atmega16U2 موجود بر روی برد، این ارتباط سریال را تبدیل به یک ارتباط USB نموده و در نهایت یک پورت سریال مجازی(COM) روی کامپیوتر شما ایجاد می کند. برنامه موجود بر روی Atmega16U2 از درایور USB Com استفاده می نماید و به هیچ درایور جانبی نیاز ندارد. به هر حال در سیستم عامل ویندوز یک فایل.inf مورد نیاز می باشد. نرم افزار آردوینو شامل یک بخش کنترل پورت سریال است که به شما اجازه می دهد داده های متнی را به آردوینو ارسال، یا از آن دریافت نمایید. چراغهای RX و TX موجود بر روی برد در زمان ارسال و دریافت اطلاعات از طریق پردازشگر مبدل USB به سریال و یا اتصال USB به کامپیوتر (به غیر از ارتباط سریالی پین های 0 و 1) در حالت چشمک زن قرار می گیرد.

یک تابع کتابخانه ای SoftwareSerial (برای ایجاد پورت سریال نرم افزاری) امکان ارتباط سریال را بر روی هریک از پین های دیجیتال فراهم می کند.

همچنین Atmega328 از ارتباط SPI(TWI) و I2C(TWI) پشتیبانی می نماید. نرم افزار آردوینو شامل یک تابع کتابخانه ای به نام Wire Library جهت ساده سازی استفاده از درگاه I2C می باشد.

مشخصه فیزیکی برد

حداکثر طول و عرض PCB برد Uno به ترتیب 6.58 و 5.33 سانتی متر می باشد که با احتساب کانکتور USB و پاورجک، ابعاد اصلی آن افزایش می یابد. چهار سوراخ موجود بر روی برد به شما اجازه می دهد که برد را بر روی یک سطح یا جعبه پیچ نمایید. توجه کنید که فاصله بین پین های دیجیتال 7 و 8 برابر 160 میلی متر (0.16 اینچ) و فاصله بین سایر پین ها 100 میلی متر می باشد.

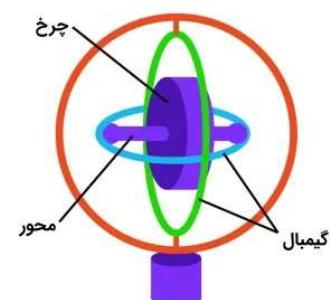


ژیروسکوپ

ژیروسکوپ دستگاهی است که شامل یک چرخ سریع یا پرتویی چرخان از نور است که برای تشخیص انحراف یک جسم از جهت مورد نظر خود استفاده می‌شود. ژیروسکوپ‌ها در قطب نما و خلبان اتوماتیک در کشتی‌ها و هواپیماها، در مکانیسم‌های هدایت اژدرها و در سیستم‌های هدایت داخلی نصب شده در وسایل پرتاب فضایی، موشک‌های بالستیک و ماهواره‌های در حال چرخش استفاده می‌شوند. ژیروسکوپ‌ها را به دو دسته کلی تقسیم می‌کنیم که عبارت از ژیروسکوپ مکانیکی و ژیروسکوپ اپتیکی هستند.

ژیروسکوپ مکانیکی

ژیروسکوپ‌های مکانیکی بر اساس اصل کشف شده در قرن نوزدهم توسط ژان برنارڈ لئون فوکو (Jean-Bernard Léon Foucault) فیزیکدان فرانسوی که نام ژیروسکوپ را به چرخ یا روتور نصب شده در حلقه‌های جیمبال یا گیمبال داده بود، نامگذاری و شناخته شدند. تکانه زاویه‌ای روتور در حال چرخش باعث شد تا حتی زمانی که مجموعه جیمبال کج بود، حالت خود را حفظ کند. در دهه 1850 فوکو آزمایشی را با استفاده از چنین روتوری انجام داد و نشان داد که چرخ در حال چرخیدن جهت اصلی خود را در فضا بدون توجه به چرخش زمین حفظ می‌کند.



این قابلیت کاربردهای متعددی را برای ژیروسکوپ به عنوان نشانگر جهت پیشنهاد کرد و در سال 1908 اولین ژیروسکوپ قابل کار توسط مخترع آلمانی آنشووتر کیمفه (H. Anschütz-Kaempfe) برای استفاده در شناور توسعه یافت. در سال 1909، مخترع آمریکایی المر. اسپری (Elmer A. Sperry) اولین خلبان اتوماتیک را با استفاده از ژیروسکوپ برای حفظ هواپیما در مسیر ساخت. اولین خلبان اتوماتیک کشتی‌ها در کشتی مسافری دانمارکی توسط یک شرکت آلمانی در سال 1916 نصب شد و در همان سال از ژیروسکوپ در طراحی اولین افق مصنوعی هواپیماها استفاده شد.

ژیروسکوپ‌ها از زمان موشک‌های V-1 و V-2 آلمان‌ها در جنگ جهانی دوم برای هدایت خودکار، تصحیح حرکت از راه دور و چرخش در موشک‌های کروز و بالستیک استفاده می‌شوند. همچنین در طول این جنگ، توانایی ژیروسکوپ‌ها برای تعیین جهت با درجه بالایی از دقت، همراه با مکانیزم‌های پیچیده کنترلی، منجر به توسعه تفنگ‌های ثابت شده، بمب افکن و سکوهایی برای حمل اسلحه و آنتن‌های راداری روی کشتی‌ها شد. سیستم‌های هدایت داخلی مورد استفاده فضایپیماهای مداری به یک سکوی کوچک نیاز دارند که با درجه فوق العاده‌ای از دقت ثابت شوند. این مهم توسط ژیروسکوپ‌های سنتی انجام می‌شود. دستگاه‌های بزرگتر و سنگین‌تر به نام چرخ‌های حرکت (یا چرخ‌های عکس العمل) نیز در سیستم‌های کنترل جهت جغرافیایی برخی از ماهواره‌ها استفاده می‌شوند.

ژیروسکوپ نوری یا اپتیکی

ژیروسکوپ‌های نوری، بدون قطعات متحرک، در هواپیماهای جت تجاری، موشک‌های تقویت کننده یا بوستر و ماهواره‌های در حال چرخش استفاده می‌شوند. چنین دستگاه‌هایی بر اساس اثر سایناک Georges Sagnac (ساخته شده‌اند و اولین بار توسط دانشمند فرانسوی جرج سایناک در سال 1913 به جامعه معرفی شدند. در چیزی که سایناک ارائه داد، یک پرتو نوری به شکلی تقسیم شد که بخشی در جهت عقربه‌های ساعت و بخشی در جهت خلاف جهت عقربه‌های ساعت در اطراف یک سکوی چرخشی حرکت می‌کرد. اگر چه هر دو پرتو در یک حلقه بسته حرکت می‌کردند، اما پرتویی که در جهت چرخش سکو حرکت می‌کرد، کمی بعد از اینکه پرتوی مخالف چرخش حرکت سکو به نقطه آغازین بازگشت، به نقطه مبداء بازگشت. در نتیجه یک الگوی تداخل متناوب (نوارهای متناوب روشن و تاریک) تشخیص داده شد که بستگی به میزان دقیق چرخش صفحه گردان دارد.

ژیروسکوپ‌های مورد استفاده از اثر سایناک در دهه 1960 و پس از اختراع لیزر و توسعه فیبر نوری شروع به تکامل و پیشرفت کردند. در ژیروسکوپ لیزری حلقه‌ای، پرتوهای لیزر منحرف و تقسیم شده و سپس از طریق سه حلقه نوکالی عمود بر هم متصل به وسیله نقیله در مسیرهای مخالف هدایت می‌شوند. در واقع حلقه‌ها معمولاً مثلث، مربع یا مستطیل هستند که از گازهای بی اثر پر شده‌اند و از طریق آن‌ها پرتوها بر روی آینه منعکس می‌شوند. همان طور که وسیله نقیله حرکت چرخشی یا گام برداشتن را انجام می‌دهد، الگوهای تداخلی ایجاد شده در حلقه‌های مربوط به ژیروسکوپ توسط سلول‌های فوتولکتریک اندازه گیری می‌شوند. سپس الگوهای هر سه حلقه به منظور تعیین میزان چرخش جسم در سه بعد به صورت عددی یکپارچه می‌شوند. نوع دیگر ژیروسکوپ نوری ژیروسکوپ فیبر نوری است که از لوله‌ها و آینه‌های نوکالی استفاده می‌کند و نور را از طریق الیاف نازکی که محکم در اطراف یک قرقه کوچک پیچیده شده است عبور می‌دهد.

ژیروسکوپ‌های سیستم‌های میکروالکترومکانیکی (MEMS)

ژیروسکوپ‌های MEMS در اصل ژیروسکوپ‌های کوچکی هستند که در دستگاه‌های الکترونیکی یافت می‌شوند. این ژیروسکوپ‌ها بر اساس ایده آونگ فوکو ساخته شده‌اند و از یک عنصر ارتعاشی استفاده می‌کنند.

ژیروسکوپ تشدید کننده نیمکره (HRG)

HRG همچنین به عنوان ژیروسکوپ لیوان نوشیدنی یا ژیروسکوپ قارچ شناخته می‌شود، این ژیروسکوپ از یک پوسته نازک نیمکره‌ای حالت جامد استفاده می‌کند که توسط یک تنه ضخیم ثابت شده است. این پوسته که توسط نیروهای الکترواستاتیک ایجاد شده، توسط الکترودهایی که به طور مستقیم بر روی ساختارهای کوارتز ذوب شده و جدا شده قرار می‌گیرند، یک رزونانس خمشی منتقل می‌شود. خاصیت اینرسی امواج ایستاده خمشی به ایجاد یک اثر ژیروسکوپی کمک می‌کند.

ژیروسکوپ ساختار ارتعاشی

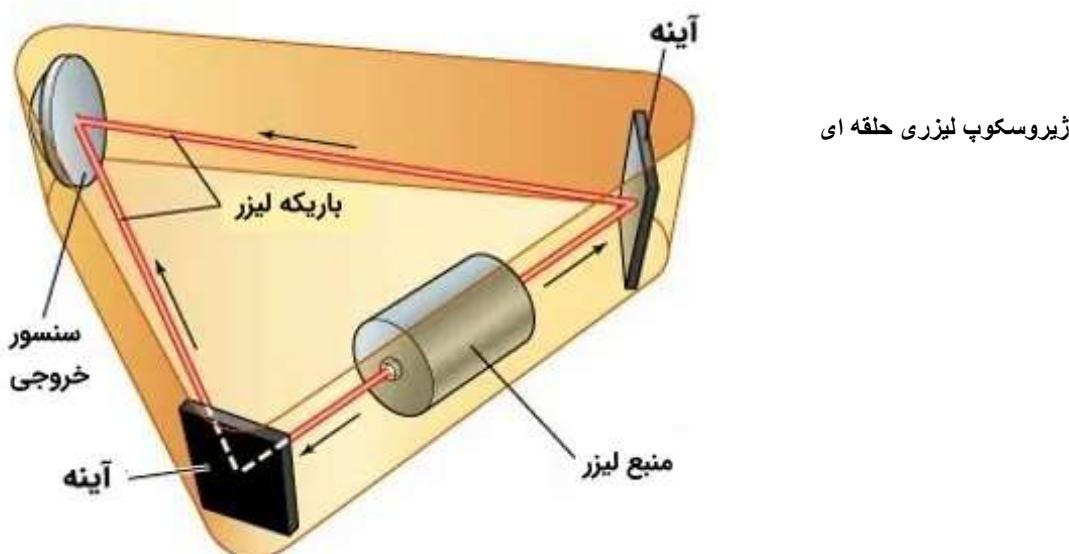
این ژیروسکوپ همچنین به عنوان ژیروسکوپ کوریولیس ارتعاشی (CVG) شناخته می‌شود، ژیروسکوپ ساختار ارتعاشی یک ژیروسکوپ است که از یک ساختار ارتعاشی برای تعیین سرعت چرخش استفاده می‌کند.

ژیروسکوپ تنظیم شده پویا(DTG)

یک روتور است که توسط یک لولای کلی با محورهای خمشی معلق شده است. سختی فنر خمشی مستقل از سرعت چرخش است. اما اینرسی پویا (از اثر واکنش ژیروسکوپی) از جیمبال، سختی منفی فنر مناسب با مربع سرعت چرخش را ایجاد می‌کند. بنابراین با سرعت خاصی، این دو گشتاور اثر یکدیگر را حذف کرده و روتور را از گشتاور آزاد می‌کنند و آن را به یک ژیروسکوپ ایده آل تبدیل می‌کنند.

ژیروسکوپ لیزری حلقه

یک ژیروسکوپ لیزری حلقه‌ای از اثر سایناک برای محاسبه چرخش با اندازه گیری تغییر الگوی تداخل یک پرتو به دو نیمه، حتی زمانی که دو نیمه در جهت مخالف حلقه حرکت می‌کنند، استفاده می‌کند. در اثر سایناک، یک پرتو نور تقسیم می‌شود و دو پرتو به گونه‌ای ساخته می‌شوند که مسیری یکسان اما در جهت مخالف را دنبال کنند. در بازگشت به نقطه ورود، دو پرتو نور اجازه خروج از حلقه و ایجاد تداخل را دارند.



ژیروسکوپ فیبر نوری

ژیروسکوپ فیبر نوری از تداخل نور برای تشخیص چرخش مکانیکی استفاده می‌کند. دو نیمه پرتوی تقسیم شده در یک سیم پیچ از کابل فیبر نوری به طول 5 کیلومتر در جهت مخالف حرکت می‌کنند.

سنسور ژیروسکوپ چیست؟

سنسور ژیروسکوپ سرعت زاویه ای یا شبیب یا جهت گیری جانبی جسم را اندازه گیری می کند. سنسورهای ژیروسکوپ برای چندین محور نیز موجود است.

سنسور ژیروسکوپ وسیله ای است که می تواند جهت گیری و سرعت زاویه ای یک جسم را اندازه گیری و حفظ کند. اینها پیش‌رفته تر از شتاب سنج ها هستند. اینها می توانند شبیب و جهت گیری جسمی را اندازه بگیرند در حالی که شتاب سنج فقط می تواند حرکت خطی را اندازه گیری کند. سنسورهای ژیروسکوپ را سنسور Angular Velocity یا Angular Rate Sensor نیز می نامند. این حسگرها در برنامه هایی نصب می شوند که جهت گیری جسم توسط انسان دشوار است. اندازه گیری شده بر حسب درجه در ثانیه، سرعت زاویه ای تغییر زاویه چرخش جسم در واحد زمان است.

سنسور ژیروسکوپ چطور عمل میکند؟

حسگر های ژیروسکوپ علاوه بر حس سرعت زاویه ای ، می توانند حرکت جسم را نیز اندازه گیری کنند. برای سنجش حرکات دقیق و دقیق تر ، در الکترونیک مصرفی سنسورهای ژیروسکوپ با سنسورهای شتاب سنج ترکیب می شوند.

بسته به جهت سه نوع اندازه گیری سرعت زاویه ای وجود دارد:

Yaw-چرخش افقی روی یک سطح صاف وقتی جسم از بالا دیده می شود.

Pitch-چرخش عمودی همانطور که جسم از جلو دیده می شود.

Roll-چرخش افقی وقتی جسم از جلو دیده می شود.

مفهوم نیروی کوریولیس در سنسورهای ژیروسکوپ به کار میرود. در این سنسور برای اندازه گیری سرعت زاویه ای ، سرعت چرخش سنسور به سیگنال الکتریکی تبدیل می شود. با مشاهده سنسور ژیروسکوپ ارتعاش می توان به اصل کار سنسور ژیروسکوپ پی برد. این سنسور از یک عنصر ارتعاشی داخلی تشکیل شده است که از مواد بلوری به شکل ساختار T2- ساخته شده است. این سازه شامل یک قسمت ثابت در مرکز است که "Sensing Arm" به آن متصل است و "Drive Arm" در هر دو طرف آن است.

این ساختار دوتایی متقابن است. وقتی یک میدان الکتریکی با ارتعاش متناسب به بازو های محرک اعمال شود ، ارتعاشات جانبی مداوم تولید می شود. از آنجا که بازو های Drive متقابن هستند ، وقتی یک بازو به سمت چپ حرکت می کند ، بازوی دیگر به سمت راست حرکت می کند ، بنابراین لرزش های نشتی را لغو می کند. این قسمت ثابت را در مرکز نگه می دارد و بازوی سنجش ثابت است.

هنگامی که نیروی چرخشی خارجی به سنسور وارد می شود ، ارتعاشات عمودی بر روی بازو های Drive ایجاد می شود. این امر منجر به لرزش بازو های Drive بالا و پایین می شود که در نتیجه آن یک نیروی چرخشی بر روی قسمت ثابت در مرکز اثر می گذارد.

چرخش قسمت ثابت منجر به ارتعاشات عمودی بازو های حسگر می شود. این ارتعاشات ایجاد شده در بازوی حسگر به عنوان تغییر در بار الکتریکی اندازه گیری می شود. این تغییر برای اندازه گیری نیروی چرخشی خارجی اعمال شده به سنسور به عنوان چرخش زاویه ای استفاده می شود.

این ساختار دوتایی متقاض است. وقتی یک میدان الکتریکی با ارتعاش متناوب به بازو های محرک اعمال شود ، ارتعاشات جانبی مداوم تولید می شود. از آنجا که بازو های Drive متقاض هستند ، وقتی یک بازو به سمت چپ حرکت می کند ، بازوی دیگر به سمت راست حرکت می کند ، بنابراین لرزش های نشتی را لغو می کند. این قسمت ثابت را در مرکز نگه می دارد و بازوی سنجش ثابت است.

چرخش قسمت ثابت منجر به ارتعاشات عمودی بازو های حسگر می شود. این ارتعاشات ایجاد شده در بازوی حسگر به عنوان تغییر در بار الکتریکی اندازه گیری می شود. این تغییر برای اندازه گیری نیروی چرخشی خارجی اعمال شده به سنسور به عنوان چرخش زاویه ای استفاده می شود.

أنواع سنسور ژيروسکوب

با پیشرفت تکنولوژی دستگاه های بسیار دقیق ، قابل اعتماد و مینیاتوری در حال تولید هستند. اندازه گیری دقیق تر جهت گیری و حرکت در یک فضای سه بعدی با ادغام سنسور ژيروسکوب امکان پذیر شد. ژيروسکوب ها نیز در اندازه های مختلف با عملکرد متفاوت در دسترس هستند.

حسگر های ژيروسکوب بر اساس اندازه های هایی کوچک و بزرگ تقسیم می شوند. از بزرگ به کوچک سلسله مراتب سنسور های ژيروسکوب را می توان به عنوان ژيروسکوب لیزری Fluid ، ژيروسکوب فیبر نوری ، ژيروسکوب Ring و ژيروسکوب Vibration ذکر کرد.

کوچک بودن و استفاده راحت تر از ژيروسکوب ارتعاشی از محبوبیت بیشتری برخوردار است. دقت ژيروسکوب ارتعاشی به مواد عنصر ساکن مورد استفاده در سنسور و تفاوت های ساختاری بستگی دارد. بنابراین ، تولید کنندگان از مواد و ساختار های مختلفی برای افزایش دقت ژيروسکوب ارتعاشی استفاده می کنند.

أنواع ژيروسکوب ارتعاشي

برای مبدل های پیزو الکتریک ، از مواد مانند کریستال و سرامیک برای قسمت ثابت سنسور استفاده می شود. در اینجا برای ساختار های مواد کریستالی مانند ساختار T2 ، از چنگال تنظیم و از چنگال تنظیم H استفاده می شود. هنگامی که از مواد سرامیکی استفاده می شود ساختار منشوری یا ستونی انتخاب می شود.

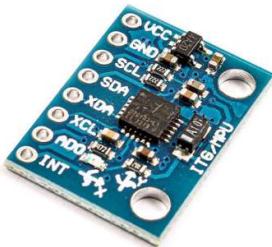
مشخصات سنسور ژيروسکوب ارتعاش شامل فاکتور مقیاس ، ضریب فرکانس دما ، اندازه جمع و جور ، مقاومت در برابر ضربه ، پایداری و ویژگی های نویز است.

کاربرد سنسور ژیروسکوپ

سنسورهای ژیروسکوپ کاربردهای مختلفی دارد Gyros. از لیزر حلقه ای در شاتل های Aircraft و Source استفاده می شود در حالی که Gyros فیبر نوری در اتومبیل های اتومبیلرانی و قایق های موتوری استفاده می شود.

سنسورهای ژیروسکوپ ارتعاشی در سیستم های ناوبری اتومبیل ، سیستم های کنترل پایداری الکترونیکی وسایل نقلیه ، حسگر حرکت برای بازی های موبایل ، سیستم های تشخیص لرزش دوربین در دوربین های دیجیتال ، هلیکوپتر های رادیو کنترل ، سیستم های رباتیک و غیره استفاده می شود.

سنسور MPU6050



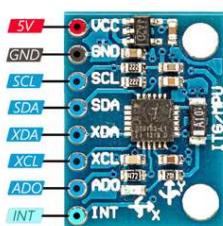
★ سنسور MPU6050 علاوه بر ژیروسکوپ و شتاب سنج ، دارای یک دماسنچ دیجیتال هم می باشد.

سنسور MPU6050 شامل یک شتابسنج و یک ژیروسکوپ مبتنی بر فناوری میکروالکترومکانیکی (MEMS) و یک سنسور دما است. این سنسور در تبدیل مقادیر آنالوگ به دیجیتال بسیار دقیق است؛ زیرا برای هر کanal یک مبدل آنالوگ به دیجیتال 16 بیتی دارد. همچنین قادر به ضبط همزمان مقادیر شتاب و سرعت زاویه‌ای در سه جهت محور مختصات است. پروتکل ارتباطی این مژول برای برقراری ارتباط با پردازنده I2C بوده و وجود پردازشگر حرکتی دیجیتال (DMP) در این مژول به دلیل جمع‌آوری و پردازش داده‌ها از شتابسنج و ژیروسکوپ باعث می‌شود بار محاسباتی از پردازشگر میزبان، مانند آردوینو و ... کم شود.

معرفی پایه ها (Pinout) مژول GY-521 سنسور MPU6050



ماژول GY-521 دارای 8 پایه به شرح زیر است:



VCC: تغذیه مژول - 3 تا 5 ولت

GND: زمین

SCL: پایه کلاک پروتکل I2C

SDA: پایه دیتا پروتکل I2C

XDA: پایه دیتا پروتکل I2C کمکی

XCL: پایه کلاک پروتکل I2C کمکی

ADO: تنظیمات آدرس پروتکل I2C

INT: وقفه

کنترل موتور DC

برای اینکه بتوانیم کنترل کاملی بر موتور DC داشته باشیم، باید سرعت و جهت چرخش آن را کنترل کنیم. با تلفیق این دو تکنیک می‌توان به این مهم دست یافت:

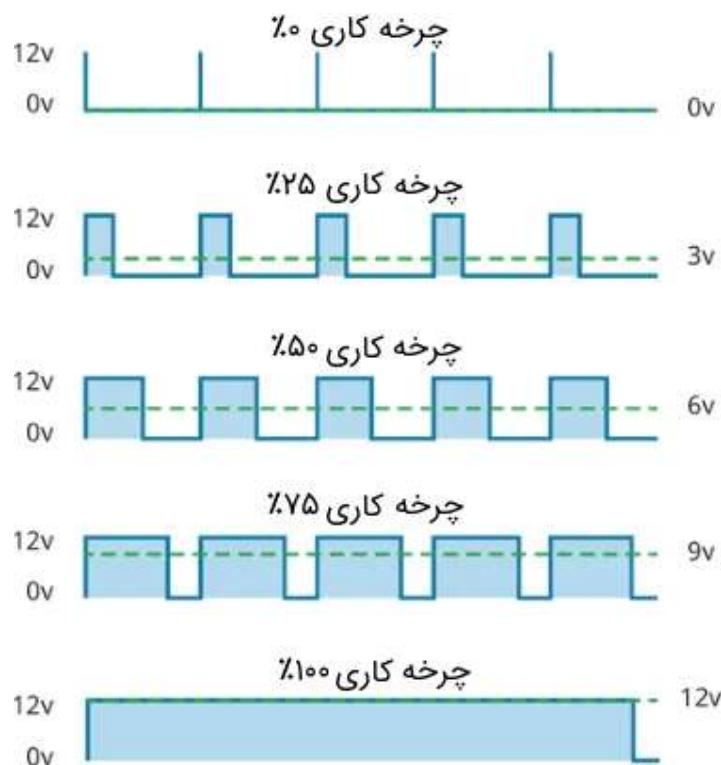
- مدولاسیون پهنه‌ای پالس (PWM) برای کنترل سرعت
- پل اج (H-Bridge) برای کنترل جهت چرخش

مدولاسیون پهنه‌ای پالس برای کنترل سرعت

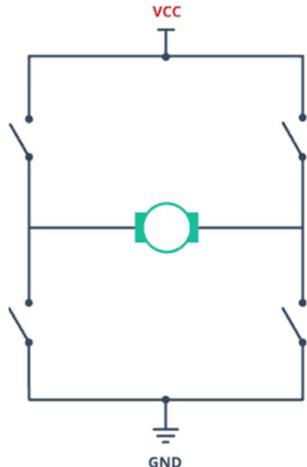
سرعت یک موتور DC را می‌توان با تغییر ولتاژ ورودی آن کنترل کرد. یک روش معمول برای انجام این کار استفاده از PWM است. مدولاسیون پهنه‌ای پالس روشی است که در آن مقدار متوسط ولتاژ ورودی با تولید دنباله‌ای از پالس‌های ON-OFF تنظیم می‌شود. ولتاژ متوسط متناسب با پهنه‌ای پالس‌ها، معروف به چرخه کاری (Duty Cycle)، است.

هر چه چرخه کاری بزرگ‌تر باشد، متوسط ولتاژ اعمال شده بر موتور DC بیشتر و در نتیجه سرعت بیشتر و هر چه چرخه کاری کمتر باشد، ولتاژ متوسط اعمال شده روی موتور DC کمتر و در نتیجه سرعت آن کمتر است.

تصویر زیر تکنیک PWM را با چرخه‌های کاری مختلف و ولتاژ متوسط متناظر با آن‌ها نشان می‌دهد.



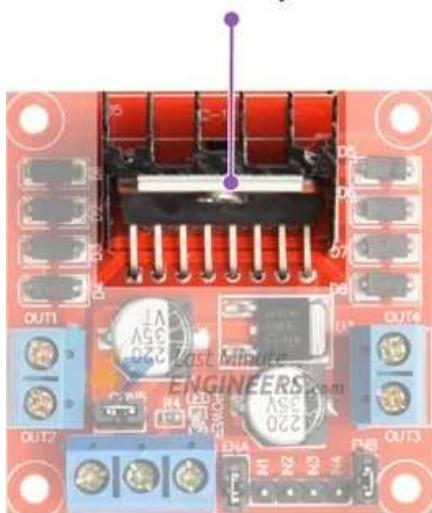
پل اج برای کنترل جهت چرخش



جهت چرخش موتور DC را می‌توان با تغییر قطب ولتاژ ورودی آن کنترل کرد. یک روش معمول برای انجام این کار استفاده از «پل اج» (H-Bridge) است. مدار پل اج شامل چهار سوئیچ است که موتور در مرکز آن قرار دارد و یک آرایش مانند حرف انگلیسی H را تشکیل می‌دهند.

بسته شدن همزمان دو کلید خاص، قطب ولتاژ اعمال شده به موتور را بر عکس می‌کند و این امر باعث تغییر جهت چرخش موتور می‌شود. تصویر متحرک زیر، عملکرد مدار پل اج را نشان می‌دهد.

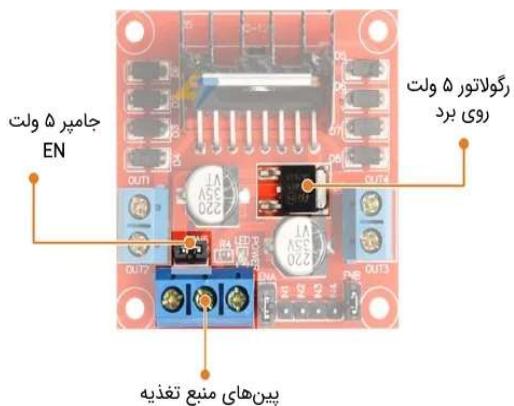
تراشه L298N



اجزای مازول:

در قلب مازول L298 تراشه بزرگ و مشکی L298N همراه با هیت سینک قرار دارد. مازول L298 و به طور دقیق‌تر، مازول L298N یک درایور موتور پل اج دو کاناله است که می‌تواند یک جفت موتور DC را کنترل کند. این بدان معنی است که مازول L298 می‌تواند به صورت جداگانه تا دو موتور را هدایت کند و این قابلیت، مازول L238 را برای ساخت کنترل‌کننده‌های رباتیک دوچرخ ایده‌آل می‌کند.

منبع تغذیه مازول:



مازول درایور موتور L298N از طریق ۳ پین می‌لی متري تغذیه می‌شود و شامل پین‌هایی برای منبع تغذیه موتور (Vs)، زمین و منبع تغذیه ۵ ولت (Vss) است.

تذکر: آیسی درایور 298N در واقع دارای دو پایه توان ورودی است "Vs" و "Vss". پل اج از پین Vs توان خود را برای هدایت موتورها دریافت می‌کند. ولتاژ این پین می‌تواند ۵ تا ۳۵ ولت باشد. ولتاژ Vss برای تغذیه مدارهای منطقی است که مقدار آن ممکن است بین ۵ تا ۷ ولت باشد. هر دوی این ولتاژ‌ها زمین مشترکی به نام "GND" دارند.

مازول تصویرداری یک رگولاتور ولتاژ M0578 با مقدار ۵ ولت است که سازنده آن اس‌تی‌مایکروالکترونیکس (STMicroelectronics) است. این رگولاتور روی برد، از طریق جامپر فعال یا غیرفعال می‌شود.

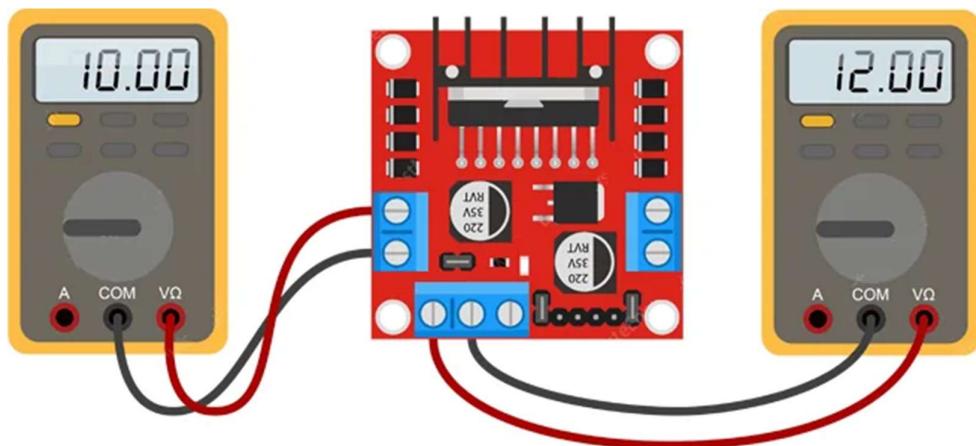
وقتی این جامپر در مدار قرار داده شود، رگولاتور 5 ولت فعال می‌شود و منبع تغذیه منطقی (V_{SS}) را از منبع تغذیه موتور (V_s) تأمین می‌کند. در این حالت، ترمینال ورودی 5 ولت به عنوان بین خروجی عمل می‌کند و 5 ولت 0,5 آمپری را تحويل می‌دهد. می‌توان از این رگولاتور برای تغذیه آردوبینو یا مدارهای دیگری که به منبع تغذیه 5 ولت نیاز دارند، استفاده کرد.

هنگامی که جامپر را برداریم، رگولاتور 5 ولت از کار می‌افتد و باید 5 ولت را از طریق ترمینال ورودی 5 ولت جداگانه تأمین کنیم.

اخطار: اگر منبع تغذیه موتور کمتر از 12 ولت باشد، می‌توانید جامپر را در در مدار قرار دهید. اگر ولتاژ بیش از 12 ولت است، باید جامپر را بردارید تا از خراب شدن رگولاتور 5 ولت جلوگیری کنید. همچنین، دقت کنید زمانی که جامپر در جای خود قرار دارد، منبع تغذیه موتور و منبع تغذیه 5 ولت را جداگانه تأمین نکنید.

افت ولتاژ ماثول L298N

اگر 12 ولت را به ترمینال منبع تغذیه موتور وصل کنیم، موتورها ولتاژ حدود 10 ولت دریافت می‌کنند. این بدین معناست که یک موتور 12 ولت DC هرگز با حداکثر سرعت خود نخواهد چرخید.

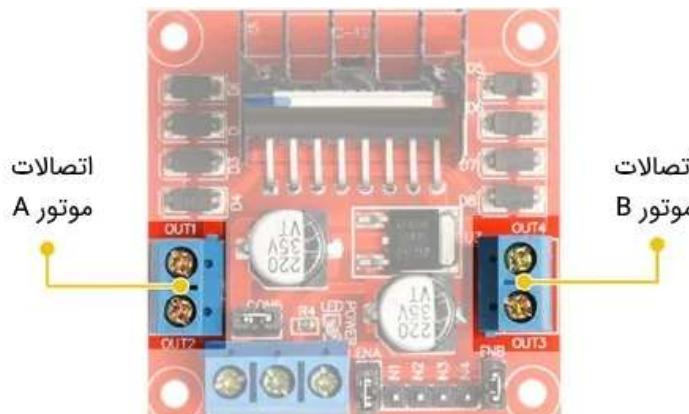


برای خارج شدن حداکثر سرعت از موتور ، منبع تغذیه موتور باید کمی ولتاژ بالاتر (2 ولت) از ولتاژ واقعی موتور باشد.

با توجه به افت ولتاژ 2 ولت، اگر از موتورهای 5 ولت استفاده می‌کنید، باید 7 ولت در ترمینال منبع تغذیه موتور تأمین کنید. اگر موتور 12 ولت دارد، ولتاژ تغذیه موتور شما باید 14 ولت باشد.

L298 پین‌های خروجی مازول

کانال‌های خروجی درایور موتور L298N برای موتور A و B با دو پین 3,5 میلیمتری به لبه مازول متصل می‌شوند.

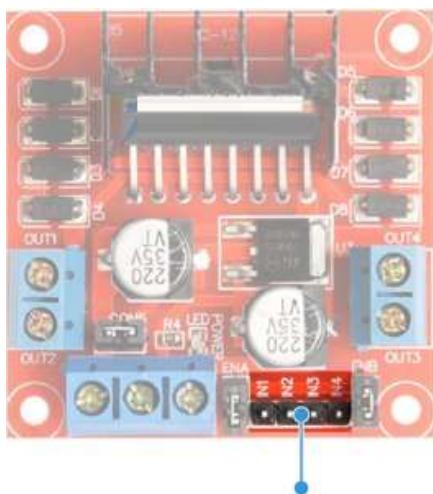


می‌توانید دو موتور DC با ولتاژ بین 5 تا 35 ولت را به این ترمینال‌ها متصل کنید. هر کانال روی مازول می‌تواند حداکثر A2 به موتور DC تحویل دهد. با این حال، میزان جریان تحویل داده شده به موتور به منبع تغذیه سیستم بستگی دارد.

L298 پین‌های کنترل مازول

برای هریک از کانال‌های L298N، دو نوع پین کنترل وجود دارد که به ما امکان کنترل همزمان سرعت و چرخش موتورهای DC را می‌دهد. پین‌های کنترل جهت و پین‌های کنترل سرعت.

پین‌های کنترل جهت در شکل مشخص شده‌اند.

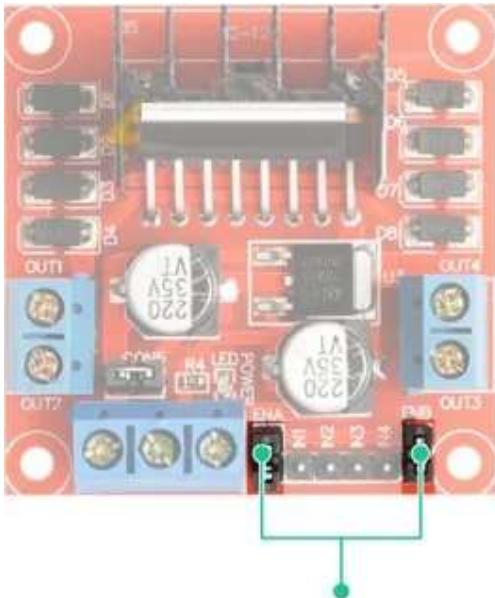


پین‌های کنترل جهت

با استفاده از پین‌های کنترل جهت، می‌توانیم موتور را در دو جهت بچرخانیم. این پین‌ها در واقع سوئیچ‌های مدار پل اج داخل آی‌سی L298N را کنترل می‌کنند. مازول می‌کنند. مازول دارای دو پایه کنترل جهت برای هر کانال است. پایه‌های IN1 و IN2 جهت چرخش موتور A را کنترل می‌کنند، در حالی که IN3 و IN4 موتور B را کنترل می‌کنند. جهت چرخش یک موتور را می‌توان با اعمال منطق HIGH (5 ولت) یا منطق LOW (زمین) به این ورودی‌ها کنترل کرد. جدول زیر نحوه انجام این کار را نشان می‌دهد.

جهت چرخش	ورودی ۲	ورودی ۱
موتور خاموش	Low (0)	Low (0)
مستقیم	Low (0)	High (1)
عکس	High (1)	Low (0)
موتور خاموش	High (1)	High (1)

پین‌های کنترل سرعت در شکل مشخص شده‌اند.



پین‌های کنترل سرعت

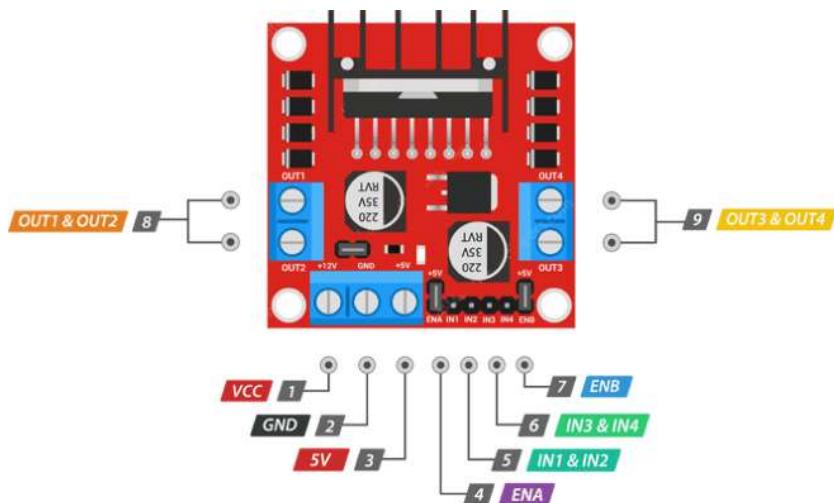
پین‌های کنترل سرعت، یعنی **ENA** و **ENB**، برای روشن یا خاموش کردن موتورها و کنترل سرعت آن‌ها استفاده می‌شوند.

با اعمال **HIGH**، این پایه‌ها موتور را می‌چرخانند و اعمال **LOW** باعث متوقف شدن آن‌ها می‌شود، اما با **PWM** می‌توانیم سرعت موتورها را کنترل کنیم.

معمولًاً جامپر در جای خود روی مازول قرار دارد. در این صورت، موتور فعال است و با حداکثر سرعت می‌چرخد. اگر بخواهیم سرعت موتورها را به صورت برنامه‌ریزی شده کنترل کنیم، باید جامپر را برداشته و به **PWM** آردوینو را به آن متصل کنیم.

پین‌های مازول L298N

شکل زیر مازول L298 و پین‌های آن را نشان می‌دهد.



- پین **VCC** توان موتور را تأمین می‌کند. ورودی این پین می‌تواند هر مقداری بین 5 تا 35 ولت باشد. به یاد داشته باشید اگر جامپر 5 ولت EN در جای خود قرار داشته باشد، برای رسیدن به حداقل سرعت، باید 2 ولت بیشتر از ولتاژ واقعی موتور را تأمین کنید.
- پین **GND** پایه زمین مشترک است.
- پین **5V** منبع تغذیه مدارهای منطقی سوئیچینگ داخل آی‌سی L298N است. اگر جامپر 5 ولت EN در جای خود باشد، این پین به عنوان یک خروجی عمل می‌کند و می‌توان از آن برای تغذیه آردوینو استفاده کرد. اگر جامپر 5 ولت EN برداشته شود، باید آن را به پایه 5 ولت آردوینو وصل کنید.
- از پین‌های **ENA** برای کنترل سرعت موتور A استفاده می‌شود. اعمال HIGH به این پایه HIGH نگه داشتن جامپر (باعت چرخش موتور A می‌شود و LOW شدن آن باعث متوقف شدن موتور می‌شود. برداشتن جامپر و اتصال ورودی PWM به این پایه، باعث می‌شود بتوانیم سرعت موتور A را کنترل کنیم.
- از پین‌های **IN1** و **IN2** برای کنترل جهت چرخش موتور A استفاده می‌شود. هنگامی که یکی از آن‌ها HIGH و دیگری LOW باشد، موتور A می‌چرخد. اگر هر دو ورودی HIGH یا LOW باشند، موتور A متوقف می‌شود.
- پین‌های **IN3** و **IN4** برای کنترل جهت چرخش موتور B به کار می‌روند. وقتی یکی از این پین‌ها HIGH باشد و دیگری LOW، موتور B می‌چرخد. اگر هر دو ورودی HIGH یا LOW باشند، موتور B متوقف می‌شود.
- از پین‌های **ENB** برای کنترل سرعت موتور B استفاده می‌شود. اعمال HIGH به این پایه HIGH نگه داشتن جامپر باعث چرخش موتور B می‌شود، LOW کردن آن باعث متوقف شدن موتور می‌شود. با برداشتن جامپر و اتصال این پایه به ورودی PWM می‌توانیم سرعت موتور B را کنترل کنیم.
- پین‌های **OUT2** به موتور A متصل می‌شوند.
- پین‌های **OUT3** و **OUT4** به موتور B متصل می‌شوند.

باتری ها

باتری های قابل شارژ بر اساس ساختار شیمیایی و سیکل شارژی به دو دسته بسیار عامه تقسیم میشوند. یکی باتری های لیتیوم یون Li-Io و دیگری باتری های لیتیوم پلیمر Li-Po میباشند. این دو مدل باتری Battery سبک و قابلیت اطمینان بالایی دارند. از این رو در زندگی روزمره ما در بسیاری از گجت ها به کار رفته اند. از موبایل، لپ تاپ، پاور بانک Power Bank ، دستگاه های پزشکی قابل حمل، تبلت و ... همگی بسته به نوع کاربردشان از یکی از این دو مدل باتری استفاده میکنند.

باتری لیتیوم یون

تولید انبوہ و کاربردی این نوع باتری از سال ۱۹۹۲ شروع شد. البته استفاده عمومی و کاربری برای عامه نداشتند. تا اینکه در سال ۱۹۹۱ توسط شرکت سونی در دوربین های هندی کم و واک من و ... از این باتری استفاده شد. به عبارتی در طی دهه ۹۰ میلادی ورودی عمومی باتری های لیتیوم یون Li-Io بین کاربران بوده است. باتری های لیتیوم یون ظرفیت انرژی بالایی دارند و به نسبت انرژی خروجی از قیمت مقرن به صرفه تری در مقایسه با باتری های لیتیوم پلیمری برخوردارند. اما باتری لیتیوم یونی عمر کوتاه تری دارند و اگر سیکل شارژ آنها درست رعایت نشود و یا پس از مدتی شارژ نشوند، سلول مربوطه از بین خواهد رود. از این رو نسبت به زمان شارژ شدن بسیار حساس خواهند بود. اخیراً مصرف این باتری ها در ساخت پاور بانک Power Bank بسیار کاربرد پیدا کرده است. از مدل های پر مصرف این باتری لیتیومی نوع ۱۸۶۵۰ میباشد.



باتری لیتیوم یون Li-Ion ۱۸۶۵۰ در دستگاه‌های مختلفی از جمله جارو برقی شارژی، دوربین‌های DSLR، دستگاه ریش تراش، فلزیاب‌های صنعتی، تلفن بیسیم خانگی، دستگاه واکی تاکی صنعتی و نظامی و... همگی از باتری لیتیوم یون استفاده می‌کنند. برخی از آن‌ها مدل Li-Ion 18650 استفاده می‌کنند. مانند تصویر بالا این باتری ظاهری استوانه‌ای دارد، ولی ابعاد آن با باتری قلمی معمولی متفاوت می‌باشد.

فعالیت شیمیایی	دارای الکترولیت
دماهی کاری	۶۰° C - ۲۰° C
کاربری	لپ تاپ، موبایل و کلا گجت‌های مصرفی
ولتاژ کاری	۳/۶ و ۷/۲ ولت
ظرفیت	تقریباً دو برابر ظرفیت باتری نیکل کادمیوم
نرخ دشارژ	یکنواخت
چرخه شارژ	۳۰۰ - ۴۰۰ سیکل برای هربار تخلیه و شارژر کامل
دماهی شارژ	۶۰° C تا ۰° C
میزان شارژ برای انبار	کمتر از ۱۰٪ درصد از ظرفیت باتری
دماهی نگهداری	۶۰° C - ۲۰° C
بازیافت	خطرناک برای محیط زیست و حتماً در زباله‌های قابل بازیافت قرار گیرد.
سایر توضیحات	<ul style="list-style-type: none"> • این نوع باتری برای استفاده درون گجتها و وسائل الکترونیکی طراحی شده است. • ساختار شیمیایی آن باعث محدودیت در شکل ظاهر سلول‌ها می‌شود.

در پرژه ربات تعادلی از چهار عدد باتری لیتیوم یون به صورت سری استفاده شده است که جریان ثابت و مقدار ولتاژ جمع می‌شود که به عدد ۱۴.۸ میرسد که همان طور که در بالا گفته شد مقدار افت ولتاژی به خاطر مصرف راه انداز موتور مصرف شده باقیمانده حدود ۱۲ ولت می‌شود که مناسب پروژه است.

موتور گیربکس DC

موتور DC چیست؟

برای تبدیل انرژی الکتریکی به انرژی مکانیکی به دستگاهی به نام الکتروموتور نیاز داریم. الکتروموتور DC، دسته ای از الکتروموتورها هستند که جریان مستقیم برق را به انرژی مکانیکی مورد نیاز برای بسیاری از دستگاه‌ها تبدیل می‌کنند.

الکتروموتور DC از قسمت‌های اصلی روتور، استاتور، کموتاتور و جاروبک تشکیل می‌شوند. استاتور در واقع نوعی آهنربای ثابت است و کوتاتور سیم پیچ مخصوصی است که با عبور جریان برق به آهنربای الکتریکی تبدیل می‌شود و با کمک کموتاتور قطب مثبت و منفی آن جا به جا می‌شوند.

در حقیقت می‌توان گفت موتورهای دی‌سی با کمک میدان مغناطیسی که از جریان الکتریکی به وجود می‌آید برای چرخش مکانیکی استفاده می‌کنند. گشتاور و سرعت خروجی در این الکتروموتورها هم به میزان جریان ورودی و هم به طراحی موتور بستگی دارد.

موتور گیربکس چیست؟

موتور گیربکس که به آن الکتروگیربکس نیز گفته می‌شود، یکی از تجهیزات صنعتی متشکل از یک گیربکس و یک موتور الکتریکی است. این وسیله صنعتی برای ایجاد گشتاوری بالا در سرعت کم کاربرد دارد. در این ترکیب، گیربکس با تغییر دور موتور عمل می‌کند. به بیانی دیگر می‌توان گفت که در موتور گیربکس‌ها، وظیفه اصلی گیربکس به عنوان یک مبدل گشتاور است. به این ترتیب، این قطعه می‌تواند برای موتورهای کوچک، گشتاورهای بزرگی را تولید کند.

به بیانی دیگر، با استفاده از موتور گیربکس‌ها دیگر نیازی به نصب جدأگانه موتور الکتریکی و گیربکس در پروژه‌های مهندسی وجود ندارد. مصرف بهینه انرژی از دیگر نقاط قوت استفاده از این وسیله صنعتی به شمار می‌آید. حذف تلفات انرژی به دلیل عدم نیاز به تراز کردن و کوپلینگ موتور گیربکس‌ها نیز از مزایای دیگر کاربرد این قطعه محسوب می‌شود. سایر مزایای موتور گیربکس‌ها عبارتند از:



- کاهش آلدگی صوتی و افزایش طول عمر
- امکان انتقال گشتاور و سرعت بالا به دلیل طراحی دقیق دنده‌های چرخ دنده‌ها
- ترکیب بهینه گیربکس و الکتروموتور



VS



 Efficiency
 Low Price

 Prescision
 Torque
 Efficiency

در مورد استفاده از موتور گیربکس دار با انکدر اثر هال در پروژه باید گفت که در وله اول مزایای این موتور عبارت است از قیمت نسبتاً پایین تر نسبت به سرو موتور ها و بعدی نیز کاربردی بودن و داشتن سنسور اثر هال است ولی در طرف دیگر قضايا در موتور های سرو مقدار تنظیم پذیری بالاتر بوده و دقیق تر میتوان شافت موتور را کنترل کرد و مقدار گشتاور نیز بهره بهتری دارد ولی مقدار بهره وری نسبتاً پایین تری را شاهد هستیم.

موتور استفاده شده در پروژه با پارت نامبر **JGA25** می باشد.

مشخصات موتور عبارتند از:

Name: JGA25-13CPR DC geared motor
Voltage: 6-12V
Nominal Voltage: 12V
Pulse: 26 / ring
Speed: 240rpm(6V), 352rpm(12V)
Shaft diameter: 4mm
Length: 12mm
Encoder motor end 11 signals
Terminal connection length 20cm



Load Torque:

Torque: 0.85 - 1.2kg.cm
Stall Torque: 5.6 - 8 kg. cm

Connection:

red: motor positive
Black: negative Motor
Green: Encoder negative
Blue: encoder positive
white and yellow: motor A-phase and B-phase



مقدار ولتاژ کاری موتور از 6 ولت تا 12 ولت است که ولتاژ مناسب نامی 12 ولت است.

سرعت موتور در 12 ولت حدود 350 دور در دقیقه است، اندازه شافت موتور 4 میلیمتر و طول آن 1.2 سانتی متر است.

مقدار اندازه گشتاور از 850 گرم تا 1.2 کیلوگرم در سانتی متر است.

سیم های اتصال موتور نیز

رنگ قرمز مقدار ولتاژ 12 ولت

رنگ مشکی اتصال به زمین تغذیه

رنگ سبز مقدار منفی انکدر

رنگ ابی مقدار مثبت انکدر

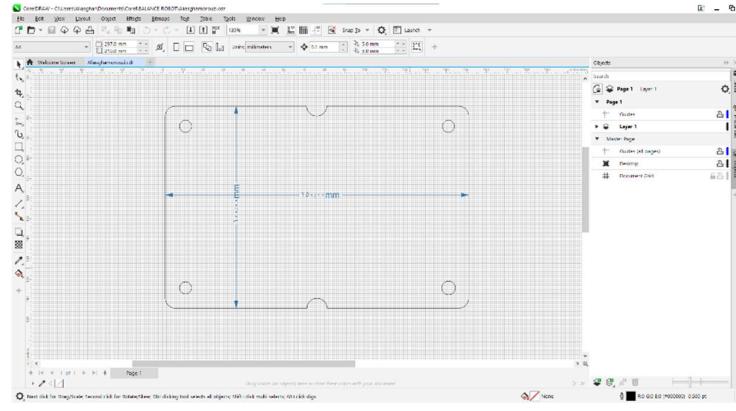
رنگ های سفید و مشکی نیز نشان دهنده فاز حرکتی موتور است که نوع A و نوع B میباشد.

طراحی بدنه و شاسی و اتصالات ربات

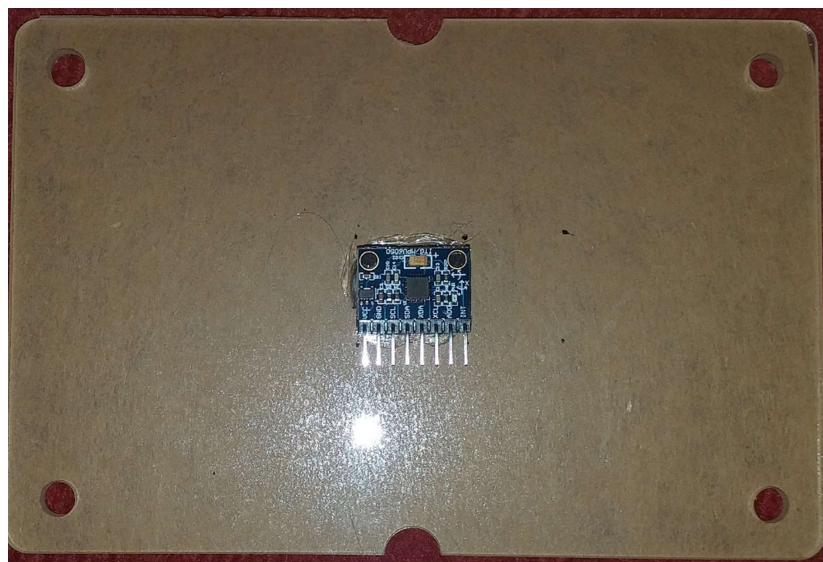
طراحی بدنه و شاسی ربات توسط نرم افزار کورل دراو (Corel draw) انجام شده است.
صفحه های ربات توسط این برنامه طراحی شد و سپس توسط دستگاه برش لیزر روی ورق های پلکسی 3 میلی متری شیشه ای توسط دستگاه اعمال شد.



نمونه دستگاه برش لیزری

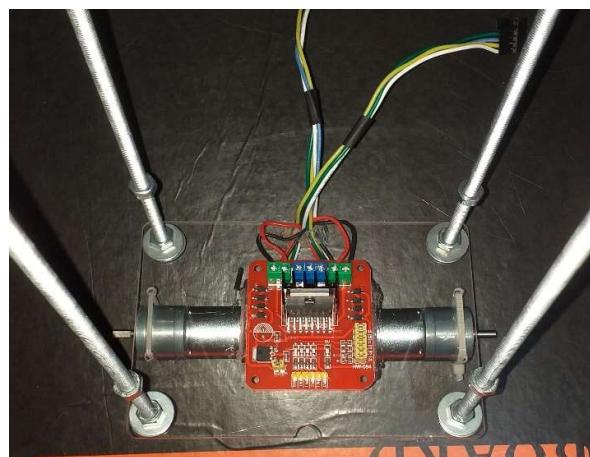
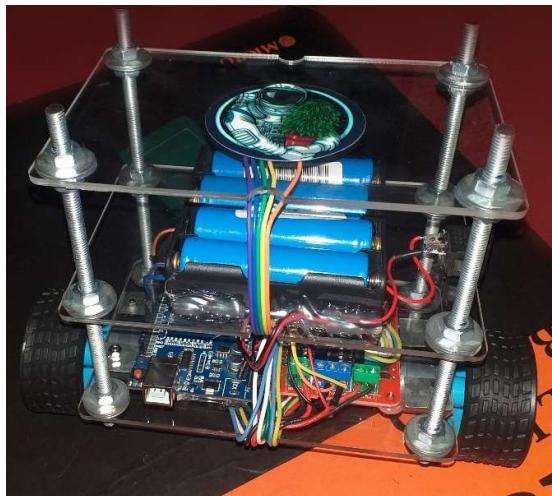
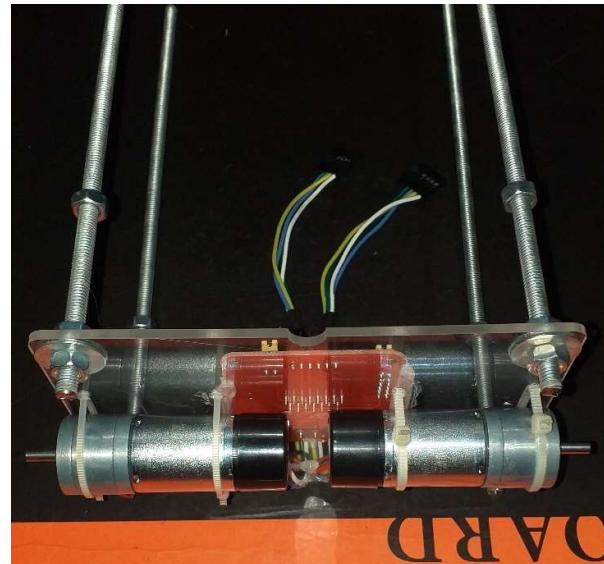


عکس از ورق اماده شده:



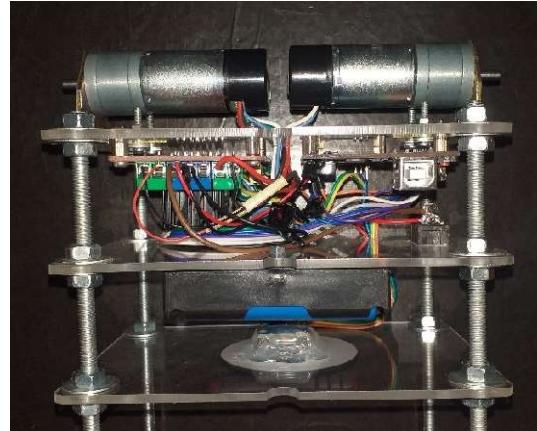
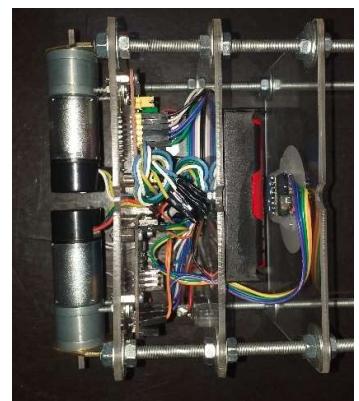
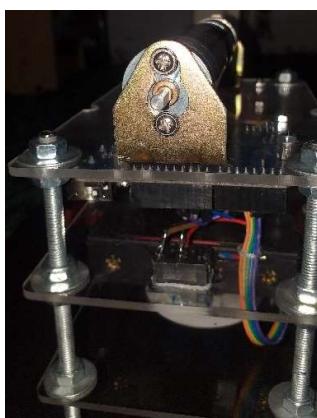
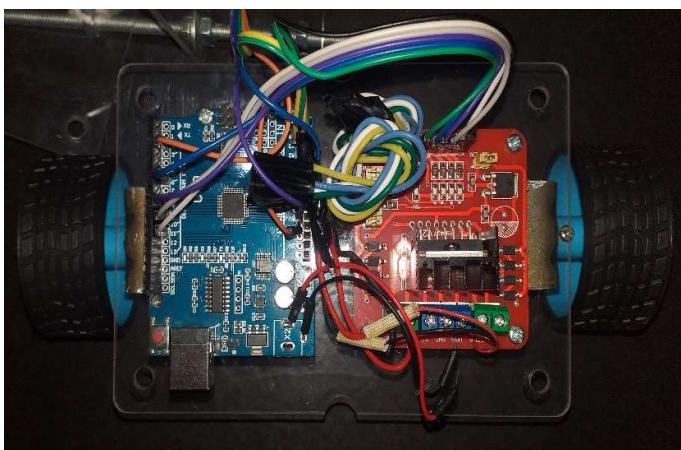
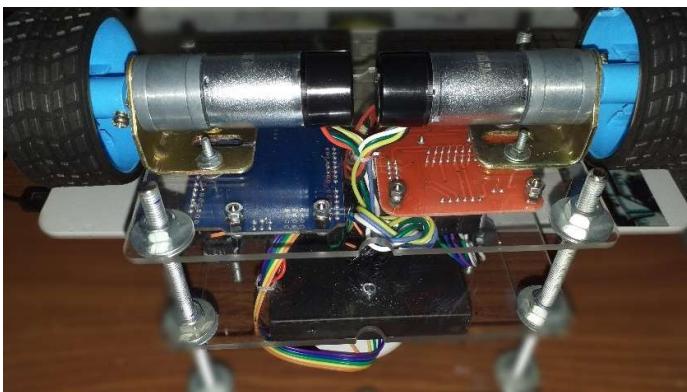
تصاویری از ورق ها و اتصالات فلزی

نمونه اولیه ساخته شده برای تست اول:

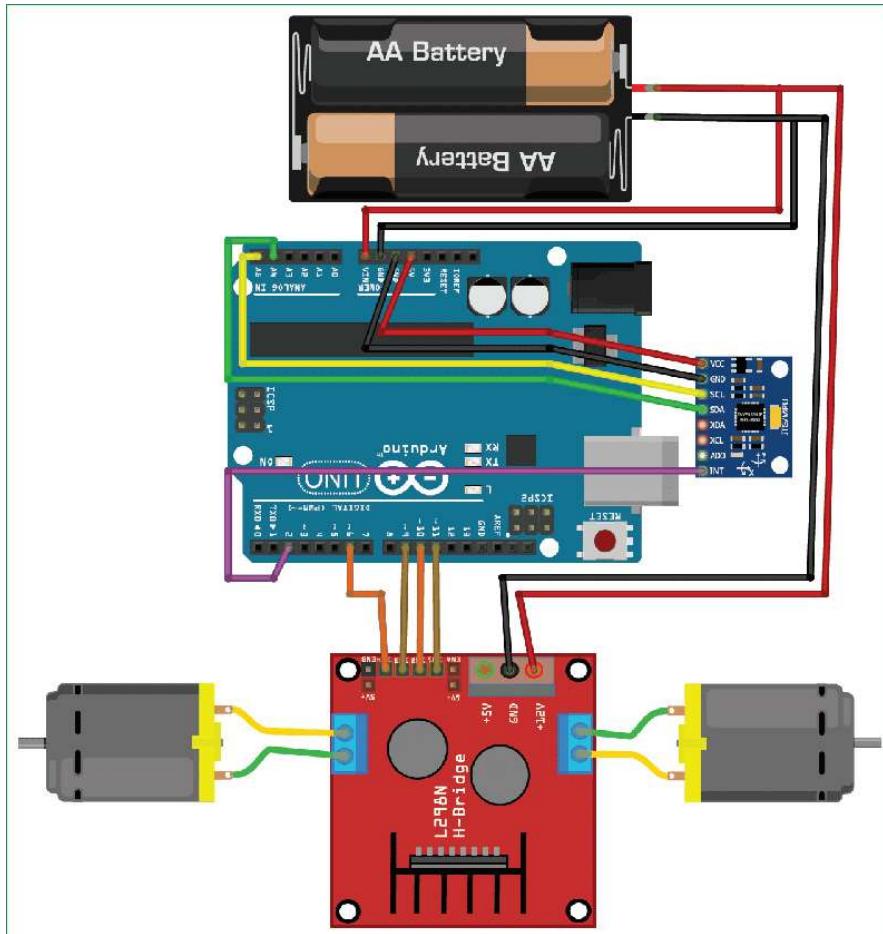


پس از ساخت نمونه اولیه و تست آن و اندازه گیری مقادیر سپس دوباره اتصالات تعویض شد و از اتصالات فلزی با استحکام بیشتر استفاده شد و برای عبور سیم ها نیز از داخل بدنه سوراخ هایی ایجاد شد که به مقاومت بیشتر ربات و نیز به لرزش گیری حین کار ربات افزوده شد.

نمونه دوم ساخته شده نهایی :



اتصالات مدار



آردوینو و درایور موتور مستقیماً توسط باتری تغذیه می شوند. موتورهای DC می توانند از ولتاژ 5 تا 12 ولت استفاده کنند. تغذیه موتورها توسط درایور نیز تامین می شود. تغذیه مازول IMU توسط آردوینو تامین می شود. در جدول و دیاگرام نحوه اتصالات نیز مشخص شده است. مازول از طریق پروتکل I2C با آردوینو ارتباط برقرار می کند. بنابراین باید به پین های آردوینو (A4,A5) متصل شود. موتورهای DC نیز به پین های PWM یعنی پین های D6 ، D9 و D11 متصل می شوند. زیرا لازم است با تغییر Duty Cycle سیگنال های PWM سرعت موتور DC کنترل شود .

در شکل بالا با توجه به توضیحات داده شده بر روی درایور موتور ها دو پایه enable(EN) بر روی درایور نیز به 3.3 ولت برد آردوینو بر روی ربات اتصال داده شده است تا موتور ها در حالت روشن قرار گیرند.

کنترل کننده PID

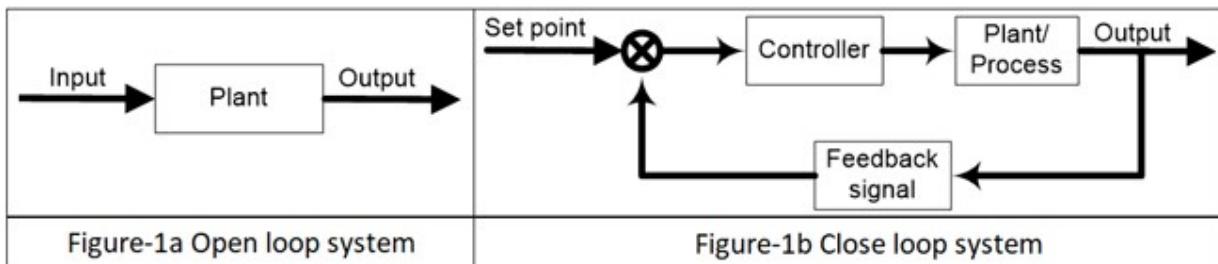
کنترل کننده PID مخفف Proportional نسبی و Integral انتگرالی و Derivative مشتقی مبادله می‌باشد.

تصور کنید ربات در حال تمیز کردن سطوح است که به یک راه پله نزدیک می‌شود. یک حسگر مجاورت در زیر ربات قرار دارد که این وضعیت را تشخیص می‌کند و برق موتور را قطع می‌کند. اما این ربات فوراً متوقف نمی‌شود و در صورت وقوع این اتفاق، ممکن است ربات از پله ها سقوط کند و آسیب ببیند. حالا تصور کنید یک خودرو هوشمند دارید و می‌خواهید آن را در موقعیت‌های خاص متوقف کنید. این مسئله بدون استفاده از کنترلر PID بسیار دشوار است زیرا اگر برق را قطع کنید ماشین هدف خود را از دست میدهد.

دو نوع سیستم وجود دارد: سیستم حلقه باز و سیستم حلقه بسته. همچنین سیستم حلقه باز به عنوان سیستم کنترل نشده و سیستم حلقه بسته به عنوان سیستم کنترل شده شناخته می‌شود.

در سیستم حلقه باز (open loop)， خروجی کنترل نمی‌شود زیرا این سیستم فیدبکی ندارد و در یک سیستم حلقه بسته، خروجی با کمک کنترل کننده کنترل می‌شود و این سیستم به یک یا چند مسیر فیدبک نیاز دارد.

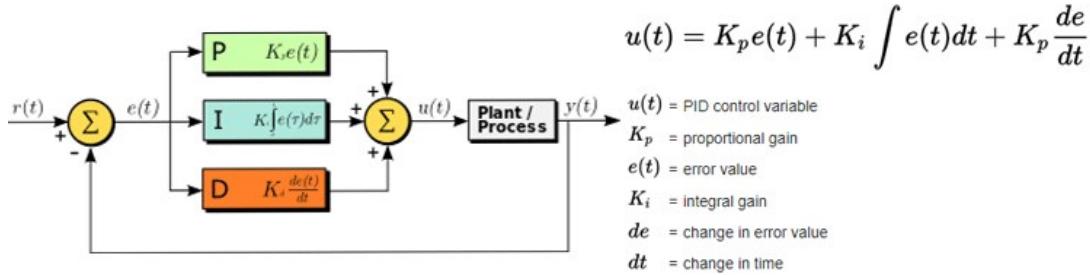
یک سیستم حلقه باز بسیار ساده است اما در کاربردهای کنترل صنعتی استفاده نمی‌شود زیرا این سیستم کنترل نشده است. سیستم حلقه بسته پیچیده است اما برای کاربردهای صنعتی بسیار مفید است، زیرا در این سیستم خروجی می‌تواند در مقدار دلخواه پایدار باشد، PID نمونه‌ای از سیستم حلقه بسته است. بلوک دیاگرام این سیستم‌ها در شکل 1 نشان داده شده است.



سیستم حلقه بسته همچنین بعنوان سیستم کنترل فیدبک نیز معروف است و از این نوع سیستم برای طراحی سیستم پایدار اتوماتیک در خروجی یا مرجع (ورودی) استفاده می‌شود. به همین دلیل، سیگنال خطایی ایجاد می‌کند. سیگنال خطای (t) e و تفاوت بین خروجی (t) y و سیگنال مرجع (t) u است. وقتی این خطای صفر باشد به این معنی است که خروجی مورد نظر حاصل شده و در این شرایط خروجی همان سیگنال مرجع است.

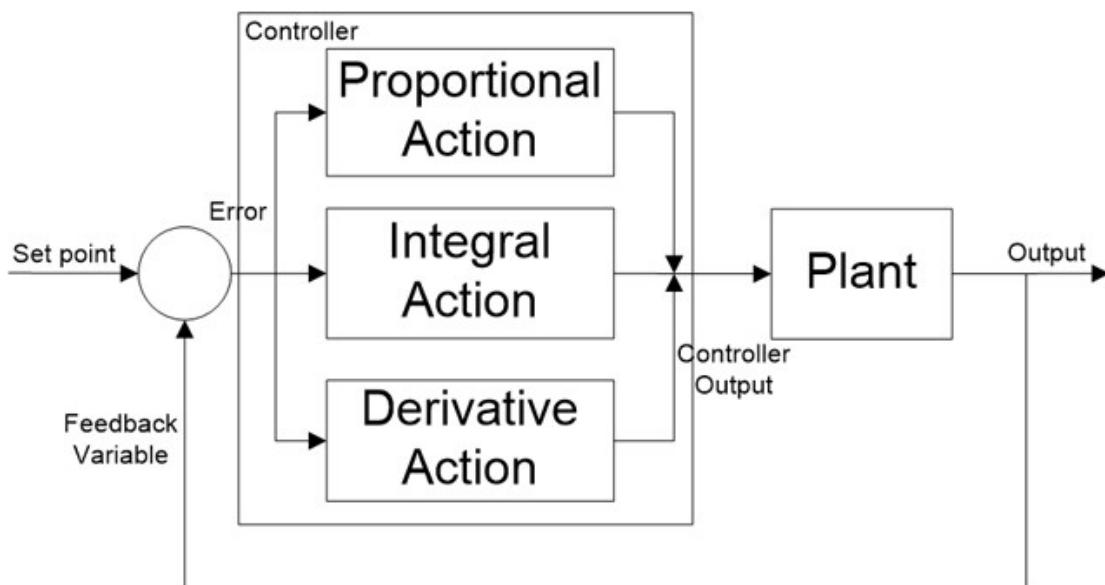
برای مثال، یک خشک کن چندین بار در حال انجام یک کار است که زمان آن از قبل تنظیم می‌شود. وقتی خشک کن روشن شد، تایмер شروع می‌شود و تا پایان تایمر خشک کن کار می‌کند و خروجی می‌دهد (پارچه خشک). این یک سیستم حلقه باز ساده است، جایی که خروجی نیازی به کنترل ندارد و به هیچ مسیر فیدبکی نیاز ندارد. در این سیستم، ما از یک سنسور رطوبت استفاده می‌کنیم که مسیر فیدبک را ارائه می‌دهد و این را با نقطه تنظیم مقایسه می‌کند و یک خطای ایجاد می‌کند. خشک کن تا زمانی که این خطای صفر باشد اجرا می‌شود. این به این معنی است که وقتی رطوبت

پارچه همان نقطه تنظیم شده باشد، خشک کن از کار می‌افتد. در سیستم حلقه باز، خشک کن بدون در نظر گرفتن خشکی یا مرطوب بودن لباس برای مدت زمان مشخصی کار خواهد کرد. اما در سیستم حلقه بسته، خشک کن برای مدت زمان مشخصی کار نمی‌کند بلکه تا خشک شدن لباس کار می‌کند. این مزیت سیستم حلقه بسته و استفاده از کنترل کننده است.



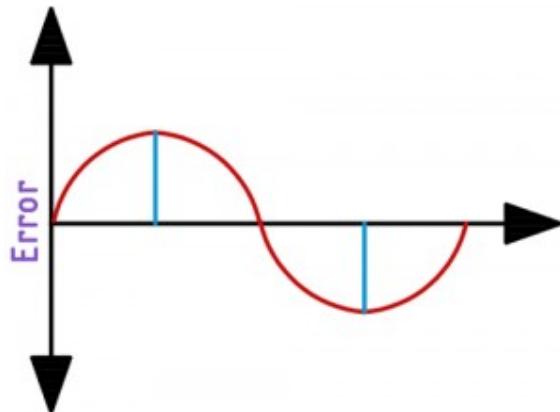
کنترل کننده PID به طور جهانی پذیرفته شده است و معمولاً در کاربردهای صنعتی از کنترلر استفاده می‌شود زیرا کنترل کننده PID ساده است و پایداری خوب و پاسخ سریع را فراهم می‌کند. PID مخفف تناسبی، انتگرال، مشتق است. در هر کاربرد، ضریب این سه عمل متفاوت است تا پاسخ و کنترل بهینه به دست آید. ورودی کنترل کننده سیگنال خطاست و خروجی به دستگاه یا فرایند داده می‌شود. تلاش بر این است که سیگنال خروجی کنترل کننده به گونه‌ای تولید شود که خروجی دستگاه به مقدار دلخواه برسد.

کنترل کننده PID یک سیستم حلقه بسته است که دارای سیستم کنترل فیبدک است و متغیر فرایند (متغیر فیبدک) را با نقطه تنظیم مقایسه می‌کند و یک سیگنال خطاب تولید می‌کند و بر اساس آن خروجی سیستم را تنظیم می‌کند. این فرآیند تا زمانی ادامه می‌یابد که این خطاب به صفر برسد یا مقدار متغیر فرایند برابر با نقطه تنظیم شود.



انواع مد کنترلی

پاسخ تناسبی (P):



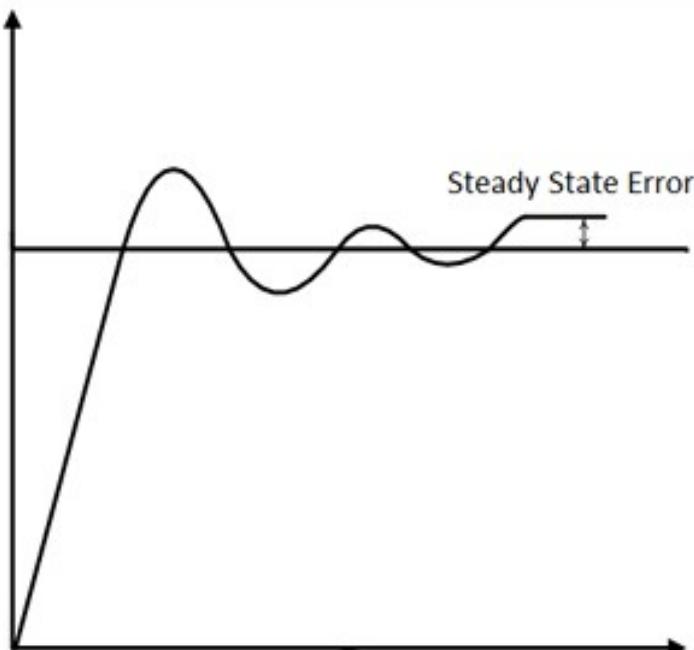
Proportional Controller

اصطلاح "P" متناسب با مقدار واقعی خطا است. اگر خطای زیاد باشد، خروجی کنترل نیز بزرگ است و اگر خطای کم باشد، خروجی کنترل نیز کوچک است و ضریب بهره (K_p) است.

همچنین در نظر بگیرید که سرعت پاسخ نیز با ضریب بهره تناسبی (K_p) متناسب است. بنابراین، سرعت پاسخ با افزایش مقدار K_p افزایش می‌یابد اما اگر K_p بیش از حد نرمال افزایش یابد، متغیر فرآیند با سرعت بالا شروع به نوسان می‌کند و سیستم را ناپایدار می‌کند.

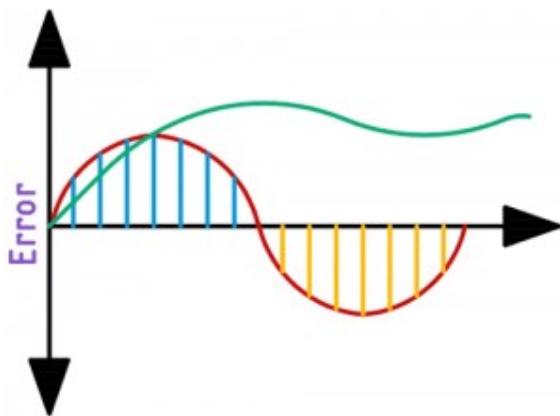
$$y(t) \propto e(t)$$

$$y(t) = k_i * e(t)$$



در اینجا، خطای حاصل با ضریب بهره تناسبی (ثابت تناسبی) ضرب می‌شود، همانطور که در معادله بالا نشان داده شده است. اگر فقط از کنترل کننده P استفاده شود، در آن زمان، به تنظیم مجدد سنتی نیاز دارد زیرا خطای حالت پایدار را حفظ می‌کند.

پاسخ انتگرالی (I):

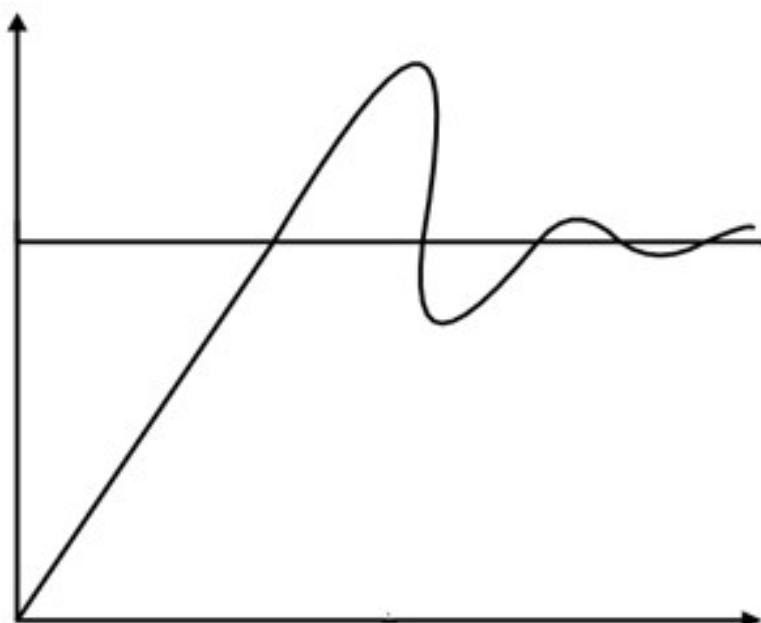


Integral Controller

به طور کلی از کنترل کننده انتگرالی برای کاهش خطای حالت پایدار استفاده می شود. اصطلاح "I" انتگرال مقدار واقعی خطا است (بر اساس زمان). با گرفتن انتگرال، مقدار بسیار ناچیزی از خطا بدست می آید که منجر به پاسخ انتگرالی بسیار بزرگی می شود. عملکرد کنترل کننده انتگرالی تا زمان صفر شدن خطا همچنان ادامه می یابد.

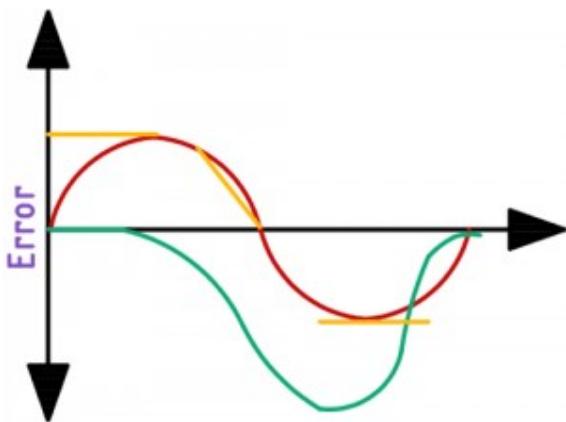
$$y(t) \propto \int e(t)$$

$$y(t) = k_i \int e(t)$$



بهره انتگرالی با سرعت پاسخ نسبت معکوس دارد. با افزایش K_i ، سرعت پاسخ کاهش می یابد. از ترکیب کنترل کننده های تناسبی و انتگرالی (کنترل کننده PI) برای سرعت خوب پاسخ و پاسخ حالت پایدار استفاده می شود.

پاسخ مشتقی (D):

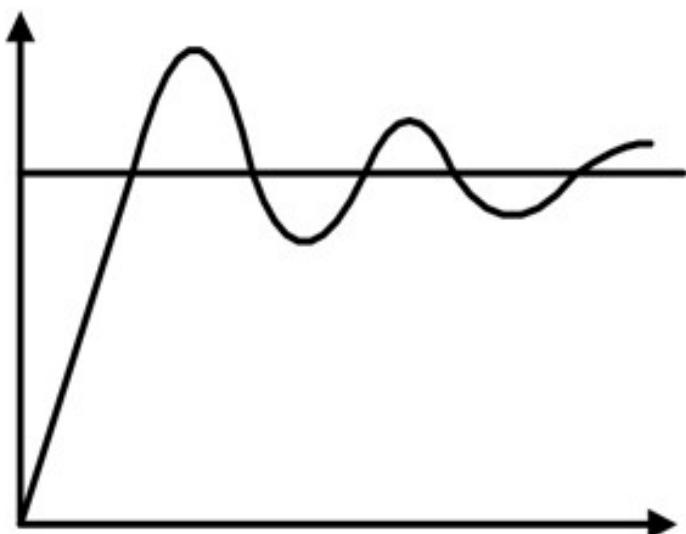


Derivative Controller

کنترل کننده مشتقی با ترکیبی از PID یا PD استفاده می شود. هرگز به تنهایی استفاده نمی شود، زیرا اگر خط ثابت باشد (غیر صفر باشد)، خروجی کنترل کننده صفر خواهد بود. در این شرایط، کنترل کننده خطای صفر را نشان می دهد، اما در واقع برخی خطاهای ثابت وجود دارند. خروجی کنترل کننده مشتقی مستقیماً متناسب با میزان تغییر خطا نسبت به زمان است که در معادله نشان داده شده است. با حذف علامت تناوب، ثابت بهره مشتقی را (K_d) بدست می آوریم. به طور کلی، کنترل کننده مشتقی هنگامی استفاده می شود که متغیرهای پردازنده شروع به نوسان کنند یا با سرعت بسیار بالایی تغییر کنند. این کنترل کننده همچنین برای پیش بینی رفتار آینده خطا توسط منحنی خطا استفاده می شود. معادله ریاضی آن به شرح زیر است.

$$y(t) \propto de(t) / dt$$

$$y(t) = K_d * de(t) / dt$$



مد های کنترلی ترکیبی

کنترل کننده تناسبی و انتگرالی:

این ترکیبی از کنترل کننده P و I است. خروجی کنترل کننده جمع دو پاسخ (تناسبی و انتگرالی) است. معادله ریاضی آن به شرح زیر است.

$$y(t) \propto (e(t) + \int e(t) dt)$$

$$y(t) = k_p * e(t) + k_i \int e(t) dt$$

کنترل کننده تناسبی و مشتقی:

این ترکیبی از کنترل کننده P و D است. خروجی کنترل کننده جمع پاسخ های تناسبی و مشتقی است. معادله ریاضی کنترل کننده PD به شرح زیر است.

$$y(t) \propto (e(t) + de(t)/dt)$$

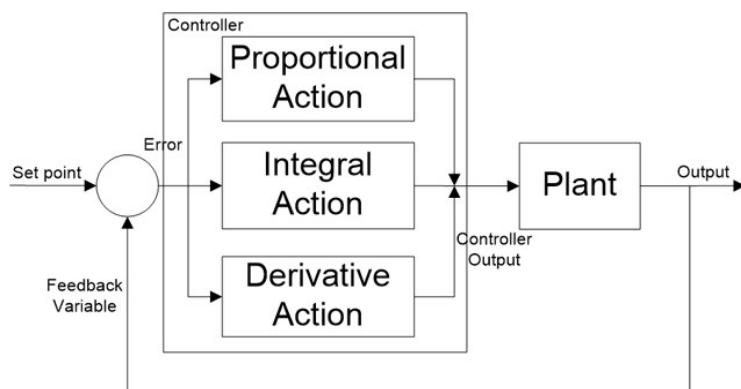
$$y(t) = k_p * e(t) + k_d * de(t)/dt$$

کنترل کننده تناسبی، انتگرالی و مشتقی:

این ترکیبی از هر سه کنترل کننده P، I و D است. خروجی کنترل کننده جمع پاسخ های تناسبی، انتگرالی و مشتقی است. معادله ریاضی کنترل کننده PID به شرح زیر است.

$$y(t) \propto (e(t) + \int e(t) dt + de(t)/dt)$$

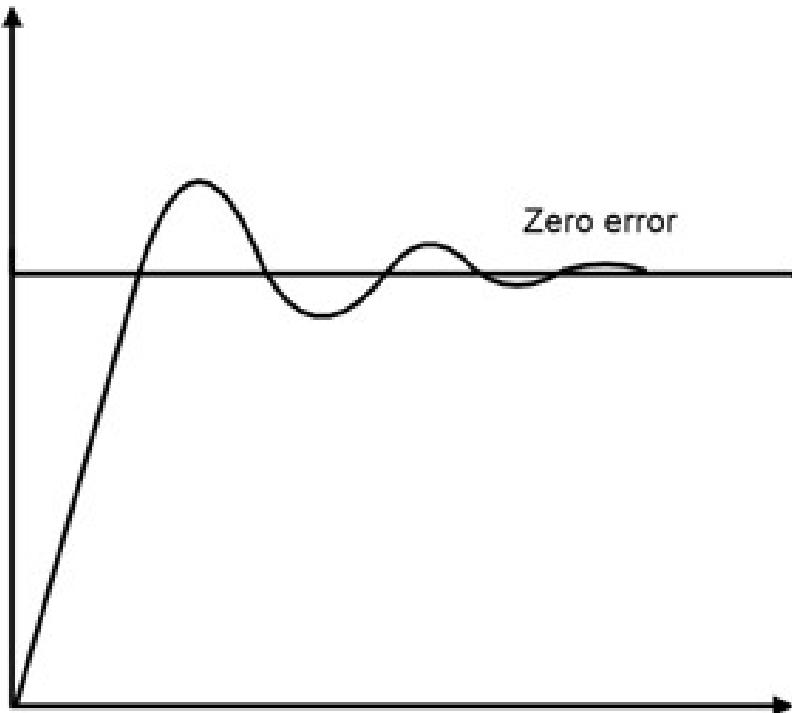
$$y(t) = k_p * e(t) + k_i \int e(t) dt + k_d * de(t)/dt$$



بنابراین، با ترکیب پاسخ کنترلی تناسبی، انتگرالی و مشتقی، یک کنترل کننده PID تشکیل می شود.

روش های تنظیم مقادیر PID

برای خروجی دلخواه، این کنترل کننده باید به درستی تنظیم شود. فرآیند دریافت پاسخ ایده آل از کنترل کننده PID با تنظیم کنترل کننده نامیده می شود. تنظیم PID به معنی تنظیم مقدار بهینه بهره تنسابی (k_p)، مشتقی (k_d) و انتگرالی (k_i) است. کنترل کننده PID تنظیم شده است برای رد اختلال به معنای ماندن در یک نقطه تنظیم و رديابی دستور است، به اين معنی که اگر نقطه تنظیم تغییر کند، خروجی کنترل کننده از نقطه تنظیم جدید پیروی می کند. اگر کنترل کننده به درستی تنظیم شود، خروجی کنترلر با نوسان کمتر و میرایی کمتر، از تنظیمات متغیر پیروی می کند.



روش های مختلفی برای تنظیم کنترل کننده PID و دریافت پاسخ مطلوب وجود دارد. روش های تنظیم کنترل به شرح زیر است:

روش آزمون و خط

تکنیک منحنی واکنش فرآیند

روش زیگلر-نیکولز

روش رله

استفاده از نرم افزار

روش آزمون و خطا:

روش آزمون و خطا به عنوان روش تنظیم دستی نیز شناخته می شود و این روش ساده ترین روش است. در این روش ابتدا مقدار kp را افزایش دهید تا زمانی که سیستم به پاسخ نوسانی برسد اما سیستم نباید ناپایدار شود و مقدار kd و ki را صفر نگه دارید. بعد از آن مقدار ki را به گونه ای تنظیم کنید که، نوسان سیستم متوقف شود. بعد از آن مقدار kd را برای پاسخ سریع تنظیم کنید.

تکنیک منحنی واکنش فرآیند:

این روش به روش تنظیم کوهن-کوون نیز معروف است. در این روش ابتدا یک منحنی واکنش فرآیند در پاسخ به یک اختلال ایجاد می کند. با استفاده از این منحنی می توان مقدار بهره کنترل کننده، زمان انتگرال و زمان مشتق را محاسبه کرد. این منحنی با انجام دستی در مرحله حلقه باز فرآیند مشخص می شود. پارامتر مدل می تواند با مرحله اولیه اختلال درصدی را پیدا کند. از این منحنی باید شبیب، زمان نشست و زمان صعود منحنی را پیدا کنیم که چیزی جز مقدار kp ، ki و kd نیست.

روش زیگلر-نیکولز:

در این روش همچنین ابتدا مقدار ki و kd را روی صفر تنظیم کنید. بهره تناسبی (kp) افزایش می یابد تا زمانی که به حداقل بهره (k_u) برسد. بهره نهایی چیزی نیست جز بهره ای که در آن خروجی حلقه شروع به نوسان می کند. این k_u و دوره نوسان T_u برای بدست آوردن بهره کنترل کننده PID از جدول زیر بدست می آید.

kd	ki	kp	نوع کنترل کننده
•	•	$k_u \cdot 0.5$	P
•	$k_u/T_u \cdot 0.54$	$k_u \cdot 0.45$	PI
$k_u T_u / 40 \cdot 3$	$k_u/T_u \cdot 1.2$	$k_u \cdot 0.60$	PID

روش رله:

روش رله همچنین بعنوان روش Astrom-Hugglund شناخته می شود. در اینجا خروجی بین دو مقدار از متغیر کنترل سوئیچ می شود اما این مقادیر به گونه ای انتخاب می شوند که فرآیند باید از نقطه تنظیم عبور کند. وقتی متغیر فرآیند کمتر از نقطه تنظیم باشد، خروجی کنترل روی مقدار بالاتر تنظیم می شود.

هنگامی که مقدار فرآیند از نقطه تنظیم بیشتر باشد، خروجی کنترل روی مقدار پایین تنظیم می شود و شکل موج خروجی تشكیل می شود. دوره و دامنه این شکل موج نوسانی اندازه گیری شده و برای تعیین بهره نهایی k_u و دوره T_u استفاده می شود که در روش بالا بکار گرفته می شود.

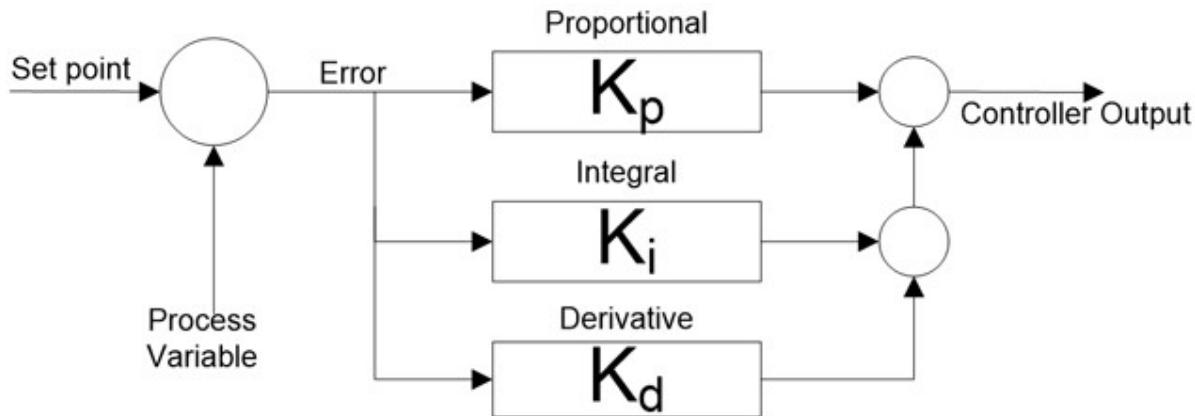
استفاده از نرم افزار:

برای تنظیم PID و بهینه سازی حلقه، بسته های نرم افزاری موجود هستند. این بسته های نرم افزاری داده ها را جمع آوری می کنند و یک مدل ریاضی از سیستم می سازند. با استفاده از این مدل، نرم افزار یک پارامتر تنظیم بهینه را از تغییرات مرجع پیدا می کند.

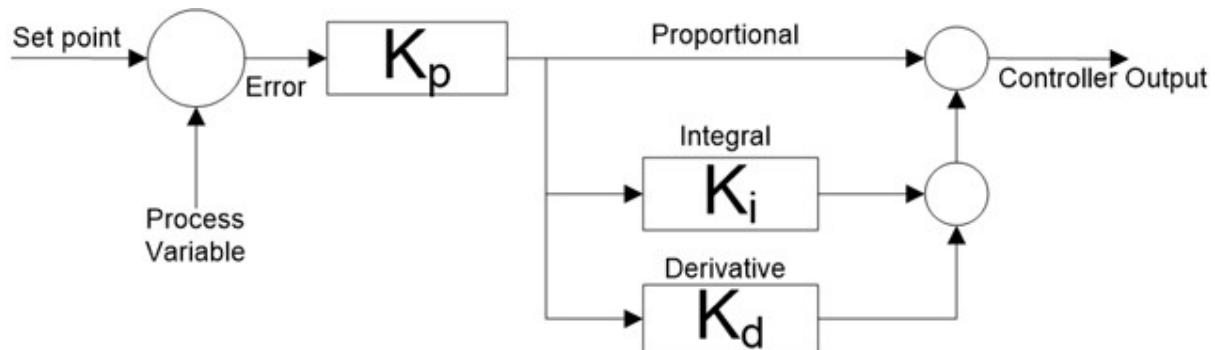
ساختار کنترل کننده های PID

کنترل کننده های PID بر اساس فناوری ریزپردازنده طراحی شده اند. صنایع مختلف از ساختار و معادله PID مختلفی استفاده می کنند. معادلات PID مورد استفاده عبارتند از: معادله PID موازی، ایده آل و سری.

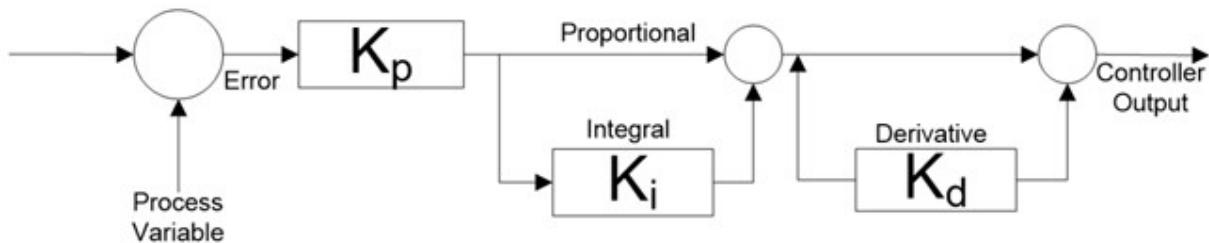
در معادله PID موازی، عمل های تناسبی، انتگرالی و مشتقی به طور جداگانه با یکدیگر کار می کنند و اثر ترکیبی این سه عمل در سیستم انجام می شود. بلوک دیاگرام این نوع PID به شرح زیر است:



در معادله PID ایده آل، بهره K_p ثابت بدست آمده در تمام روابط توزیع می شود. بنابراین، تغییرات در K_p بر سایر روابط معادله تأثیر می گذارد.



در معادله PID سری، ثابت بدست آمده در تمام عبارات معادله PID ایده آل توزیع می شود، اما در این معادله ثابت انتگرال و مشتق بر عملکرد تناسبی تأثیر دارند.



معرفی کتابخانه ها

برنامه نویسی ربات در محیط توسعه خود آردوینو یعنی Arduino IDE انجام شده است که بخشی از کد ها بخشی از پروژه آماده بوده است فلذا دانش زبان سی برای خواندن کد و مشخص کردن بخش های قابل تنظیم و ارتقاء دارای اهمیت بود.

به شرح کار هر کتابخانه پرداخته و کد نیز در ادامه تحلیل میشود:

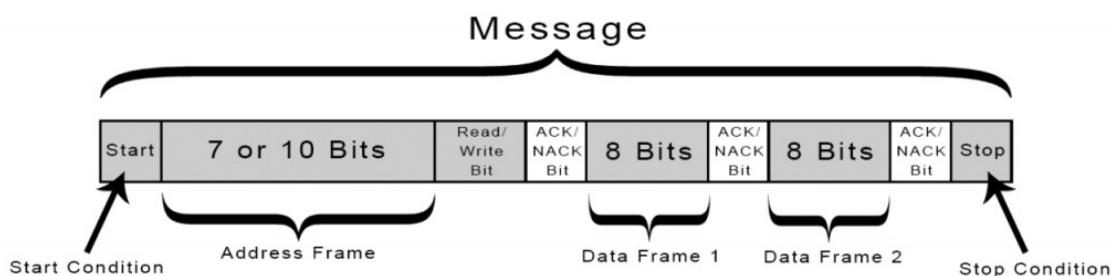
ارتباط I2C

کتابخانه های موجود در پروژه از چندین پارت تشکیل شده است که شامل کتابخانه ارتباط سریال I2C که برای خواندن دیتا های ارسالی از سنسور ژیروسکوپ استفاده میشود.

ارتباط I2C از طریق SMBus Phillips Semiconductor ساخته شد و سالها بعد اینتل پروتکل I2C را به عنوان I2C تعریف کرد. I2C بهترین ویژگی های SPI و UART را با هم ترکیب می کند. با استفاده از I2C می توانید چندین Slave به یک Master واحد متصل کنید (مانند SPI) و همچنین می توانید چندین Master را کنترل کنید که یک یا چند Slave را کنترل کنند.

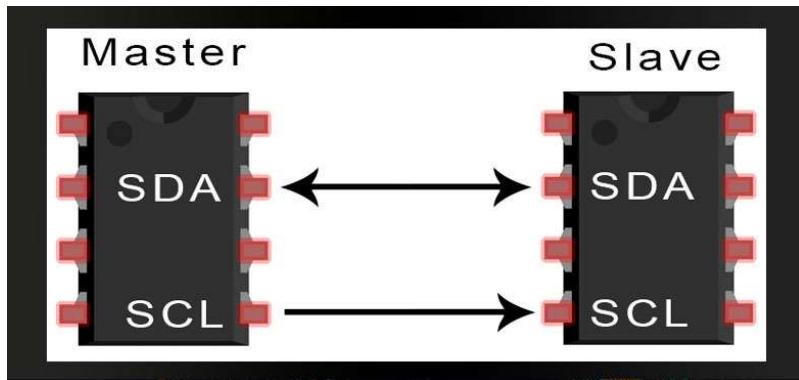
- SDA (Serial Data) برای ارسال و دریافت داده ها Slave و Master خطی برای.
- SCL (Serial Clock) خطی که سیگنال ساعت را حمل می کند.

مانند SPI ، ارتباط I2C نیز همگام است ، بنابراین خروجی بیت ها با نمونه برداری از بیت ها توسط یک سیگنال ساعت مشترک بین Master و Slave هماهنگ می شوند. سیگنال ساعت همیشه توسط Master کنترل می شود. با استفاده از I2C ، داده ها در پیام ها منتقل می شوند. پیام ها به فریم های داده تقسیم می شوند. هر پیام دارای یک فریم آدرس است که شامل آدرس باینری Slave و یک یا چند فریم داده است که حاوی داده های منتقل شده است. این پیام همچنین شامل شرایط شروع و توقف ، بیت های خواندن / نوشتن و بیت های ACK / NACK بین هر قاب داده است:



- **شرایط شروع:** خط SDA از سطح ولتاژ بالا به سطح ولتاژ پایین تغییر می کند قبل از اینکه خط SCL از بالا به پایین تغییر کند.
- **شرایط توقف:** خط SDA پس از اینکه خط SCL از کم به زیاد تغییر می کند از سطح ولتاژ پایین به سطح ولتاژ بالا تبدیل می شود.
- **فریم آدرس:** توالی 7 یا 10 بیتی منحصر به فرد برای هر Slave که وقتی که Master می خواهد با آن صحبت کند ، Slave را مشخص می کند.

- بیت خواندن / نوشتن: یک بیت واحد که مشخص می کند Master داده را به Slave ارسال می کند (سطح ولتاژ پایین) یا از آن داده می خواهد (سطح ولتاژ بالا).
- بیت ACK / NACK: هر فریم در پیام با یک بیت تأیید / عدم تأیید دنبال می شود. اگر یک فریم آدرس یا یک فریم داده با موفقیت دریافت شود، یک بیت ACK از دستگاه گیرنده به فرستنده بازگردانده می شود. در صورت عدم تأیید، بیت NACK فرستاده می شود.



کتابخانه استفاده شده:

```
#include "I2Cdev.h"
```

```

1 // I2Cdev library collection - Main I2C device class
2 // Abstracts bit and byte I2C R/W functions into a convenient class
3 // 2013-06-05 by Jeff Rowberg <jeff@rowberg.net>
4 //
5 // Changelog:
6 //   2021-09-28 - allow custom Wire object as transaction function argument
7 //   2020-01-20 - hardija : complete support for Teensy 3.x
8 //   2015-10-30 - simondlevy : support i2c_t3 for Teensy3.1
9 //   2013-05-06 - add Francesco Ferrara's Fastwire v0.24 implementation with small modifications
10 //   2013-05-05 - fix issue with writing bit values to words (Sasquatch/Farzanegan)
11 //   2012-06-09 - fix major issue with reading > 32 bytes at a time with Arduino Wire
12 //     - add compiler warnings when using outdated or IDE or limited I2Cdev implementation
13 //   2011-11-01 - fix write*Bits mask calculation (thanks sasquatch @ Arduino forums)
14 //   2011-10-03 - added automatic Arduino version detection for ease of use
15 //   2011-10-02 - added Gene Knight's NBWire TwoWire class implementation with small modifications
16 //   2011-08-31 - added support for Arduino 1.0 Wire library (methods are different from 0.x)
17 //   2011-08-03 - added optional timeout parameter to read* methods to easily change from default
18 //   2011-08-02 - added support for 16-bit registers
19 //     - fixed incorrect Doxygen comments on some methods
20 //     - added timeout value for read operations (thanks mem @ Arduino forums)
21 //   2011-07-30 - changed read/write function structures to return success or byte counts
22 //     - made all methods static for multi-device memory savings
23 //   2011-07-28 - initial release
24
25 /* =====
26 I2Cdev device library code is placed under the MIT license
27 Copyright (c) 2013 Jeff Rowberg

```

کتابخانه ذکر شده تحت لاینسس و انتشار یافته از دانشگاه ام آی تی است و در طی سال ایجاد و انتشار از سال 2011 تا 2021 مداوم در گیت هاب اپدیت هایی را دریافت کرده است.

کتابخانه سنسور MPU6050

```
#include "MPU6050_6Axis_MotionApps20.h"
```

سنسور ژیروسکوپ با توجه به داشتن انواع مختلف نظیر سه محوره یعنی بدون بخش شتاب سنج و سرعت سنج دارای کتابخانه های متفاوت نیز هست در این مورد با توجه به پارت نامبر سنسور کتابخانه بالا استفاده شده است.

MPU6050.h MPU6050_6Axis_MotionApps20.cpp MPU6050_6Axis_MotionApps20.h MPU6050_6Axes_MotionApps612.cpp MPU6050_6Axis_MotionApps612.h MPU6050_9Axis_MotionApps41.cpp MPU6050_9Axis_MotionApps41.h

ولیکن در فایل های پروژه همراه با مقاله کتابخانه های دیگر نیز اضافه شده است که کاربر بتواند با توجه به مدل خریداری شده کتابخانه را فراخوانی کرده و بدون مشکل ادامه کار را دنبال کند.

```
A// I2Cdev library collection - MPU6050 I2C device class, 6-axis MotionApps 2.0 implementation
// Based on InvenSense MPU-6050 register map document rev. 2.0, 5/19/2011 (RM-MPU-6000A-00)
// 5/20/2013 by Jeff Rowberg <jeff@rowberg.net>
// Updates should (hopefully) always be available at https://github.com/jrowberg/i2cdevlib
//
// Changelog:
// 2021/09/27 - split implementations out of header files, finally
// 2019/07/08 - merged all DMP Firmware configuration items into the dmpMemory array
//               - Simplified dmpInitialize() to accomidate the dmpMemory array alterations

/* =====
I2Cdev device library code is placed under the MIT license
Copyright (c) 2021 Jeff Rowberg

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.

=====
*/
// MotionApps 2.0 DMP implementation, built using the MPU-6050EVB evaluation board
```

PID کتابخانه

```
#include <PID_v1.h>
```

```
*****
* Arduino PID Library - Version 1.2.1
* by Brett Beauregard <br3ttb@gmail.com> brettbeauregard.com
*
* This Library is licensed under the MIT License
*****
```

- For an ultra-detailed explanation of why the code is the way it is, please visit:
<http://brettbeauregard.com/blog/2011/04/improving-the-beginners-pid-introduction/>

- For function documentation see: <http://playground.arduino.cc/Code/PIDLibrary>

کتابخانه PID نیز تحت لاینسس دانشگاه ام آی اتی است و اپدیت مستمری را دریافت میکند که به رفع مشکل ها میپردازد و ارتقا هایی نیز به شکل خروجی ها اضاف میکند لینک توضیحات بیشتر در ادامه ذکر خواهد شد.

کد های پروژه

```
#include "I2Cdev.h"
#include <PID_v1.h> //From https://github.com/br3ttb/Arduino-PID-Library/blob/master/PID_v1.h
#include "MPU6050_6Axis_MotionApps20.h" //https://github.com/jrowberg/i2cdevlib/tree/master/Arduino/MPU6050

MPU6050 mpu;

در این بخش از کد از چند متغیر برای تست کردن اماده بودن دریافت مقادیر از سنسور سپس برای چک کردن وجود داشتن وقفه در خط سوم برای بررسی خطأ و ارسال آن به خروجی و در خط چهارم و پنجم ایجاد مقادیر فضای مورد نیاز برای گرفتن مقادیر از سنسور ایجاد شده است. در خط اخیر نیز یک ارایه 64 خانه ای برای ریختن مقادیر خوانده شده در نظر گرفته شده است.

// MPU control/status vars
bool dmpReady = true; // set true if DMP init was successful
uint8_t mpuIntStatus; // holds actual interrupt status byte from MPU
uint8_t devStatus; // return status after each device operation (0 = success, !0 = error) ←
uint16_t packetSize; // expected DMP packet size (default is 42 bytes)
uint16_t fifoCount; // count of all bytes currently in FIFO
uint8_t fifoBuffer[64]; // FIFO storage buffer

در خطوط پایینی مقادیر طول، عرض، ارتفاع که همان مقادیر تعریف شده در بالا قسمت ژیروسکوپ است میباشد را در متغیر q و مقادیر مربوط به جانبه را در متغیر مشخص برای قرار گیری مشخص کرده است. در قسمت اخیر نیز یک متغیر از جنس اعشاری برای سه متغیر سنسور تعریف کرده است.

// orientation/motion vars
Quaternion q; // [w, x, y, z] quaternion container
VectorFloat gravity; // [x, y, z] gravity vector ←
float ypr[3]; // [yaw, pitch, roll] yaw/pitch/roll container and gravity vector ←

در بخش زیر که یکی از مهم ترین بخش های ربات است و نیاز است درست تنظیم شود ابتدا مقدار دقیق زاویه ربات را از ما میخواهد تا در ست پوینت لحظه شود طریقه تنظیم این مقدار به این شکل است که ابتدا باید ربات بر روی یک سطح صاف قرار گیرد (میتوان از تراز نما برای اینکار استفاده کرد) سپس تعادل ربات در نقطه تعادلی آن پیدا شود و سپس از طریق اتصال ربات به محیط ترمینال آردوینو یا دیگر ترمینال ها مقدار ست پوینت خوانده شود و سپس در کد تغییر داده شود.

در بخش بعدی نیز مقادیر را به طوری که در بخش PID و در قسمت تنظیم آمده است به صورت آزمون و خط تنظیم کرده ایم ۴۵ صفحه ۴۲ مقاله میباشد.

*****Tune these 4 values for your BOT*****
double setpoint= 178.5; //set the value when the bot is perpendicular to ground using serial monitor.
//Read the project documentation on circuitdigest.com to learn how to set these values
double Kp = 20; //Set this first
double Kd = 0.5; //Set this second
double Ki = 140; //Finally set this
*****End of values setting***** ←

مقادیر ورودی و خروجی برای استفاده در بخش خروجی موتور ها و در ادامه برای تولید سیگنال کنترل موتور ها استفاده میشود. در این بخش هم چنین تابع pid نیز که دارای 6 مدل متغیر ورودی برای تنظیم خروجی است آمده است.

double input, output;
PID pid(&input, &output, &setpoint, Kp, Ki, Kd, DIRECT); ←

در این بخش مقادیر اینترپت سنسور بررسی میشود که در این اینجا به صورت صفر شدن پایه مشخص شده است. و در خط بعدی نیز تابع آمده بودن سنسور برای ارسال دیتا بررسی میشود. ←

volatile bool mpuInterrupt = false; // indicates whether MPU interrupt pin has gone high ←
void dmpDataReady() ←
```

```

{
    mpuInterrupt = true;
}

در این بخش که بخش تابع راه اندازی اولیه است مقدار سریال 115200 تنظیم شده است و برای تغییر این مقدار باید مقدار کتابخانه ها هم تغییر بپدا
کنند.

در بخش بعدی از تابعی استفاده شده است که خطای درست بودن مقدار و فعل پودن سنسور را بررسی میکند اگر سنسور به درستی کار کند پیام کاتکشن
موفقیت امیز بود را نشان میدهد و اگر هم نباشد پیام خطای ارسال میکند تا کاربر بتواند به رفع مشکل بپردازد.

در بخش بعدی تابع راه اندازی سنسور فراخوانی شده و بعد از آن به تنظیم افست های سنسور مبپردازد در این پروژه ما نیاز به تغییر این متغیر ها پیدا
نکردیم ولی در لینک ها توضیحاتی در این مورد نیز ارائه داده شده است.

void setup() {
    Serial.begin(115200); ←
    // initialize device
    Serial.println(F("Initializing I2C devices...")); ←
    mpu.initialize();
    // verify connection
    Serial.println(F("Testing device connections..."));
    Serial.println(mpu.testConnection() ? F("MPU6050 connection successful") : F("MPU6050 connection failed"));
    // load and configure the DMP
    devStatus = mpu.dmpInitialize();
    // supply your own gyro offsets here, scaled for min sensitivity
    mpu.setXGyroOffset(220);
    mpu.setYGyroOffset(76);
    mpu.setZGyroOffset(-85);
    mpu.setZAccelOffset(1688);

    // make sure it worked (returns 0 if so)
    در تابع زیر اگر مقدار متغیر devstatus برابر صفر باشد یعنی سنسور به درستی پیام و داده هارو ارسال میکند و ارتباط موفقیت امیز
    بوده است. ولی اگر شرط برقرار نباشد پیام خطای خروجی نمایش داده خواهد شد.
    در بخش پایینی نیز مود کنترلی PID در مود خودکار قرار گرفته است و محدوده خروجی که برای درایو کردن موتور ها است در محدوده
    255 قرار داده شده است تا به صورت 8 بیتی عمل PWM در خروجی اعمال شود.

    if (devStatus == 0)
    {
        // turn on the DMP, now that it's ready
        Serial.println(F("Enabling DMP..."));
        mpu.setDMPEnabled(true);
        // enable Arduino interrupt detection
        Serial.println(F("Enabling interrupt detection (Arduino external interrupt 0)..."));
        attachInterrupt(0, dmpDataReady, RISING);
        mpuIntStatus = mpu.getIntStatus();
        // set our DMP Ready flag so the main loop() function knows it's okay to use it
        Serial.println(F("DMP ready! Waiting for first interrupt..."));
        dmpReady = true;
        // get expected DMP packet size for later comparison
        packetSize = mpu.dmpGetFIFOPacketSize();
        //setup PID
        pid.SetMode(AUTOMATIC);
        pid.SetSampleTime(10);
        pid.SetOutputLimits(-255, 255);
    }
}

```

```

else
{
    // ERROR!
    // 1 = initial memory load failed
    // 2 = DMP configuration updates failed
    // (if it's going to break, usually the code will be 1)
    Serial.print(F("DMP Initialization failed (code "));
    Serial.print(devStatus);
    Serial.println(F(")"));
}

```

در این قسمت مقادیر پایه های خروجی متصل شده به راه انداز موتور مشخص شده است که به صورت پایه های **6.9.10.11** میباشد که در مقدار اولیه صفر نیز تنظیم شده است این مقادیر به راحتی قابل تغییر اند و میتوان به سادگی تغییر داد تا از پین های دیگر برد نیز در صورت امکان و نیاز استفاده شود.

```

//Initialise the Motor output pins
pinMode (6, OUTPUT);
pinMode (9, OUTPUT);
pinMode (10, OUTPUT);
pinMode (11, OUTPUT);

//By default turn off both the motors
analogWrite(6,LOW);
analogWrite(9,LOW);
analogWrite(10,LOW);
analogWrite(11,LOW);
}

```

شروع حلقه اجرا دستورات:

```
void loop() {
```

در خط زیر مشخص میکند که اگر مقدار تابع اماده بودن سنسور و کانکشن ها یک بود به تابع برگردد و پیام های خط را در خروجی ترمینال ظاهر کند.

```

// if programming failed, don't try to do anything
if (!dmpReady) return;

driven by a condition that was met when the program started. This is done to prevent the robot from moving if the
// wait for MPU interrupt or extra packet(s) available
while (!mpuInterrupt && fifoCount < packetSize)

{
    در خطوط پایین مقادیر کنترلی ربات محاسبه میشود در خط اول تابعی از کتابخانه pid به عنوان compute
    // calculate the error between the current IMU orientation and the desired orientation
    pid.compute();
    // calculate the PID control output
    float output = pid.getOutput();
    // limit the output to a valid range
    if (output < -1.0) output = -1.0;
    if (output > 1.0) output = 1.0;
    // send the output to the motors
    analogWrite(6, map(output, -1.0, 1.0, 0, 255));
    analogWrite(9, map(-output, -1.0, 1.0, 0, 255));
}

```

در قسمتی از کد یک شرط گذاشته شده است که مقدار ورودی را بررسی میکند در واقع اینجا **INPUT** همان زاویه ما است که در این رنج زاویه ای 150 درجه و 200 درجه تنظیم شده است که هرگاه ربات در حال افتادن است موتور ها در این زاویه خاموش شوند و از اسیب خوردن به بدنه دستگاه و روش ماندن موتور ها جلوگیری کند.

```

//no mpu data - performing PID calculations and output to motors
pid.Compute();

//Print the value of Input and Output on serial monitor to check how it is working.
Serial.print(input); Serial.print(" =>"); Serial.println(output);

if (input>150&& input<200){//If the Bot is falling
    if (output>0) //Falling towards front
}

```

```

    Forward(); //Rotate the wheels forward
    else if (output<0) //Falling towards back
        Reverse(); //Rotate the wheels backward
    }
    else //If Bot not falling
        Stop(); //Hold the wheels still
    }
}

در کد های پایین نیز بررسی میشود که آیا مقدار اینترپت و وقفه از مقادیر تعیین شده بیشتر است یا نه که overflow در کد اتفاق نیفت.
// reset interrupt flag and get INT_STATUS byte
mpuInterrupt = false;
mpuIntStatus = mpu.getIntStatus();

// get current FIFO count
fifoCount = mpu.getFIFOCount();

// check for overflow (this should never happen unless our code is too inefficient)
if ((mpuIntStatus & 0x10) || fifoCount == 1024)
{
    // reset so we can continue cleanly
    mpu.resetFIFO();

    Serial.println(F("FIFO overflow!"));
    // otherwise, check for DMP data ready interrupt (this should happen frequently)
}
else if (mpuIntStatus & 0x02)
{
    // wait for correct available data length, should be a VERY short wait
    while (fifoCount < packetSize) fifoCount = mpu.getFIFOCount();
}

```

در خط های پایینی مقدار های **fifo** و **packet** را که در کد های بالا دیدیم بررسی میکنند این دو فلگ یا برمج ها به ما اجازه میدهد در کد های ارسالی از طریق ارتباط سریال بررسی کنیم که آیا مقدار پایان یافته یا شروع شده که امکان میدهد بدون حتی وقفه و سریع تر مقدار را بخوانیم. و سپس در کد خط های بعدی مقادیر ایجاد شده از سنسور توسط کتابخانه و توابع مربوطه خوانده و نوشته میشوند.

```

    // read a packet from FIFO
    mpu.getFIFOBytes(fifoBuffer, packetSize);
    // track FIFO count here in case there is > 1 packet available
    // (this lets us immediately read more without waiting for an interrupt)
    fifoCount -= packetSize;
    mpu.dmpGetQuaternion(&q, fifoBuffer); //get value for q
    mpu.dmpGetGravity(&gravity, &q); //get value for gravity
    mpu.dmpGetYawPitchRoll(ypr, &q, &gravity); //get value for ypr
    input = ypr[1] * 180/M_PI + 180;
}
}

```

در سه تابع پایین نیز که از دستور جلو و عقب و توقف استفاده میکنند مقادیر پورت ها برای هر یک از اعمال نامبرده شده صورت میگیرد.

```

void Forward() //Code to rotate the wheel forward
{
    digitalWrite(6,output);
    digitalWrite(9,0);
    digitalWrite(10,output);
    digitalWrite(11,0);
    Serial.print("F"); //Debugging information
}
void Reverse() //Code to rotate the wheel Backward
{
    digitalWrite(6,0);
    digitalWrite(9,output*-1);
    digitalWrite(10,0);
    digitalWrite(11,output*-1);
    Serial.print("R");
}
void Stop() //Code to stop both the wheels
{
    digitalWrite(6,0);
    digitalWrite(9,0);
    digitalWrite(10,0);
    digitalWrite(11,0);
    Serial.print("S");
}

```

مشاهده خروجی ترمینال در وضعیت های مختلف



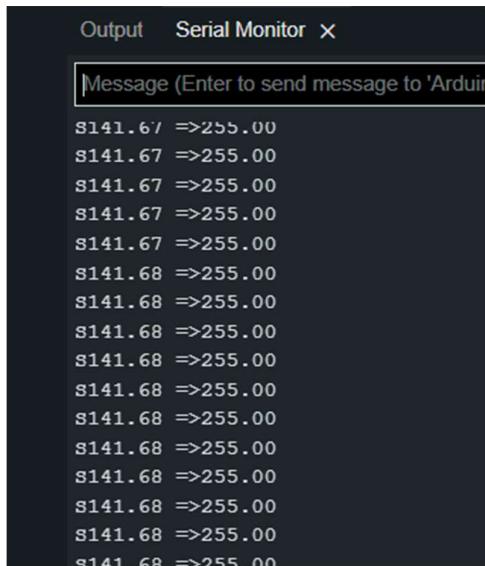
```
F176.67 =>255.00
```

در شکل روبه رو مقدار اولیه

که عدد 176.67 را نشان میدهد مقدار

درجه قرارگیری ربات است و مقدار دوم نیز که از 255
نا صفر متغیر است مقدار خروجی ارسالی به مونو را
نمایش میدهد.

حال با توجه به قرار گیری F یعنی جهت حرکت موتور
به سمت جلو خواهد بود تا ربات در حالت تعادل قرار
گیرد.



```
S141.67 =>255.00
S141.67 =>255.00
S141.67 =>255.00
S141.67 =>255.00
S141.67 =>255.00
S141.68 =>255.00
```

در شکل روبه رو مقدار ترمینال خروجی عدد کمتر از 150

را نمایش میدهد که حرف S را قبل خود دارد یعنی
مотор در حالت توقف قرار دارد. از محدوده input>150&& input<200
150 درجه تا 200 درجه است خارج شده و موتور ها
متوقف شده اند.



```
R195.62 =>-255.00
R195.62 =>-255.00
R195.62 =>-255.00
R195.62 =>-255.00
R195.62 =>-255.00
R195.70 =>-255.00
R195.78 =>-255.00
R195.78 =>-255.00
R195.78 =>-255.00
R195.78 =>-255.00
R195.78 =>-255.00
```

در شکل روبه رو نیز مقدار ترمینال خروجی عدد

195 را نمایش میدهد که موتور به فرمان R یعنی
REVERSE به عقب میرود تا تعادل خود را حفظ کند.

لینک گیت هاب برای دسترسی به فایل ها و عکس های پروژه :

https://github.com/aliasghar1379/Self_Balancing_Robot

منابع و لینک های مفید:

<https://academy.partineh.com>

<https://projecthub.arduino.cc>

<https://automaticaddison.com/how-to-build-a-self-balancing-robot-from-scratch/>

<https://blog.sanattech.com/3861/gyroscope-sensor/>

<https://thecafrobot.com/learn/interfacing-gy-521-mpu6050-3-axis-accelerometer-gyroscope-with-arduino/>

<https://blog.faradars.org>

<https://digispark.ir/>

<https://wle.ir/17602/self-balancing-robot.html>

<https://rastapad.com/articles/what-is-gearbox-motor/>

<https://melec.ir>

<https://irenx.ir/learn/pid-controller/>

<http://brettbeauregard.com/>

https://github.com/br3ttb/Arduino-PID-Library/blob/master/PID_v1.cpp

<https://playground.arduino.cc/Code/PIDLibrary/>

The End