

به نام خدا



دانشکده برق و کامپیوتر  
دانشگاه تهران

گزارش پژوهی حفظ خودکار تعادل ربات  
سیستم‌های نهفته‌ی بی‌درنگ - بهار ۱۳۹۷  
دکتر مهدی کارگهی

میلاد حکیمی

miladhakimi۳۳@gmail.com

صادق حائری

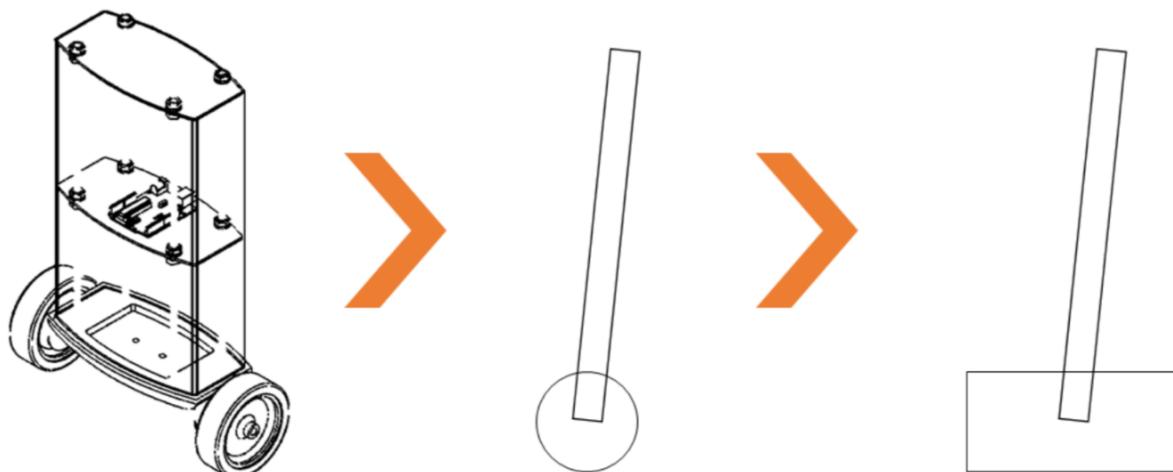
hayerisadegh@gmail.com

سحر رجبی

sahar.rajabi۷۶@gmail.com

## شرح کلی پروژه:

ربات segway با هدف حفظ تعادل، حالتی از مساله‌ی آونگ‌های معکوس است که به دلیل ناپایداری و غیرخطی بودن سیستم، موضوع مهم و قابل توجهی در مسائل مهندسی کنترل است. در این ربات که بر روی دو چرخ قرار دارد، باید به صورت متناوب با اندازه‌گیری زاویه‌ی انحراف ربات از محور تعادل، و ارسال آن به یک سیستم کنترلی و سپس بدست آوردن سرعت و جهت مناسب حرکت موتورها، ربات را در حالت تعادل نگه داشت و یا در صورت به هم خوردن این وضعیت، آن را به حال تعادل بازگرداند. (دقت شود که هدف این پروژه تنها تعادل ربات است و ثابت نگهداشتن مکان و یا نچرخیدن آن مد نظر نیست)

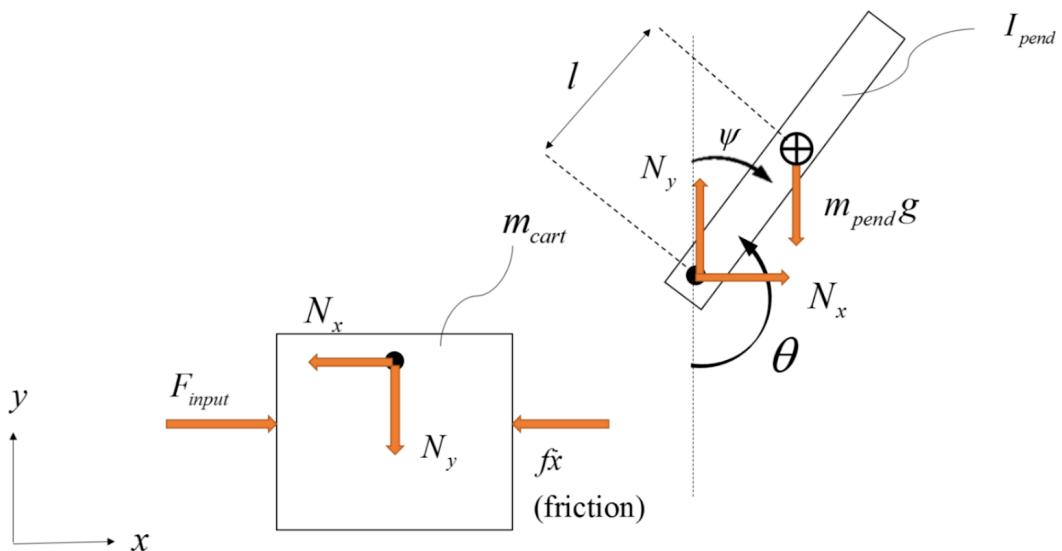


شکل (۱): مدل ساده‌شده Segway.

## مدل سیستم:

### مدل ریاضی سیستم:

همانطور که گفته شد، برای برگرداندن ربات به حالت تعادل، باید به نیروهایی که به آن وارد می‌شوند و باعث ایجاد سرعت زاویه‌ای حول محور عمود بر سطح افق می‌گردند؛ غلبه کنیم. برای این کار، باید درکی از نیروهای وارد بر این سیستم داشته باشیم. در شکل زیر، دیاگرام نیروهای وارد بر ربات کشیده شده است:



شکل (۲): دیاگرام نیروهای وارد بر ربات.

همانند شکل‌های بالا، مدل کلی را می‌توان با تقریب خوبی به مدلی که در آن آونگ معکوس روی یک ارابه قرار دارد، تبدیل کرد.

### قسمت ا (معادله‌ی اربه):

در این قسمت تنها فرمولی که برای مدل کردن حرکت اربه نیاز داریم، حرکت آن در راستای محور x است و به شکل زیر است:

$$F_{input} = m_{cart}\ddot{x} + f\dot{x} + N_x \quad (\text{A.1})$$

قسمت ۲(معادلات پاندول):

برآیند نیروهایی که در راستای محور  $x$  به پاندول وارد میشود:

$$N_x = m_{pend}\ddot{x} + m_{pend}l\ddot{\theta} \cos \theta - m_{pend}l\dot{\theta}^2 \sin \theta \quad (\text{A.2})$$

برآیند همه نیروهایی وارد شده به پاندول:

$$N_y \sin \theta + N_x \cos \theta - m_{pend}g \sin \theta = m_{pend}l\ddot{\theta} + m_{pend}\ddot{x} \cos \theta \quad (\text{A.3})$$

با سادهسازی معادله‌ی بالا، معادله‌ی زیر به دست می‌آید:

$$-N_y l \sin \theta - N_x l \cos \theta = I_{pend}\ddot{\theta} \quad (\text{A.4})$$

قسمت ۳(ترکیب معادلات بالا):

با ترکیب A.۱ و A.۲ و جایگذاری  $N_x$  در A.۱ داریم:

$$F_{input} = (m_{cart} + m_{pend})\ddot{x} + f\dot{x} + m_{pend}l\ddot{\theta} \cos \theta - m_{pend}l\dot{\theta}^2 \sin \theta \quad (\text{A.5})$$

با ترکیب A.۲ و A.۳ و جایگذاری  $N_x$  در معادله‌ی A.۳ داریم:

$$(I_{pend} + m_{pend}l^2)\ddot{\theta} + m_{pend}gl \sin \theta = -m_{pend}l\ddot{x} \cos \theta \quad (\text{A.6})$$

قسمت ۴(محاسبه‌یتابع تبدیل):

برای محاسبه‌یتابع تبدیل باید تبدیل لابلاس معادله‌ی A.۵ را خطی کنیم:

برای این کارتواج غیرخطی A.۵ و A.۶ را کمی تغییر می‌دهیم.

$$\cos \theta = \cos(\pi + \psi) \approx -1 \quad (\text{A.7})$$

$$\sin \theta = \sin(\pi + \psi) \approx -\psi \quad (\text{A.8})$$

$$\dot{\theta}^2 = \dot{\psi}^2 \approx 0 \quad (\text{A.9})$$

در ابتدا  $\theta$  را تبدیل به  $\varphi - \pi$  می‌کنیم. بر اساس شکل مدل سیستم،  $\varphi$  نزدیک به صفر است پس می‌توان با تقریب نسبتاً خوبی فرمول‌های بالا را استنباط کرد.

با جایگذاری A.۸، A.۹ و A.۱۰ در معادلات A.۵ و A.۶ به معادلات A.۱۱ و A.۱۲ می‌رسیم:

$$(I_{pend} + m_{pend}l^2)\ddot{\psi} - m_{pend}gl\psi = m_{pend}l\ddot{x} \quad (\text{A.10})$$

$$u_{input} = (m_{cart} + m_{pend})\ddot{x} + f\dot{x} - m_{pend}l\ddot{\psi} \quad (\text{A.11})$$

که با تبدیل لایپلاس گرفتن از دوطرف معادله، به معادله‌های A1۲ و A1۳ میرسیم؛ (توجه شود که تبدیل به  $U_{input}$  شده است).

$$(I_{pend} + m_{pend}l^2)\Psi(s)s^2 - m_{pend}gl\Psi(s) = m_{pend}lX(s)s^2 \quad (A.12)$$

$$U_{input}(s) = (m_{cart} + m_{pend})X(s)s^2 + fX(s)s - m_{pend}l\Psi(s)s^2 \quad (A.13)$$

در تابع تبدیل باید نسبت ورودی به خروجی را به دست آوریم. با حل  $X(s)$  از معادله‌ی A.12 به معادله‌ی زیر میرسیم:

$$X(s) = \left[ \frac{I_{pend} + m_{pend}l^2}{m_{pend}l} - \frac{g}{s^2} \right] \Psi(s) \quad (A.14)$$

با جایگذاری  $X(s)$  در A.13 به معادله‌ی زیر میرسیم:

$$U_{input}(s) = (m_{cart} + m_{pend}) \left[ \frac{I_{pend} + m_{pend}l^2}{m_{pend}l} - \frac{g}{s^2} \right] \Psi(s)s^2 + f \left[ \frac{I_{pend} + m_{pend}l^2}{m_{pend}l} - \frac{g}{s^2} \right] \Psi(s)s - m_{pend}l\Psi(s) \quad (A.15)$$

حال نسبت  $\Psi(s)$  به  $U_{input}(s)$  را به دست می‌آوریم:

$$\Psi(s) = \underbrace{\frac{\frac{m_{pend}}{q}s}{s^3 + \frac{f(I_{pend} + m_{pend}l^2)}{q}s^2 - \frac{(m_{cart} + m_{pend})m_{pend}gl}{q}s - \frac{fm_{pend}gl}{q}}} U_{input}(s) \quad (A.16)$$

where

$$q = \left[ (m_{cart} + m_{pend})(I_{pend} + m_{pend}l^2) - (m_{pend}l)^2 \right] \quad (A.17)$$

سپس تابع  $G_x(s)$  که نسبت  $U_{input}(s)$  به  $X(s)$  را نشان میدهد به دست می‌آید:

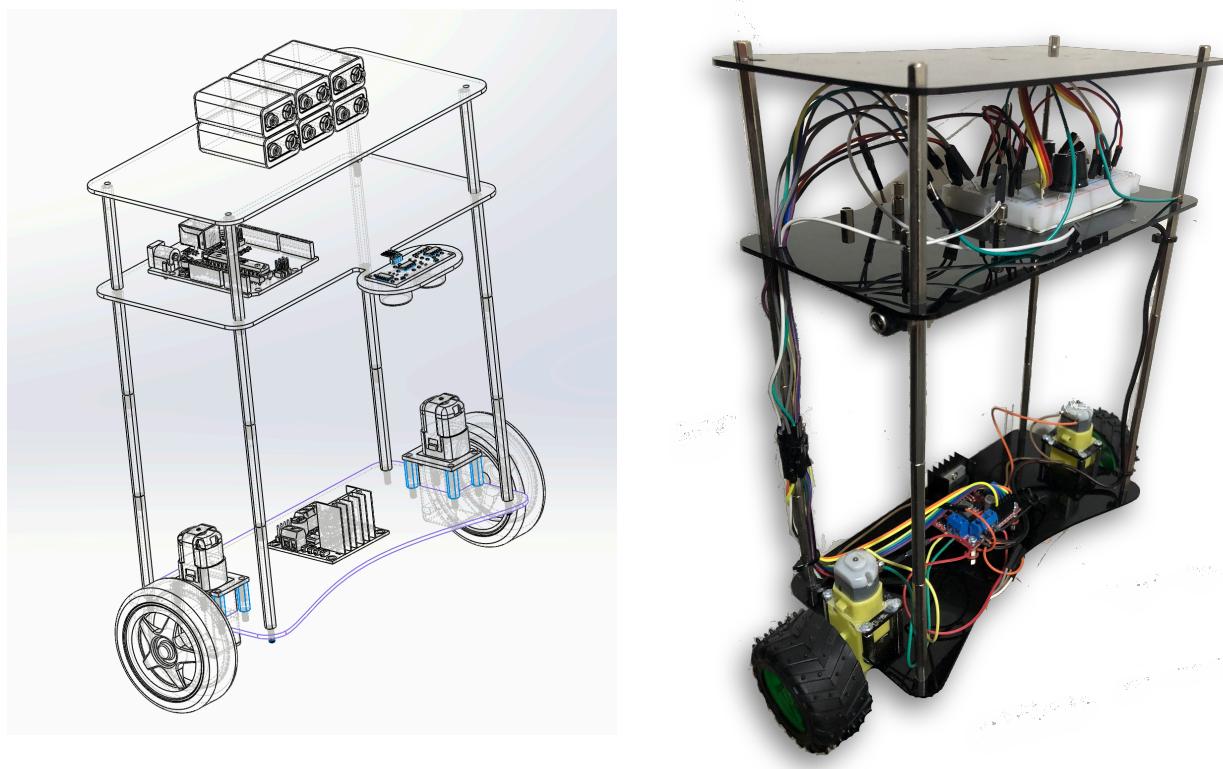
$$X(s) = \underbrace{\frac{\frac{(I_{pend} + m_{pend}l^2)s^2 - gm_{pend}l}{q}}{s^4 + \frac{f(I_{pend} + m_{pend}l^2)}{q}s^3 - \frac{(m_{cart} + m_{pend})m_{pend}gl}{q}s^2 - \frac{fm_{pend}gl}{q}s}} U_{input}(s) \quad (A.18)$$

سخت افزار:

**شاسی:**

برای طراحی بدن، با توجه به اجزای مورد نیاز برای پروژه، یک شاسی برای قرار گرفتن موتورها و درایور نیاز بود، و یک شاسی برای حمل آردوینو و سایر ماظولهای استفاده شده. همچین، ماظول GY-25 را در محور چرخها نصب کردیم که تا لرزش ناشی از حرکت موتورها اثر کمتری بر سنسور شتاب سنج که حساسیت بالایی به نویزها دارد داشته باشد.

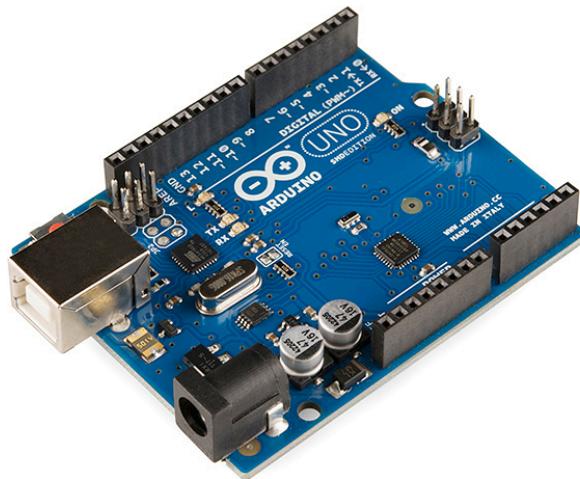
برای طراحی بدنی ربات از نرم افزار solidworks استفاده کردیم و با استفاده از ماظولهای موتورهایی که از قبل برای استفاده در نرم افزار طراحی شده بودند، جای مناسب برای هر کدام از اجزا در نظر گرفته شد. نهایتاً از روی طرح های کشیده شده، شاسی ها با جنس پلکسی ۲ میلی متر (برای سبکی سازه) توسط لیزر برش داده شدند تا با دقت حداکثری در طراحی - مخصوصاً مکان قرار گرفتن موتورها برای لرزش های احتمالی که با محکم نبودن مکان نصب می توانستند ایجاد کنند - شرایط را برای عملکرد مناسب سیستم فراهم کنیم. تصاویر زیر، طرح شاسی های کشیده شده در solidworks است و در نهایت ربات قابل مشاهده است:



شکل (۳): تصویر نهایی پیاده سازی شده توسط نرم افزار سالیدورک.

شکل (۴): تصویر نهایی پیاده سازی شده و کامل شده ربات.

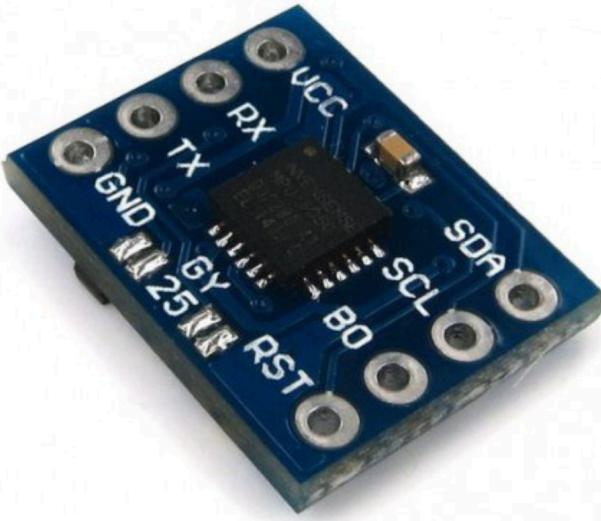
در این پروژه، از برد Arduino Uno برای برنامه ریزی و اجرای الگوریتم‌های مورد نیاز استفاده شده است. این برد، با توجه به گستردگی منابع و کتابخانه‌ها جهت پیش بردن اهداف، در دسترس بودن، قیمت مناسب، و کافی بودن امکانات برای پیاده‌سازی این ربات، با توجه به تعریفی که در بالا ارائه شد، انتخاب شد.



شکل (۵): تصویر برد آردینو UNO

با وجود اینکه بر روی این برد، تنها یک پورت سریال موجود است، و برای ارسال داده‌ها از مژول MPU6050 - که در ادامه توضیح داده خواهد شد - و ارسال دستورات کنترلی به ربات، حداقل به دو پورت احتیاج داریم، اما با امکان تعریف software serial ها، برای ارتباطاتی که از حساسیت کمتری برخوردارند - مانند ارسال دستورات حرکتی و تعیین جهت و یا تنظیم کردن پارامتر الگوریتم‌ها - می‌توان بدون در نظر گرفتن این محدودیت از همین برد برای اجرایی کردن پروژه بهره برد. برد مورد استفاده ما مبتنی بر ۳۲۸ MHz Atmega است و با فرکانس ۱۶MHz کار می‌کند.

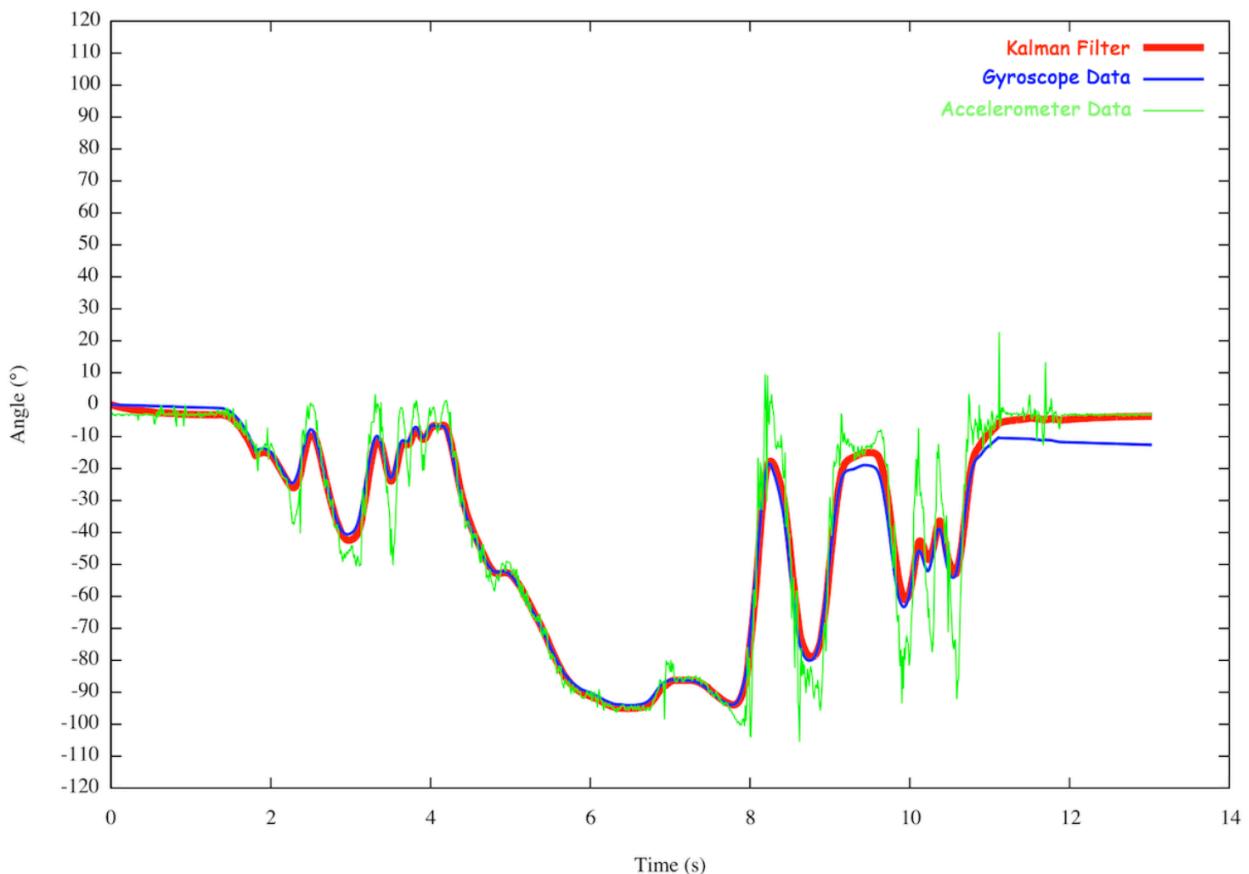
برای اندازه گرفتن میزان انحراف ربات از محور تعادل، از سنسور MPU6050 استفاده می‌شود. این مژول هم سنسور ژیروسکوپ و هم سنسور شتاب‌سنج را در خود به صورت یکجا دارد و به ما با دقت مناسبی با ۶درجه آزادی می‌دهد. یکی از مژول‌های مناسب که روی خود این سنسور را داراست مژول GY-25 می‌باشد، برتری این مژول نسبت به مژول‌های مشابه مثل GY-521 این است که علاوه بر داشتن ارتباط I2C، به ما امکان ارتباط به صورت Serial را هم می‌دهد، همچنین مژول GY-521 داده‌های ژیروسکوپ و شتاب‌سنج را به صورت خام در دسترس قرار می‌دهد و برای استفاده از آنها نیاز است که فیلتر کالمن که در ادامه توضیح خواهد داده شد به صورت نرم‌افزاری روی برد آردینو پیاده‌سازی شود که این کار باعث افزایش حجم پردازشی برد می‌شود ولی مژول GY-25 بر روی خود، به صورت سخت‌افزاری فیلتر کالمن را روی داده‌های ژیروسکوپ و شتاب‌سنج اعمال می‌کند که این کار باعث افزایش موازی‌سازی محاسبات خواهد شد.



شکل (۶): تصویر مژول GY-۲۵

سنسور ژیروسکوپ یا همان زاویه‌سنج، با انتگرال گرفتن از سرعت زاویه‌ای، مقدار چرخش حول محور را بدست می‌آورد. در نتیجه در طولانی مدت، اباشته شدن خطاهای اندازه‌گیری باعث خطای قابل توجه در مقدار گزارش شده خواهد بود. از طرفی سنسور شتاب‌سنج به نویزهای محیطی و نوسانات غیر ملموس هم حساس است، پس داده‌ی آن در طولانی مدت، به کمک مقدارهای میانگین و حذف نویزها، قابل اتکاتر خواهد بود اما در بازه‌های زمانی کوتاه، دقیق لازم برای اندازه‌گیری جابجایی را ندارد. فیلتر کالمون، با ترکیب داده‌ی این دو سنسور، از مزیت هر دوی آن‌ها استفاده می‌کند، تا داده‌ای قابل اطمینان به خروجی ارسال کند.

نحوه‌ی عملکرد این فیلتر، در عکس زیر قابل مشاهده است:



شکل (۷): شکل نمودار تاثیر فیلتر کالمون بر روی داده‌های ژیروسکوپ و شتاب‌سنج

این فیلتر، با فرض اینکه نویزهای محیط متغیرهای رندم، با توزیع Gaussian هستند؛ با نگهداری وضعیتی که بیانگر داده‌های پیشین است، در هر مرحله، با توجه به مشاهدات مرحله‌ی قبلی، و نویزهای مشاهده شده تا آن لحظه، و در نظر داشتن توزیع فرض شده، و به کمک ترکیب داده‌های قبلی و داده‌ای که در همین لحظه مشاهده شده، پیش‌بینی‌ای برای مرحله‌ی بعد انجام می‌دهد. به این ترتیب، در هر دوره، می‌تواند با مقایسه‌ی پیش‌بینی انجام شده و داده‌های دریافتی، نویزهای داده را تا حد قابل قبولی حذف کند.

همچنین می‌توان MPU6050 را برای حرکت‌های سریع و یا آهسته برنامه‌ریزی کرد، به طوری که ژیروسکوپ آن در چهار بازه‌ی ۲۵۰، ۵۰۰، ۱۰۰۰ و ۲۰۰۰ درجه بر ثانیه و سنسور شتاب‌سنج در چهار بازه‌ی ۸g، ۱۶g، ۴g، ۲g و ۰g مترمربع بر ثانیه قابل برنامه‌ریزی است. در مورد مسائله‌ی ما، با توجه به حرکت آهسته‌ی سیستم، تنظیمات ۲۵۰ درجه بر ثانیه، و ۲g مترمربع بر ثانیه مناسب و کافی به نظر می‌رسد.

برای حرکت ربات از دو DC Motor استفاده کرده‌ایم که از گیربکس ۱/۴۰ کمک می‌گرفتند که هم نیروی گشتاور لازم را می‌توانستند اعمال کنند و هم سرعت عکس‌العمل بالایی داشتند.

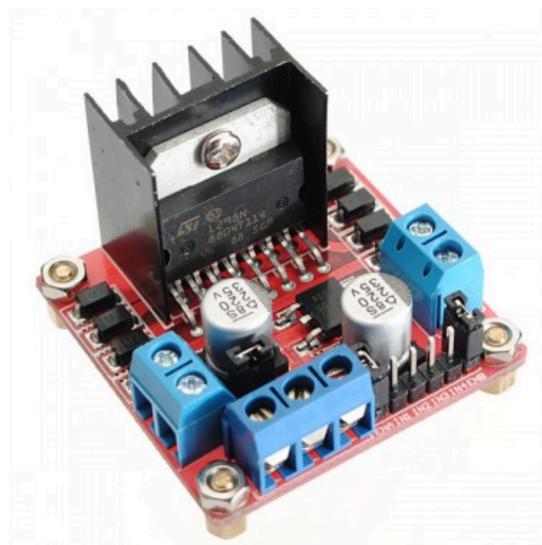


ویژگی‌های این نوع موتور در زیر آورده شده است:

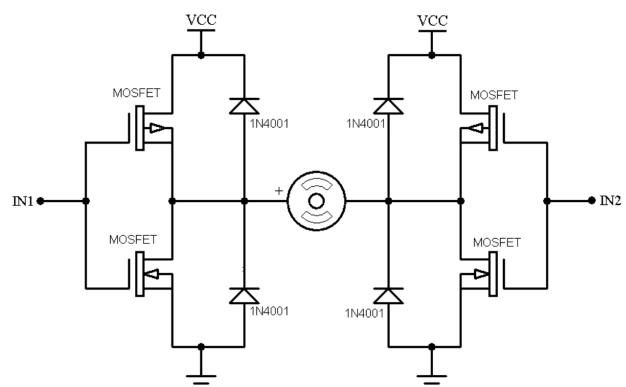
شکل (۸): موتور DC

Rated Voltage	6.0VDC
Rated Load	800 g*cm
No-load Current	70 mA max
No-load Speed	90 ±10 rpm
Loaded Current	190 mA max
Loaded Speed	4500 ±1500 rpm
Starting Torque	20 g*cm
Starting Voltage	2.0
Stall Current	500mA max
Body Size	27.5mm x 20mm x 15mm
Shaft Size	8mm x 2mm diameter
Weight	17.5 grams

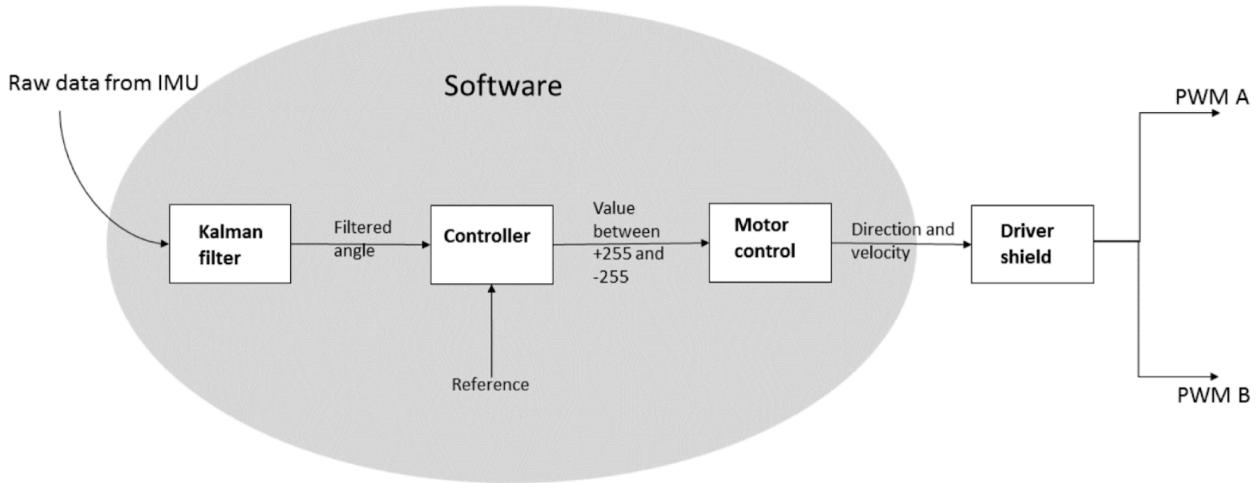
برای ایزوله کردن جریان موتورها و اجزای دیگر از ماتژول درایور L298B استفاده کردیم. این درایور با داشتن ۲ پل H می‌تواند سرعت و جهت چرخش هر دو موتور را به صورت مجزا کنترل کند که با اعمال ولتاژ به پایه‌های IN1 و IN2 امکان تغییر جهت موتور و اعمال PWM به پایه enable آن سرعت موتور را می‌توان تنظیم کرد.



شکل (۱۰): درایور موتور L298b



شکل (۹): پل H پیاده‌سازی شده در درایور موتور



شکل (۱۱): محدوده‌ی نرم‌افزار در پیاده‌سازی پروژه

### محیط توسعه

برای انجام این پروژه از برد آردوینو و محیط نرم‌افزاری Arduino IDE و زبان C++ کمک گرفته شده است.

برای ارتباط از طریق بلوتوث و ارسال داده‌ها به آردوینو از زبان پایتون کمک گرفته شد.

همچنین برای کمک گرفتن از مازول ultrasonic در قسمت امتیازی پروژه (جز اهداف پروژه نبوده است) نیاز بود تا از threadهای جداگانه استفاده شود، چون این سنسور به صورت بلاکینگ عمل می‌کند و می‌تواند تاخیر زیادی روی پردازش به وجود بیاورد، برای حل این مشکل و استفاده از threadها در آردنینو، از سیستم‌عامل FreeRTOS کمک گرفتیم اما در عمل استفاده از ریسمان‌ها در این امبدد سیستم باعث افت کارایی (احتمالاً سربار تعویض ریسمان‌ها) و پایین‌آمدن تعادل ربات شد، به همین دلیل از استفاده از ریسمان به طور کلی منصرف شدیم و روی هدف اصلی پروژه تمرکز کردیم.

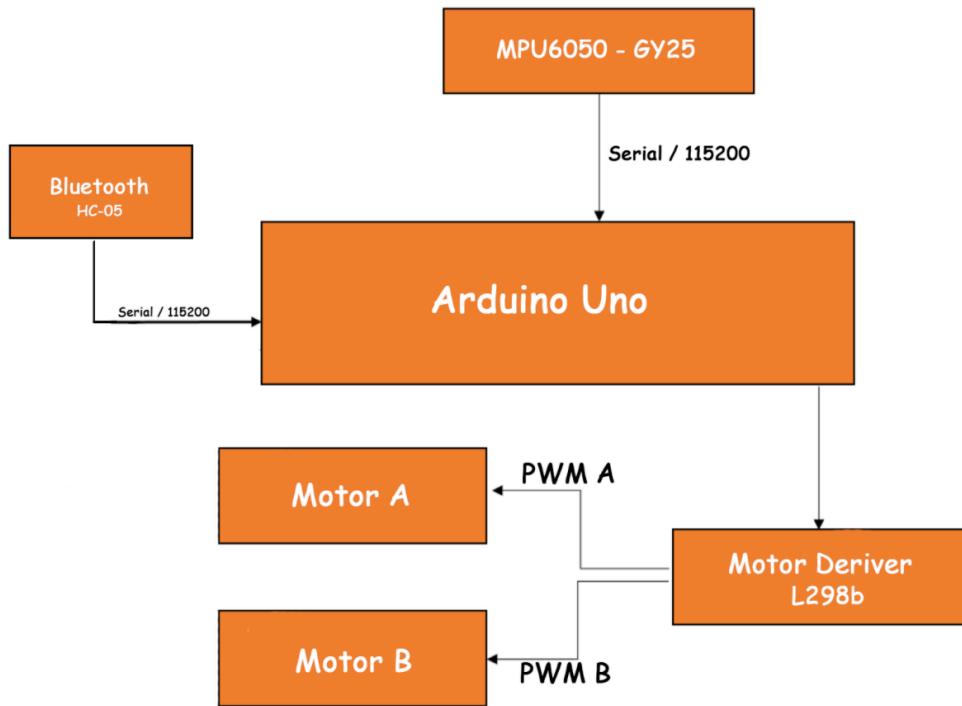
### ورودی‌ها و خروجی‌ها و ارتباط با سخت‌افزار:

همانطور که در بخش سخت‌افزار گفته شد، برای مشاهده میزان انحراف نسبت به حالت عمودی از سنسورهای MPU6050 که روی مازول GY-25 قرارگرفته، استفاده می‌شود. داده‌ها از روی پورت شماره‌ی صفر (RX) خوانده می‌شوند.

برای عملیات کنترل و ارسال دستورات نیز از software serial استفاده شده و داده‌ها از روی پورت شماره‌ی چهار خوانده می‌شوند.

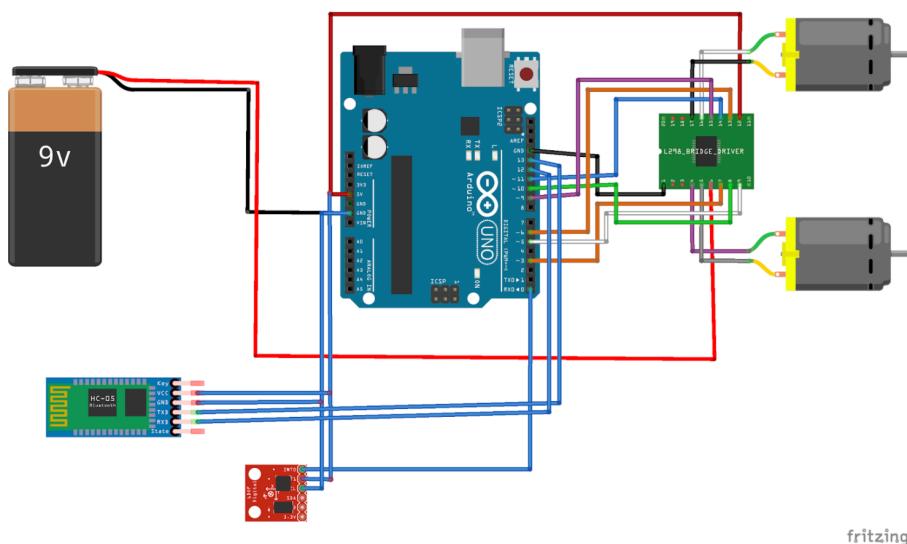
پس از پردازش ورودی‌ها، برای تعیین سرعت موتورها، عددی بین ۰ تا ۲۵۵ روی پین‌های ۶ و ۱۱ میفرستیم.

حرکت هر موتور ۳ حالت دارد؛ ساعت‌گرد، پادساعت‌گرد و بدون حرکت. پس برای تعیین جهت هر کدام از موتورها به دو پین نیاز داریم. این خروجی‌ها بر روی پین‌های ۷ و ۸ برای موتور A و پین‌های ۹ و ۱۰ برای موتور B ارسال می‌شوند.



شکل (۱۲): ارتباط انتزاعی اجزای سختافزاری ربات

## نحوه ارتباط با اجزای سختافزار:



شکل (۱۳): ارتباط دقیق اجزای سختافزاری ربات

## پیاده‌سازی (الگوریتم و نحوه کار کردن اجزای سیستم):

سخت‌افزار:

### ماژول GY-25:

برای اتصال ماژول زاویه سنج GY-25، دو نوع ارتباط I<sub>C</sub> و Serial در دسترس بود که ما برای نویز کمتر و نیاز به یک سیم کمتر از پروتکل Serial آن استفاده کردیم، که با اتصال آن به پین صفر آردوینو (سریال سخت‌افزاری) داده‌های سنسور به آردوینو منتقل می‌شوند.

### ماژول L298b:

برای اتصال این ماژول از خروجی ۵ ولت آردوینو به ورودی ۵ ولت آن متصل کردیم و برای جریان اصلی به ورودی آن از منبع ولتاژ ۱۲ ولت استفاده کردیم. برای هر موتور<sup>۳</sup> سیم مورد نیاز است، IN1 و IN2 برای کنترل کردن موتور و یک سیم که به پایه‌ی Enable متصل می‌شود که باید پالس PWM به آن داده شود. برای این کار از پین‌های ۵، ۶ و ۷ برای موتور اول و پین‌های ۸، ۹ و ۱۰ برای موتور دوم کمک گرفتیم.

### ماژول HC-05:

برای متصل کردن ماژول بلوتوث (این ماژول برای ارسال دستورات و تنظیم کردن ضریب‌های کنترلر مورد استفاده قرار می‌گیرد) تنها کافی بود از خروجی ولتاژ ۵ ولت آردوینو به ورودی ولتاژ آن وصل کنیم و RX و TX را به ترتیب به TX و RX نرم‌افزاری آردوینو متصل کنیم. (برای این کار از پین‌های ۲ و ۳ استفاده شده بود)

برای ارسال دستورات کنترلی به بلوتوث و به آردوینو از یک لپتاپ استفاده کردیم که با استفاده از زبان پایتون و کتابخانه pySerial دستورات برای آردوینو و داده‌های مورد نیاز از آردوینو به آن ارسال می‌شدند.

### موتورها:

ما از موتورهای ارزان‌قیمت برای انجام این پروژه استفاده کردیم که ولتاژ کاری آن‌ها ۶ ولت است (دیتاشیت در مراجع آورده شده است) اما با اعمال ولتاژ ۱۲ ولت به این موتورها گشتاور بالایی برای حرکت ربات ایجاد نمی‌شد. همچنین با توجه به اینکه در اکثر موارد مقدار PWM ورودی آنها مقدار کوچکی است ولی نیاز است که در زمان‌های خاصی گشتاور بالایی توسط موتور ایجاد شود. (موقع وارد شدن ضربه به ربات برای برگرداندن ربات به حالت اولیه) بجای اینکه از موتورهای دیگری استفاده کنیم و هزینه‌ی بالاتری پرداخت کنیم، تصمیم بر این شد که ولتاژ کاری موتورها را بالا ببریم که برای تغذیه بجای ۶ ولت از ۱۲ ولت استفاده کردیم. (همچنین تا ولتاژ ۲۰ ولت را برای آنها امتحان کردیم که موتورها به خوبی با این ولتاژ بالا کار کردند و در کوتاه مدت مشکلی برای آنها ایجاد نشد)

## منبع تغذیه:

در ابتدا برد آردینو و موتورها از یک مدار و یک منبع تغذیه استفاده می‌کردند که با استفاده از یک آداتپر ۱۲ ولت ۳ آمپر جریان مورد نیاز آنها تامین می‌شد و با استفاده از یک رگولاتور ۷۸۰۵ ولتاژ ۱۲ ولت به ولتاژ ۵ ولت مورد نیاز برای برد آردینو و ماژول‌های دیگر تبدیل می‌شد؛ اما در تست‌های انجام شده به دلیل ماهیت سیستم نیاز بود تا جهت حرکت موتورها به یکباره تغییر کنند، در این صورت در زمان تغییر جهت موتورها جریان بسیار زیادی به سیستم اعمال می‌شد و باعث افت جریان و در نتیجه خاموش شدن همه‌ی اجزا می‌شد، با اندازه‌گیری جریان مورد نیاز توسط مولتی‌متر متوجه شدیم که به صورت عادی جریان ۵.۰ تا ۱ آمپری مورد نیاز است ولی در زمان تغییر جهت یکباره تا جریان ۵ آمپر به سیستم اعمال می‌شود که با تعویض منبع تغذیه این مشکل برطرف شد. اما هنوز در زمان تغییر جهت موتورها نویز زیادی روی داده‌های سنسور و آردینو دیده می‌شد که با جدا کردن کامل منبع تغذیه آن‌ها این مشکل نیز برطرف شد.

## نرم‌افزار:

### سیستم کنترلی:

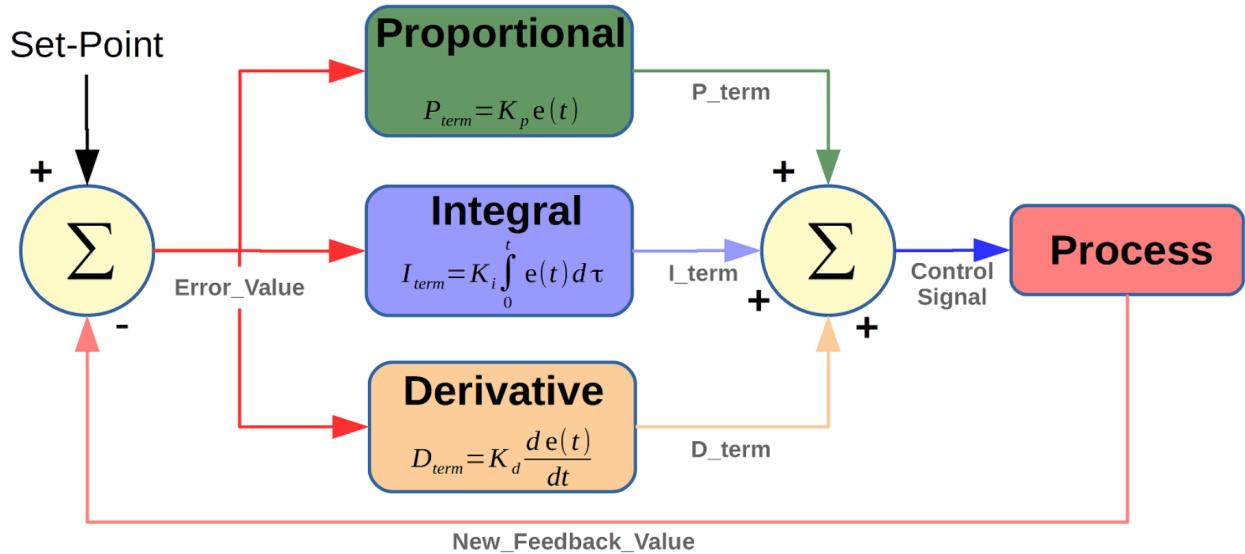
با همان PID Controller Proportional-integral-derivative-controller در پروژه‌های مشابه، از جمله کنترل آونگ معکوس استفاده‌ی بسیاری دارد. این کنترلر سعی در کمینه کردن خطای مقدار ورودی از مقدار ایده‌آل دارد. خروجی این کنترلر مطابق فرمول زیر محاسبه می‌شود:

$$u(t) = k_p e(t) + k_i \int_0^t e(\tau) d\tau + k_d \frac{de(t)}{dt}$$

که در آن set-point مقدار ایده‌آلی است که برای رسیدن به آن از کنترل PID استفاده می‌کنیم و  $e(t)$  همان مقدار خطای ورودی از set-point است.

پارامتر  $k_p$  در واقع ضریبی از مقدار خطای خروجی اضافه می‌کند که برای تنظیم قدرت ربات برای واکنش متناسب با میزان خطای ضروری است. همچنین  $k_d$  از آنجایی که ضریب مشتق تغییرات نسبت به زمان است، به نسبت بزرگی شتاب تغییرات سهمی در واکنش سیستم خواهد داشت. در نهایت  $k_i$  با استفاده از ذخیره‌ی تغییرات قبلی - انتگرال گیری - در صورتی که برای مدت زمانی خطای انباشته شده باشد، با اضافه کردن ضریبی ربات را مجبور به واکنش مناسب خواهد کرد.

شکل زیر نحوه کار این کنترلر را تصویر می‌کند:



شکل (۱۴): دیاگرام کنترلر PID

برای مدل سازی این مساله، از یک PID استفاده شده است که ورودی آن همان زاویه ای انحراف از محور تعادل است که به عنوان خروجی از MPU6050 دریافت می‌کنیم و با تعیین پارامترهای مناسب برای این کنترلر، به عنوان خروجی به ما PWM مناسب برای چرخش موتورها را خواهد داد.

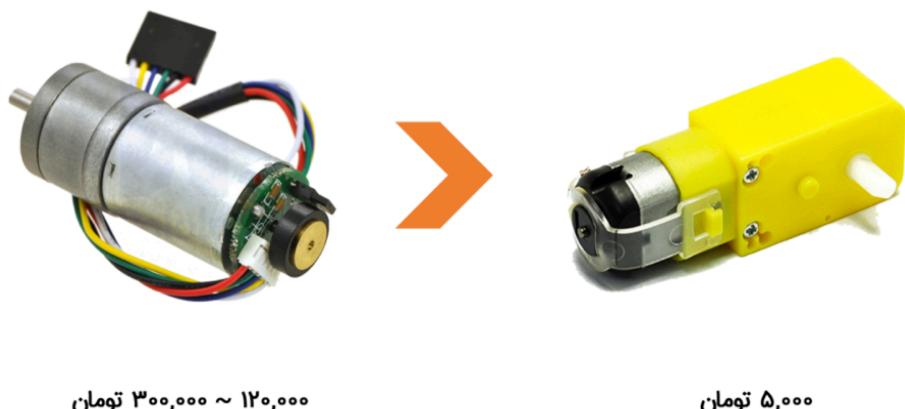
مهمترین قسمت آن پیدا کردن ضرایب  $K_p$ ،  $K_i$  و  $K_d$  است، یکی از ساده‌ترین و بهترین روش‌هایی که برای پیدا کردن ضرایب استفاده می‌شود به این صورت است که، ابتدا تمام ضرایب را برابر صفر قرار می‌دهیم؛ سپس  $K_p$  را به تدریج اضافه می‌کنیم تا ربات به آرامی نوسان کند و تعادل خود را حفظ کند. اگر این پارامتر پایین انتخاب شود، با افزایش دامنه نوسان‌ها، ربات قدرت کافی برای واکنش به زاویه ای انحرافی ایجاد شده و برگشتن به حال تعادل را نخواهد داشت. از طرفی اگر مقدار بالایی را برای آن لحظه کنیم ربات با شدت زیادی شروع به نوسان خواهد کرد که همین کار حفظ تعادل را مشکل می‌کند. پس بهترین مقدار همان مقداری است که ربات را در حالت نوسان آرام قرار دهد.

بعد از انتخاب مقدار مناسب  $K_p$ ، نوبت به تنظیم کردن  $K_d$  می‌رسد. مقدار مناسب برای این پارامتر، مقداری است که نوسانات ریز ربات که در قسمت قبل گفته شد را تا حد مناسبی کاهش دهد تا حدی که ربات تقریباً در حالت ایستا باقی بماند. همچنین مقدار مناسب برای این ضریب، ربات را در مقابل ضربه‌ها هم مقاوم خواهد کرد.

در نهایت اقدام به تنظیم ضریب  $K_i$  می‌کنیم. زمانی که سیستم را روشن می‌کنیم، در ابتدا ربات برای مدت زمانی نوسان می‌کند تا به حالت تعادل برسد. انتخاب مناسب  $K_i$  این بازه‌ی زمانی را کاهش خواهد داد.

## تغییرات نسبت به فاز پروپوژال:

یکی از تغییرات ایجاد شده نسبت به فاز پروپوژال استفاده از موتورهای معمولی و ارزان قیمت بجای استفاده از موتور انکودردار بود که دلیل این تغییر تنها قیمت بسیار بالای این نوع موتورها بود. موتورهای مورد استفاده ما همراه با گیربکس و چرخ پلاستیکی تنها ۵ هزار تومان برای هریک از مجموعه‌های چپ و راست ربات هزینه داشت. در صورتی که قیمت موتورهای شفت انکودردار موجود در بازار قیمت‌های بالای ۱۰۰ هزار تومان برای هر موتور بود.



۱۲۰,۰۰۰ ~ ۱۳۰,۰۰۰ تومان

۵,۰۰۰ تومان

با این تصمیم فیدبک موتورها را از دست دادیم ولی توانستیم از فیدبک کل سیستم (زاویه ربات) بجای آن استفاده کنیم و همچنین از زاویه نسبت به محور ۷‌ها توانستیم تفاوت سرعت دو موتور را با استفاده از یک کنترلر PID جبران کنیم؛ در کل تنها کمی نویز به سیستم اضافه شد ولی در عوض هزینه ربات کاهش بسیار زیادی داشت.

همچنین با توجه به هزینه‌ی بالای باتری، و اینکه با گذشت زمان و کاهش شارژ باتری‌ها نیروی حرکتی موتورها هم کاهش می‌یافتد، و در نتیجه با همان ضرایب قبلی تعیین شده برای PID نمی‌توانستیم تعادل ربات را حفظ کنیم؛ و همچنین کاهش وزن ربات، تصمیم گرفتیم که به جای استفاده از باتری، از یک منبع تغذیه استفاده کنیم که در همه‌ی مدت زمان کار ربات بتوانیم ولتاژ ۱۲ را از آن دریافت کنیم و نیاز به تغییرات مداوم متناسب با عمر شارژ باتری‌ها نداشته باشیم. البته این تغییر باعث بروز محدودیت‌هایی هم شد منجمله اینکه وجود برق در محل تست ربات ضروری است و همچنین سیم‌های متصل دقت مضاعفی را در زمان تست مطالبه می‌کند تا اثری بر حرکت ربات نداشته باشند.



۸۵,۰۰۰ تومان

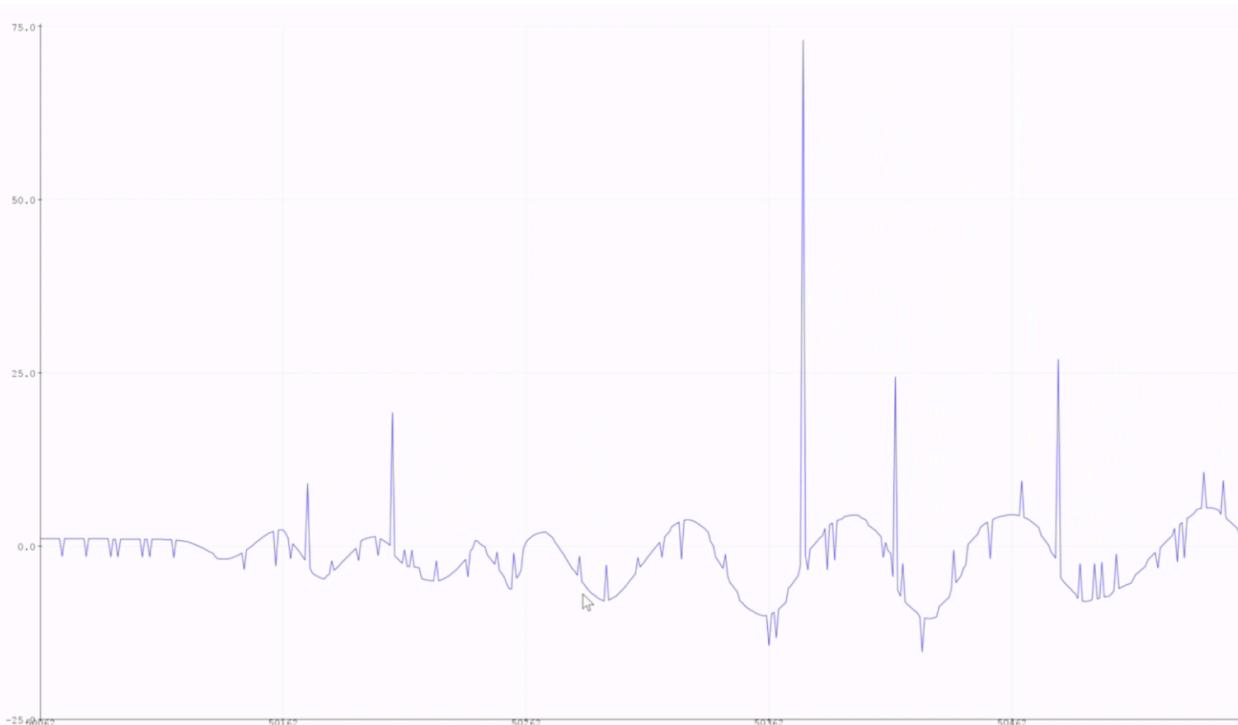
۲۰,۰۰۰ تومان

## شکست‌ها:

در حین تست ربات، زمانی که برق موتورها و Arduino از یک طریق تامین می‌شد، تغییر جهت ناگهانی حرکت چرخ‌ها باعث بروز اختلال در عملکرد پردازنده می‌شد، برای رفع این مشکل، منبع تغذیه موتورها و Arduino را جدا کردیم تا با جریان‌های الکتریکی مجزا از این اتفاق جلوگیری کنیم.

همچنین، یکسان نبودن گشتاور موتور چپ و راست، در عمل، باعث چرخیدن ربات می‌شد. برای ارائه‌ی راه حل این مشکل، از نرم‌افزار کمک گرفتیم و همانطور که قبله گفته شد، کنترل‌ری برای تعیین تفاوت سرعت دو موتور، و برگرداندن ربات به وضعیت اولیه اضافه کردیم.

علاوه بر موارد گفته شده، در ابتدای کار، از serial software برای گرفتن داده‌ها از MPUL6050 استفاده می‌کردیم. در لحظاتی، نویز بر روی داده‌های سنسور باعث حرکت اشتباه موتورها و در نتیجه از کنترل خارج شدن تعادل ربات می‌شد. داده‌ی خام، همانطور که در شکل زیر مشخص است؛ در فواصل کوتاه نویز زیادی داشت:

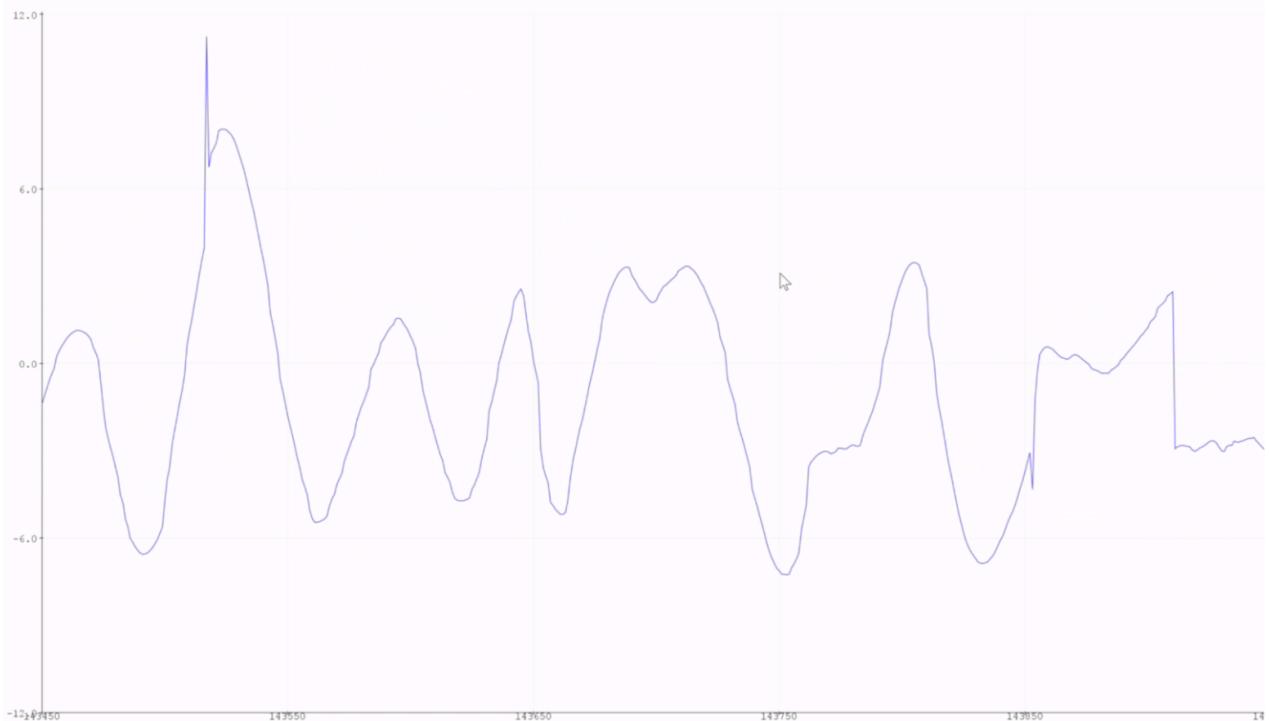


شکل (۱۵): نمودار داده‌های دریافتی از سنسور زاویه به صورت خام

نویزهای کوچک، باعث مشکل جدی در تعادل نمی‌شد، و فقط نوسانات ریز در حرکت ربات دیده می‌شد. اما نویزهای بزرگ، با اعمال pwm بزرگ، باعث سقوط سیستم می‌شوند. برای از بین بردن نویزهای موجود، از روش‌های مختلفی استفاده کردیم.

در مرحله‌ی اول، سه داده‌ی متوالی از مازلول دریافت کردیم؛ و نهایتاً میانه‌ی سه داده را به عنوان زاویه‌ی نهایی به PID برای تعیین pwm مناسب دادیم، در این روش، تعداد زیادی از نویزها حذف می‌شد، اما در لحظاتی اگر تعدادی نویز متوالی داشتیم، با همان مشکل به هم خوردن تعادل ربات درگیر بودیم.

تصویر زیر خروجی سیستم در این حالت است:

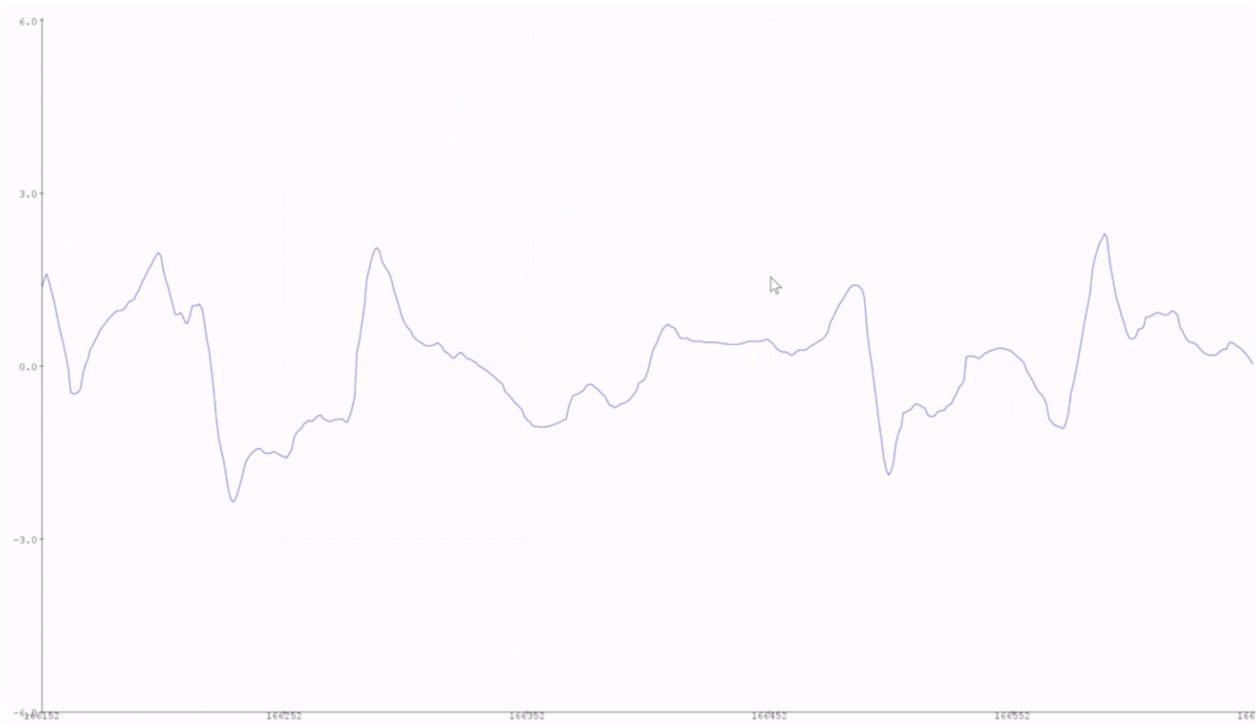


شکل (۱۶): نمودار داده‌های دریافتی از سنسور زاویه و حذف نویز با استفاده از میانه

در مرحله‌ی بعد، با بالا بردن تعداد نمونه‌ها، سعی در رفع مشکل گفته شده داشتیم. تعدادی از داده‌ها را دریافت می‌کردیم (بین ۵ تا ۱۰ داده) و با استفاده از الگوریتم Quick sort مقدار میانه را به دست می‌آوردیم. در این روش، نویزها رفع شدند، اما با پایین آمدن سرعت پردازش، ربات زمان کافی برای واکنش نداشت، و در نتیجه‌ی تأخیر در واکنش، تعادل ربات از بین می‌رفت.

بعد از امتحان روش‌های گفته شده، تلاش کردیم تا داده‌ی مازول را از Hardware serial بخوانیم؛ و عملکرد بسیار بهتر از حالت‌های پیشین بود. به نظر می‌رسد که در Sofware serial یا تعدادی داده از دست می‌روند، و یا دچار خطأ می‌شوند و درست به آردوبئینو انتقال پیدا نمی‌کنند. از این روی، دستورات کنترلی را که از اهمیت کمتری برخوردار بودند را با استفاده از Sofware serial به ربات انتقال دادیم و خواندن از سنسور را با استفاده از Hardware serial انجام دادیم.

نمونه‌ای از داده‌های دریافتی در این حالت در زیر قابل مشاهده است:



شکل (۱۵): نمودار داده‌های بدون نویز دریافتی از سنسور زاویه با استفاده از سریال سخت‌افزاری

### محیط و شرایط آزمایش:

برای حفظ تعادل ربات شرایط خاصی مورد نیاز نیست و تنها مشکل وجود نداشتن باتری برای ربات است که نیاز به منبع تغذیه وجود دارد، ربات را در سطوح‌های مختلفی تست کردیم که با توجه به نوع چرخ مورد استفاده در همه‌ی موارد تعادل ربات حفظ می‌شد که البته شاید در شرایط خاصی نیاز باشد تا ضرایب مربوط به کنترل تغییر کنند تا حفظ تعادل به صورت بهتری انجام شود. همچنین ربات را در سطح شیبدار هم تست کردیم که باز هم تعادل ربات حفظ می‌شد و ربات عمود بر سطح بود ولی ربات به سمت پایین سطح شیبدار کشیده می‌شد.

### نتایج

هدف اصلی پروژه تنها حفظ تعادل ربات بود، نگهداشتن ربات در یک نقطه، کنترل از راه دور ربات، جلوگیری از برخورد ربات با موانع از اهداف ثانویه و امنیازی پروژه بودند. ما با استفاده از برد آردوینو به هدف اصلی پروژه دست پیدا کردیم و تعادل ربات به خوبی حفظ می‌شد و حتی در مقابل ضربات نسبتاً شدید هم می‌توانست تعادل خود را حفظ کند. بخش اصلی و مهم برای حفظ تعادل، تنظیم مناسب ضرایب کنترلر بود. جدول زیر مقدارهای نهایی که به کمک روش‌های توضیح داده شده در بخش‌های قبل بدست آمده است را نمایش می‌دهد:

بعد از رسیدن به هدف اصلی پروژه برای اجرای بخش‌های اضافه و امتیازی تلاش کردیم و کدهای قسمت ثابت نگهداشت ربات در یک نقطه، کنترل از راه دور ربات را روی برد آردوینو پیاده سازی کردیم اما در تست‌های انجام شده این محاسبات اضافه باعث بالا رفتن بار پردازشی برد و کاهش تعادل ربات می‌شد، برای رسیدن به این اهداف از برد رزبری‌پای استفاده کردیم، کدهای تعادل ربات را دوباره با استفاده از زبان پایتون در رزبری‌پای پیاده‌سازی کردیم، با این‌کار به دلیل داشتن توان پردازشی بالاتر رزبری‌پای حتی نتایج بهتری نسبت به برد آردوینو به دست آمد، سپس کدهای مربوط به کنترل از راه دور ربات و نگهداشت ربات در یک نقطه را به آن اضافه کردیم که با این روش توانستیم جهت چرخش ربات را با استفاده از لپتاپ و از راه دور کنترل کنیم و حتی ربات را به جلو و عقب برانیم، همچنین ربات تلاش می‌کرد که در یک نقطه مرکزی ثابت باقی بماند و جایجایی‌های اضافه‌ی خود را جبران کند ولی در نهایت نتایج خیلی مطلوبی به دست نیامد و با اینکه ربات تلاش می‌کرد در یک نقطه ثابت بماند ولی تعادل آن کاهش پیدا می‌کرد.

ضرایب بدست آمده با استفاده از برد رزبری در جدول زیر قابل مشاهده است:

P	23
I	0.01
D	46

## هزینه نهایی

نام	تعداد	قیمت واحد	قیمت کل
بدنه (پلکسی ۲ میلیمتری)	۱	۲۰۰۰۰ تومان	۲۰۰۰۰ تومان
میله‌های فلزی برای ستون (spacer)	۱۶	۶۰۰ تومان	۱۰۰۰۰ تومان
Arduino Uno	۱	۲۵۰۰۰ تومان	۲۵۰۰۰ تومان
درایور موتور	۱	۱۵۰۰۰ تومان	۱۵۰۰۰ تومان
مجموعه‌ی موتور، گیربکس، چرخ	۲	۵۰۰۰ تومان	۵۰۰۰ تومان
ماژول GY-۳۵	۱	۳۸۰۰۰ تومان	۳۸۰۰۰ تومان
Bread board کوچک	۱	۴۰۰۰ تومان	۴۰۰۰ تومان
منبع تغذیه ۱۲ ولت با جریان ۵ آمپری	۱	۲۰۰۰۰ تومان	۲۰۰۰۰ تومان
ماژول HC-05	۱	۱۵۰۰۰ تومان	۱۵۰۰۰ تومان
مجموع			۱۵۵۰۰۰ تومان

## منابع:

- 1- <https://kth.diva-portal.org/smash/get/diva2:916184/FULLTEXT01.pdf>
- 2- <https://www.arduino.cc/en/Guide/Introduction>
- 3-<https://www.invensense.com/wp-content/uploads/2015/02/MPU-6000-Datasheet1.pdf>
- 4-<https://static1.squarespace.com/static/56b6357e01dbaea0266fe701/t/56c430b8859fd092ec1d23d3/1455698104834/Dennis-Jin-Development-of-a-stable-control-system-for-a-segway.pdf>
- 5- <http://wired.chillibasket.com/2015/03/pid-controller/>
- 6-<https://cdn.hackaday.io/files/16098688736832/Progress%20Report%20-%20AG%20-%209097951%20-%20final.pdf>
- 7-<https://www.mathworks.com/matlabcentral/fileexchange/58257-unified-tuning-of-pid-load-frequency-controller-for-multi-area-power-systems-via-imc>
- 8-<http://icircuit.net/arduino-boards-pin-mapping/141>
- 9- <http://www.bzarg.com/p/how-a-kalman-filter-works-in-pictures/>
- 10-[https://en.wikipedia.org/wiki/Kalman\\_filter](https://en.wikipedia.org/wiki/Kalman_filter)
- 11-[http://biorobotics.ri.cmu.edu/papers/sbp\\_papers/integrated3/kleeman\\_kalman\\_basics.pdf](http://biorobotics.ri.cmu.edu/papers/sbp_papers/integrated3/kleeman_kalman_basics.pdf)
- 12-<http://www.diyspacepk.com/product/dc-motor-and-robot-wheels/>
- 13- <https://uge-one.com/l298-dual-motor-driver.html>
- 14- <https://medium.com/pollenrobotics/how-to-read-a-dc-motors-datasheet-f70fa440452b>
- 15-<https://maker.pro/arduino/projects/build-arduino-self-balancing-robot> -
- 16- <http://elcplanet.com/آشنایی-با-موتورهای-dc/>
- 17-<https://maker.pro/arduino/projects/build-arduino-self-balancing-robot>
- 18-<https://www.hindawi.com/journals/mpe/2012/469491/>
- 19-<https://www.youtube.com/watch?v=poWP-RpPa3g>
- 20-<https://www.youtube.com/watch?v=g-dsHU0mBYc>
- 21-<https://www.youtube.com/watch?v=fFwkQZr8wnQ>
- 22-<https://www.youtube.com/watch?v=K2erYf9mqpE>
- 23-<https://bit.ly/2IXxb1S>
- 24-<https://www.youtube.com/watch?v=EwrQEsfmL4E>