



دانشگاه صنعتی امیرکبیر

(پلی تکنیک تهران)

دانشکده مهندسی برق

پروژه کارشناسی

گرایش کنترل

طراحی و ساخت ربات چهارپا

نگارش

محمد برآبادی

استاد راهنما

دکتر محمد اعظم خسروی

شهریور ۱۴۰۲

بِسْمِ اللّٰهِ الرَّحْمٰنِ الرَّحِيْمِ



به نام خدا

تاریخ: شهریور ۱۴۰۲

تعهدنامه اصالت اثر

دانشگاه صنعتی امیرکبیر
(پلی‌تکنیک تهران)

اینجانب محمد برآبادی متعهد می‌شوم که مطالب مندرج در این پایان‌نامه حاصل کار پژوهشی اینجانب تحت نظرارت و راهنمایی استادی دانشگاه صنعتی امیرکبیر بوده و به دستاوردهای دیگران که در این پژوهش از آنها استفاده شده است مطابق مقررات و روال متعارف ارجاع و در فهرست منابع و مأخذ ذکر گردیده است. این پایان‌نامه قبلاً برای احراز هیچ مدرک هم‌سطح یا بالاتر ارائه نگردیده است. در صورت اثبات تخلف در هر زمان، مدرک تحصیلی صادر شده توسط دانشگاه از درجه اعتبار ساقط بوده و دانشگاه حق پیگیری قانونی خواهد داشت.

کلیه نتایج و حقوق حاصل از این پایان‌نامه متعلق به دانشگاه صنعتی امیرکبیر می‌باشد. هرگونه استفاده از نتایج علمی و عملی، واگذاری اطلاعات به دیگران یا چاپ و تکثیر، نسخه‌برداری، ترجمه و اقتباس از این پایان‌نامه بدون موافقت کتبی دانشگاه صنعتی امیرکبیر ممنوع است. نقل مطالب با ذکر مأخذ بلامانع است.

محمد برآبادی

امضا

تقدیر و تشکر

تشکر قلبی و لسانی خود را از استاد عالی قدر جناب دکتر محمد اعظم خسروی که زحمت راهنمایی این پایان نامه را عهده دار گردیدند و در تمامی مراحل انجام رساله از راهنمایی های مددگرانه ایشان استفاده نمودم ابراز می دارم و توفیقات روز افرون ایشان را توأم با صحت و سعادت خواستارم.
از جناب دکتر مسعود شفیعی که زحمت داوری و تصحیح این پایان نامه را به عهده داشتند کمال سپاس را دارم.

از همگروهی عزیز مهندس سجاد قدیری که همواره یار و یاوری صبور بود کمال تشکر را دارم.
و در پایان از تمامی عزیزانی که در طول انجام این پژوهش مرا یاری کرده اند کمال تشکر و قدردانی را ابراز می نمایم.

چکیده

امروزه ربات‌ها نقش بهسزایی در زندگی انسان‌ها دارند. ربات‌های متحرک به دلیل نحوه تعامل‌شان با جهان، با سایر ماشین‌ها متفاوت هستند. آنها می‌توانند بر اساس اعمال خودشان تغییراتی در محیط اطراف خود ایجاد کنند و به دنیای اطراف خود پاسخ دهند. کاربردهای متنوع ربات‌ها در راستای ساده‌تر کردن زندگی انسان‌ها باعث شده مهندسین توجه ویژه‌ای به ربات‌ها داشته باشند.

از میان تمامی ربات‌های متحرک، ربات‌های چهارپا یک نوع ربات پادار هستند که به دلیل توانایی آن‌ها برای اکتشاف در همه انواع زمین‌ها، مشابه انسان و حیوانات، نسبت به ربات‌های چرخدار برتری دارند. مزیت ربات‌های چهارپا در این است که هنگام ایستادن نیز حالت پایدار خود را حفظ نموده، اما در هنگام راه رفتن احتیاج به کنترل کردن دارند.

در این پایان‌نامه، ابتدا با ربات‌ها و انواع آنها با تاکید بر ربات‌های چهارپای ساخته شده و کاربردشان آشنا شده و در ادامه وارد بحث طراحی مدل و ساخت ربات با استفاده از نرم‌افزار سالیدورکس شده‌ایم. در راستای این طراحی، یک مدل اولیه ایجاد شده و پس از رفع عیوب ربات، تلاش بر آن شد که عملکرد آن بهبود یابد و یک نمونه نهایی اصلاح شده پرینت شد. سپس به مبحث سخت‌افزار و قطعات استفاده شده مانند موتور، میکروکنترلر و مازول‌های مختلف در ربات پرداخته شده است. در پایان نیز الگوریتم‌های استفاده شده و برنامه‌نویسی روی میکروکنترلر ربات شرح داده شده‌اند.

واژه‌های کلیدی:

ربات متحرک^۱، ربات چهارپا^۲، ربات پادار^۳، سالیدورکس^۴، الگوریتم

¹Mobile Robot

²Quadrupedal Robot

³Legged Robot

⁴SolidWorks

فهرست مطالب

صفحه

عنوان

۱	۱	مقدمه
۲	۱-۱	مقدمه
۲	۲-۱	ربات و انواع آن
۸	۳-۱	ربات چهارپا
۸	۱-۳-۱	ویژگی‌های ربات چهارپا
۸	۲-۳-۱	موارد کاربرد ربات‌های چهارپا
۹	۳-۳-۱	نمونه‌های پیشین
۱۶	۴-۱	جمع بندی
۱۷	۵-۱	ساختار پایان نامه
۱۸	۲	طراحی ربات
۱۹	۱-۲	مقدمه
۱۹	۲-۲	الگوبرداری و انتخاب مدل
۲۰	۳-۲	نرمافزار سالیدورکس
۲۱	۴-۲	طراحی و محاسبه ابعاد قطعات
۲۲	۱-۴-۲	نمونه اولیه
۲۵	۲-۴-۲	بهبود مدل
۲۸	۳-۴-۲	نمونه نهایی
۲۹	۵-۲	جمع بندی
۳۰	۳	سخت‌افزار ربات
۳۱	۱-۳	مقدمه
۳۱	۲-۳	سرو موتور
۳۲	۱-۲-۳	اجزای سرو موتور
۳۲	۲-۲-۳	انواع سرو موتور
۳۴	۳-۳	میکروکنترلر
۳۵	۱-۳-۳	انواع میکروکنترلر
۳۶	۲-۳-۳	مزایا
۳۶	۳-۳-۳	کاربرد
۳۷	۴-۳	ماژول STM32F103C8T6
۳۸	۵-۳	ماژول Arduino-Uno
۳۸	۶-۳	ماژول ESP32-CAM
۳۹	۷-۳	منبع تغذیه دی سی

٤٠	برد الکترونیکی	٨-٣
٤٠	Breadboard	١-٨-٣
٤٠	برد PCB	٢-٨-٣
٤١	هزار سوراخ	٣-٨-٣
٤٢	جمع بندی	٩-٣
٤٣	نرم افزار و الگوریتم های مورد استفاده	٤
٤٤	مقدمه	١-٤
٤٤	پروتکل UART	٢-٤
٤٥	تکنیک کنترلی PWM	٣-٤
٤٥	PWM مفهوم	١-٣-٤
٤٦	کاربرد PWM در سرورموتور	٢-٣-٤
٤٧	نرم افزار CubeMX	٤-٤
٤٧	مفهوم و کاربرد	١-٤-٤
٤٧	کتابخانه های برنامه نویسی	٢-٤-٤
٤٩	کافیگ پایه های میکرو کنترلر	٣-٤-٤
٤٩	تنظیم کلاک	٤-٤-٤
٥٠	تنظیمات موردنیاز برای ربات چهار پا	٥-٤-٤
٥٣	برنامه نویسی در نرم افزار Keil	٥-٤
٥٣	نرم افزار Keil چیست؟	١-٥-٤
٥٣	تنظیم تایم رها	٢-٥-٤
٥٤	تابع حرکت موتور	٣-٥-٤
٥٦	چهار حرکت اصلی ربات	٤-٥-٤
٥٩	عملکرد کلی ربات	٥-٥-٤
٦١	نتیجه گیری و پیشنهادات	٥
٦٢	نتیجه گیری	١-٥
٦٥	پیشنهادات	٢-٥
٦٦	منابع و مراجع	
٦٩	پیوست	

فهرست تصاویر

صفحه

شکل

۱-۱	نمونه ربات صنعتی	۳
۲-۱	نمونه‌ای از ربات خانگی	۴
۳-۱	ربات جراح	۴
۴-۱	یک نمونه ربات سرگرمی	۴
۵-۱	نمونه‌ای از ربات خودران	۵
۶-۱	ربات نظامی	۵
۷-۱	ربات آموزگار	۶
۸-۱	یک نمونه ربات کشاورز	۶
۹-۱	یک نمونه ربات پزشکی	۷
۱۰-۱	ربات انسان‌نما	۷
۱۱-۱	ربات چهارپایی آبیو	۹
۱۲-۱	ربات جیداگ	۹
۱۳-۱	ربات سگ کوچک	۱۰
۱۴-۱	ربات سگ بزرگ	۱۱
۱۵-۱	حسگرهای ربات سگ بزرگ و محل قرارگیری آنها	۱۲
۱۶-۱	معماری نرمافزار ربات سگ بزرگ	۱۲
۱۷-۱	دو تصویر از ربات سامانه پشتیبان گروه	۱۳
۱۸-۱	گربه وحشی	۱۳
۱۹-۱	ربات چهارپایی پیرو	۱۴
۲۰-۱	نمای ربات آیدین (۱ راست) و آیدین (۲ چپ)	۱۴
۲۱-۱	ربات چهارپایی هانما	۱۵
۲۲-۱	ربات چهارپایی هایکیو	۱۶
۱-۲	ربات چهارپا	۱۹
۲-۲	ربات چهارپایی Titan VIII	۲۰
۳-۲	لایه بالایی و پایینی نمونه اولیه	۲۲
۴-۲	لینک اول نمونه اولیه	۲۳
۵-۲	لینک دوم نمونه اولیه	۲۳
۶-۲	لینک سوم نمونه اولیه	۲۴
۷-۲	نسخه اولیه ربات	۲۴
۸-۲	لایه بالایی و پایینی نمونه نهایی	۲۵
۹-۲	لینک اول نمونه نهایی	۲۶

۲۷	۱۰-۲ لینک دوم نمونه نهایی
۲۷	۱۱-۲ لینک سوم نمونه نهایی
۲۸	۱۲-۲ نسخه نهایی ربات
۲۸	۱۳-۲ قطعات پرینت شده ربات
۲۹	۱۴-۲ ربات اسمبل شده نهایی
۳۱	۱-۳ سرو موتور SG90
۳۴	۲-۳ اجزای میکروکنترلر
۳۵	۳-۳ انواع میکروکنترلر
۳۶	۴-۳ کاربردهای میکروکنترلر
۳۷	۵-۳ ماژول STM32F103C8T6
۳۸	۶-۳ ماژول Arduino Uno
۳۹	۷-۳ ماژول ESP32-CAM
۳۹	۸-۳ منبع ولتاژ DC
۴۰	۹-۳ Breadboard
۴۱	۱۰-۳ یک نمونه برد مدار چاپی
۴۱	۱۱-۳ یک نمونه برد هزار سوراخ
۴۴	۱-۴ سیگنال های فرستنده گیرنده UART
۴۵	۲-۴ رابط سریال UART
۴۶	۳-۴ چرخه کار Duty Cycle
۴۷	۴-۴ فرکانس و دوره تناوب SG90
۴۹	۵-۴ کانفیگ میکروکنترلر
۵۰	۶-۴ تنظیم کلاک
۵۱	۷-۴ تنظیمات تایمر
۵۲	۸-۴ تنظیمات ارتباط سریال
۶۲	۱-۵ ربات نهایی
۶۳	۲-۵ ربات نهایی
۶۴	۳-۵ ربات بدون هیچ فرمان
۶۴	۴-۵ ربات در حالت اولیه

فهرست نمادها

نماد	مفهوم
$C++$	نوعی زبان برنامه‌نویسی
C	نوعی زبان برنامه‌نویسی
mm	میلی‌متر
cm	سانسی‌متر
PB	پورت B
CLK	کلک

مقدمه

فصل اول

۱-۱ مقدمه

با پیشرفت‌های چشمگیر در فناوری و مهندسی در چند دهه اخیر، یک انقلاب قابل توجه در زمینه رباتیک رخ داده است. رباتیک به عنوان یک زمینه چندجانبه و قابل اجرا، شامل تحقیقات و توسعه در طراحی، ساخت، کنترل و بهینه‌سازی ربات‌ها می‌شود. با ترکیب فناوری‌های مختلف از جمله مکانیک، الکترونیک، نرم‌افزار و هوش مصنوعی، ربات‌ها به وجود می‌آیند که قادر به انجام طیف گسترده‌ای از وظایف هستند؛ از فعالیت‌های صنعتی و خدماتی تا کارهای پزشکی، کشاورزی و حتی بررسی فضا.

امروزه از علم رباتیک برای ایجاد دستگاه‌های خودکار به کمک هوش مصنوعی که بتوانند وظایفی مشابه انسان‌ها را با دقت و کارایی بیشتری انجام دهند، استفاده می‌شود. برای رسیدن به این هدف، محققان و مهندسان در زمینه رباتیک تلاش می‌کنند تا مکانیسم‌های پیچیده‌تری برای حرکت، احساس، تصمیم‌گیری و تعامل با محیط ایجاد کنند.

۲-۱ ربات و انواع آن [۲۱]

ربات یک ماشین است (به خصوص ماشینی که توسط کامپیوتر قابل برنامه‌نویسی باشد) که می‌تواند مجموعه کارهای پیچیده‌ای را به صورت خودکار انجام دهد. ربات‌ها ممکن است توسط یک دستگاه کنترل خارجی، کنترل شوند یا این که دستگاه کنترلی در داخل آن‌ها قرار بگیرد. ربات‌ها ممکن است به گونه‌ای ساخته شوند که ظاهری شبیه به انسان داشته باشند اما بیشتر ربات‌ها، ماشین‌هایی هستند که برای انجام کاری ساخته می‌شوند و ظاهر آن‌ها اهمیتی ندارد.

ربات‌ها ممکن است خودران^۱ یا نیمه خودران^۲ باشند و انواع مختلفی دارند؛ از قبیل ربات‌های انسان‌نما، مانند ربات ASIMO شرکت هوندا و ربات پینگ‌بونگ باز شرکت TOSY ، ربات‌های صنعتی، ربات‌های جراحی پزشکی، ربات‌های کوچک که از هوش جمعی بهره می‌برند، پهبد هایی مانند هواپیمای بدون سرنشیں Predator MQ-1 که توسط نیروی هوایی ایالات متحده مورداستفاده قرار می‌گیرد و حتی ربات‌های میکروسکوپی (نانو ربات‌ها) [۲۱]. ربات‌ها ممکن است با تقلید از ظاهر موجودات زنده و یا شبیه‌سازی حرکات آن‌ها، حس هوشمند بودن و یا توانایی فکر کردن را به انسان القا کنند. انتظار می‌رود تا در دهه آتی، اشیا خودران گسترش چشمگیری پیدا کنند. از ارکان اصلی این گسترش می‌توان به ربات‌های خانگی و اتومبیل‌های خودران اشاره کرد.

¹Autonomous

²Semi-Autonomous

ربات‌ها انواع گوناگونی دارند و هر نوع برای کاربردهای خاصی طراحی شده است. در ادامه به بررسی برخی از انواع مختلف ربات‌ها و ویژگی‌های منحصر به فرد آنها می‌پردازیم:

۱- ربات‌های صنعتی: امروزه پیچیدگی‌های فرایند تولید در صنایع مختلف، افزایش حجم کار و البته رقابتی بودن بازار، منجر به جایگزینی ربات صنعتی با نیروی انسانی شده است. واقعیت این است که در نظر گرفتن حفظ سلامت نیروی کار، جلوگیری از بروز خطا حین فرایند تولید و افزایش سرعت ایجاب می‌کند که از ربات‌ها به ویژه ربات صنعتی به عنوان بهترین جایگزین برای نیروی انسانی استفاده شود.

ربات صنعتی در واقع یک سیستم اتوماتیک است که برای انجام مراحل مختلف فرآیند تولید محصول در انواع صنایع استفاده می‌شود. اغلب این ربات‌ها به طور کامل قابل برنامه‌ریزی هستند و سرعت و دقت انجام کار را به طور چشم‌گیری افزایش می‌دهند. ربات‌های صنعتی در ابعاد مختلف تولید می‌شوند؛ از نانو و میکرو گرفته تا ابعاد بزرگتر و کارخانه‌ای، قابلیت اضافه شدن به خط تولید را دارند.



شکل ۱-۱: نمونه ربات صنعتی

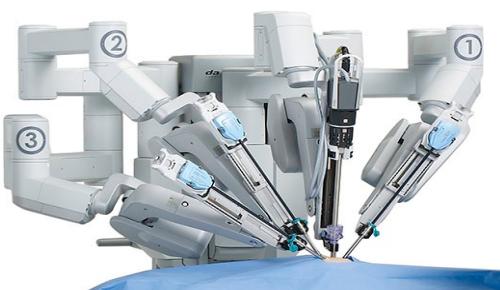
۲- ربات‌های خدماتی: ربات‌های خدماتی، نوعی سیستم هوش مصنوعی، به عنوان یک پیشرفت فناوری مهم در سال‌های اخیر ظهرور کرده‌اند. این ربات‌ها برای انجام وظایف و عملکردهای متنوع در محیط‌های مختلف طراحی و برنامه‌ریزی شده‌اند تا زندگی انسان‌ها را بهبود بخشند و خدمات گوناگونی را فراهم آورند. برخلاف ربات‌های صنعتی که عمدتاً در کارخانه‌ها عمل می‌کنند، ربات‌های خدماتی برای تعامل مستقیم با انسان‌ها طراحی شده‌اند و در محیط‌های مختلفی نظیر خانه‌ها، مراکز درمانی، بخش هتل‌ها، فضاهای عمومی و غیره عمل می‌کنند.

- ربات‌های خانگی: این ربات‌ها در کمک به کارهای خانگی نظیر جاروکردن، کوتاه کردن چمن و تمیز کردن استفاده می‌شوند.



شکل ۱-۲: نمونه‌ای از ربات خانگی

- ربات‌های پزشکی: در محیط‌های درمانی برای انجام عملیات‌های جراحی، مراقبت از بیماران و بازسازی استفاده می‌شوند.



شکل ۱-۳: ربات جراح

- ربات‌های سرگرمی: برای اهداف سرگرمی طراحی شده‌اند، از اسباب‌بازی‌ها و همراهان رباتی تا کاربردهای سرگرمی دیگر.



شکل ۱-۴: یک نمونه ربات سرگرمی

۳- **وسایل خودران:** ربات‌های خودران می‌توانند وظایف خود را به صورت کاملاً خودکار و بدون نیاز به ناظارت و کنترل انسان انجام دهند. این ربات‌ها معمولاً برای فعالیت در محیط‌های باز که فعالیت کردن در آن‌ها نیازی به ناظارت انسان ندارد، طراحی شده‌اند. چنین ربات‌هایی طراحی منحصر به‌فردی دارند؛ زیرا با برخورداری از سنسورهای مختلف می‌توانند محیط اطراف خود را به خوبی درک کنند و سپس با بهره‌مندی با آن دسته از تجهیزات خود که برای تصمیم‌گیری به ربات کمک می‌کنند (معمولًا کامپیوترهای خاصی این کار را انجام می‌دهند)، بر اساس داده‌های در اختیار خود و همچنین مأموریتی که به آن‌ها محول شده است، تصمیم مناسب را برای انجام کار بعدی به بهترین شکل ممکن می‌گیرند.



شکل ۱-۵: نمونه‌ای از ربات خودران

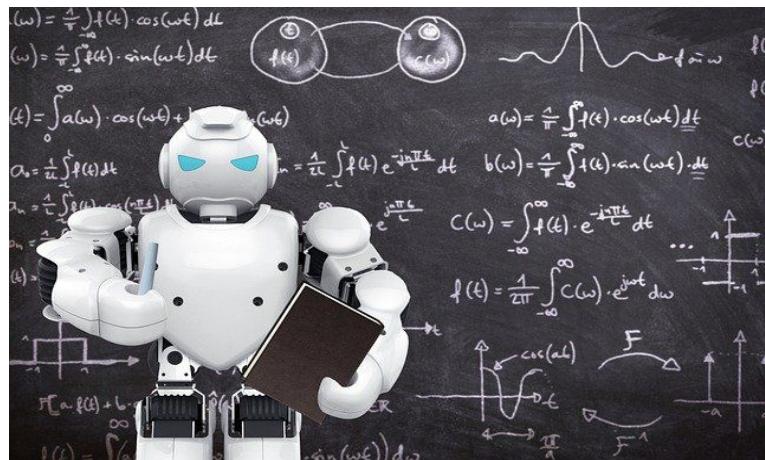
۴- **ربات‌های نظامی و دفاعی:** این ربات‌ها برای کاربردهای نظامی طراحی شده‌اند، از جمله ناظارت، انجام مأموریت‌های مهم و مبارزه از راه دور. آنها در محیط‌های چالشی عمل می‌کنند تا خطرات برای سربازان انسانی را کمینه کنند. سلاح‌ها و تجهیزات نظامی رباتیک مجهز به هوش مصنوعی نیز می‌توانند یک برگ برنده طلایی برای پیروزی در جنگ‌ها باشند؛ به عنوان مثال سامانه پیشرفته نظامی مدلار مارس MAARS^۱ با ظاهری شبیه به تانک مجهز به قابلیت پرتاب گاز اشک‌آور و انتشار اشعه لیزر جهت سوردرگم کردن دشمن و همچنین یک نارنجک‌انداز برای زمین‌گیر کردن دشمن است [۲۱].



شکل ۱-۶: ربات نظامی

^۱Modular Advanced Armed Robotic System

۵- ربات‌های آموزشی: ربات‌های آموزشی برای آموزش برنامه‌نویسی، مهندسی و مهارت‌های حل مسئله به دانش‌آموزان استفاده می‌شوند. آنها از کیت‌های ساده برای مبتدیان تا پلتفرم‌های پیچیده‌تر برای دانشجویان پیشرفته متفاوتی دارند.



شکل ۱-۷: ربات آموزگار

۶- ربات‌های کشاورزی: انجام برخی از امور کشاورزی توسط کشاورزان وقت زیادی از آن‌ها می‌گیرد. چنانچه آن‌ها برای انجام چنین کارهایی از ربات‌ها کمک بگیرند، شاهد نتیجه بهتری خواهند بود. این امور شامل فعالیت‌های مرتبط با کاشت و بذرپاشی، کنترل علفهای هرز، برداشت و سایر فعالیت‌های مشابه می‌شود.

برداشت محصولات کشاورزی با کمک ربات‌ها در مقایسه با انجام این کار به صورت سنتی و توسط کشاورزان، نیز نتیجه بهتری دارد. برای انجام این کار نیز ربات‌های مخصوصی طراحی شده‌اند. برخی از این ربات‌ها دارای بازوی مخصوص برای برداشت و چیدن میوه‌های تابستانی حساس با بافت نرم مثل انگور، انواع توت‌ها و... جهت جلوگیری از آسیب دیدن آن‌ها هستند.



شکل ۱-۸: یک نمونه ربات کشاورز

۷- ربات‌های پزشکی: ربات‌ها در این حوزه می‌توانند به عنوان دستیار جراح استفاده شوند. آن‌ها می‌توانند در جراحی قسمت‌های حساس بدن مثل گردن و ستون فقرات که کوچک‌ترین اشتباه در جراحی آن‌ها آسیب‌های جبران ناپذیری را به بار می‌آورد، کمک دست جراحان شوند و دقیق‌ترین اشتباه در جراحی را افزایش دهنند. حتی برخی از جراحی‌ها به قدری دقیق بالایی می‌طلبد که انسان قادر به انجام آن‌ها نیست و برای انجام آن‌ها ناگزیر باید از ربات استفاده کند؛ به عنوان مثال شرکت نورالینک^۱ قصد دارد یک تراشه مغزی برای برقراری ارتباط بین مغز انسان و دستگاه‌های مختلف جهت کنترل ذهن‌ها و همچنین درمان اختلالات و بیماری‌های مغزی عرضه کند و جراحی لازم برای قرار دادن این تراشه مغزی به قدری دقیق زیادی می‌طلبد که انسان قادر به انجام آن نیست و نورالینک یک ربات برای این جراحی نیز طراحی کرده است.



شکل ۹-۱: یک نمونه ربات پزشکی

۸- ربات‌های انسان‌نما: ربات‌های انسان‌نما ظاهری شبیه به انسان دارند کارهای و رفتارهای او را تقلید می‌کنند. این ربات‌ها قادر به انجام فعالیت‌های شبیه به فعالیت‌های انسانی هستند که از میان آن‌ها می‌توان به دویدن، پریدن و حمل اشیا اشاره کرد. ظاهر برخی از ربات‌های انسان‌نما کاملاً شبیه انسان‌های واقعی است و اگر به صورت گذرا به آن‌ها نگاه کنیم یا از دور آن‌ها را بینیم، اصلاً متوجه این موضوع نمی‌شویم.



شکل ۱۰-۱: ربات انسان‌نما

¹Neuralink

۱-۳ ربات چهارپا

در دنیای رباتیک، انواع مختلفی از ربات‌ها وجود دارند که با توجه به تنوع ویژگی‌هایشان، برای انجام وظایف گوناگون طراحی و ساخته می‌شوند. یکی از این انواع ربات‌ها که در سال‌های اخیر توجه زیادی به خود جلب کرده است، ربات‌های چهارپا (ربات‌های با چهار پا) هستند. این ربات‌ها با تقلید از حرکات و رفتار حیوانات دارای چهار پا مانند سگ، گربه و حتی عنکبوت‌ها طراحی می‌شوند.

۱-۳-۱ ویژگی‌های ربات چهارپا

تمایل انسان به به کارگیری ظرفیت‌های طبیعت، بدست آوردن فهم بهتر از راه رفتن و حرکات چابک جانوران طبیعی، و الگوبرداری و شبیه‌سازی ماشینی رفتار حیوانات از جمله دلایل رویکرد به ربات‌های پادار است. در میان ربات‌های پادار، ربات‌های چهارپا به علت پایداری مناسب‌تر نسبت به ربات‌های دو پا و نیز تعداد پاهای کمتر نسبت به ربات‌های شش پا، پیچیدگی کمتر را در طراحی داشته و مانورهای مختلف ممکن را که از ربات‌های پادار انتظار می‌رود دارا است.

ربات‌های چهارپا نسبت به ربات‌های چرخ‌دار و شنی‌دار دارای مزایایی هستند که استفاده از آنها را در برخی زمینه‌ها توجیه می‌کند. از جمله این قابلیت‌ها می‌توان از سازگاری با محیط‌های ناهموار، جایجایی کارآمد، تعليق پویا و فعال با زاویه دادن به بندگاه‌ها^۱، عبور از موانع، مصرف یکسان انرژی در محیط‌های هموار و ناهموار، امکان کاهش سر خوردگی با تماس عمودی با زمین نام برد.

با وجود مزایای ذکر شده، ربات‌های پادار راه حل کامل مسئله حرکت نیست و مشکلات آنها باعث شده تا به کارگیری آنها در صنایع و بخش‌های خدماتی با کندی صورت گیرد. از جمله این مشکلات پیچیدگی، توان مصرفی بالا در مقایسه با ربات‌های چرخ‌دار در محیط‌های هموار، و سرعت کم آنها است.

۱-۳-۲ موارد کاربرد ربات‌های چهارپا

موارد کاربرد ربات‌های چهارپا به مزیت‌های آنان نسبت به ربات‌های چرخ‌دار باز می‌گردد که شامل کمک به جمع‌آوری اطلاعات در امداد و نجات شهری به هنگام زلزله، آتش‌سوزی، و یا حوادث تروریستی و در جاهایی که حضور پرسنل انسانی دارای ریسک می‌باشد با امکان بالا رفتن و عبور از سطوح ناهموار، حمل بار از نقاط صعب العبور با کنترل از راه دور، اکتشافات سطحی و جستجو در کف دریاها، سطوح کرات دیگر، قابلیت استفاده در محیط‌های پرمانع مانند محیط‌های جنگلی، کمک به افراد معلول در حرکت در محیط‌های ناهموار و نیز به عنوان ربات‌های توان‌بخشی می‌باشد [۱۹].

¹Joints

۳-۳-۱ نمونه‌های پیشین

- ربات آیبو^۱: آیبو یک ربات کوچک چهارپا به شکل سگ است که در شرکت سونی ساخته شده است. پروژه آیبو از سال ۱۹۹۰ شروع شده و مدل‌های مختلفی از آن ساخته شد. مدلی از این ربات که در شکل ۱۱-۱ مشاهده می‌شود در سال ۲۰۰۳ به منظور مسابقات جهانی ارائه گردید. این ربات به ازای هرپا دارای سه درجه آزادی است و با هدف سرگرمی ساخته شده است که در طراحی این ربات از سرو موتورهای جدیدی با گشتاور بالا استفاده شده است. این ربات برای یک ساعت حداقل بدون شارژ کردن مجدد می‌تواند کار کند^[۷].



شکل ۱۱-۱: ربات چهارپای آیبو^[۷]

- ربات جی-داگ^۲: کمپانی رباتیکی ژاپنی HPI در حال تبلیغ جدیدترین محصول تجاری خود به نام جی-داگ ۳ با قیمتی معادل ۵۵۰ یورو می‌باشد که در شکل ۱۲-۱ مشاهده می‌شود. واحد پردازشگر این ربات از نوع 11-RPU است و ارتفاع ربات به ۱۹ سانتیمتر می‌رسد. از طریق درگاه RS232 می‌توان اتصال میان ربات و کامپوتر برقرار و حرکات آن را برنامه‌ریزی نمود. این ربات دارای نه سرو موتور است که در هر پا دو موتور قرار گرفته است که در کل نه درجه آزادی به این ربات می‌دهد. قابلیت اضافه کردن موتورهای اضافه نیز به این ربات وجود دارد^[۶].

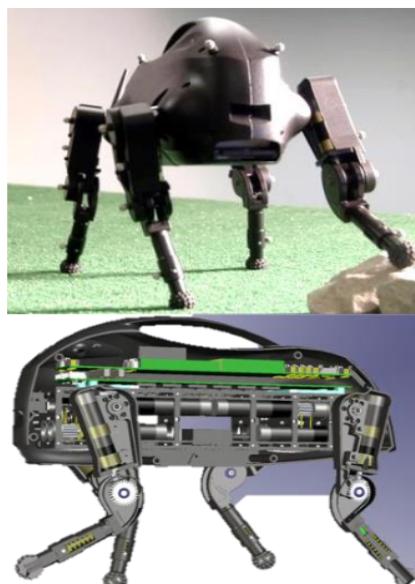


شکل ۱۲-۱: ربات جی‌داگ^[۶]

^۱AIBO

^۲G-Dog

- سگ کوچک^۱: سگ کوچک یک ربات چهارپاست که به منظور انجام تحقیقات در خصوص یادگیری گامبرداری^۲ به خصوص در زمینهای ناهموار توسط شرکت بوستون داینامیکس در سال ۲۰۰۶ میلادی با حمایت DARPA^۳ ساخته شده [۸] که در شکل ۱۳-۱ نمایش داده شده است. ربات سگ کوچک دارای ۳۰ سانتیمتر طول، ۱۸ سانتیمتر عرض، و ۲۶ سانتیمتر ارتفاع است و وزن آن تقریباً برابر با ۲/۵ کیلوگرم است [۱۴]. این ربات دارای سه موتور الکتریکی برای کنترل هر پا می‌باشد که مجموعاً ۱۲ درجه آزادی را به آن می‌دهند. در طراحی این ربات سعی شده که موتورها تا حد امکان در نزدیکی بدن قرار گرفته تا اندازه پاهای کوچک بمانند. همچنین از یک فر برای گرفتن ارتعاشات سطوح در پایین هر پا استفاده شده است. هر کدام از راهاندازنه موتورها با فرکانس ۱۰۰ اهرتز با یک کنترلگر PD زاویه و یا گشتاور توسط یک کامپیوتر تحت سیستم عامل Linux کنترل می‌گردد. سگ کوچک دارای چهار حسگر تماس در پاهای و یک حسگر مجاورتی^۴ در سر و حسگر زاویه در تمامی بندگاه‌ها، و یک واحد اندازه‌گیری اینرسی (IMU)^۵ برای تنظیم موقعیت بدن و شتاب آن است. سگ کوچک دارای یک باتری قابل شارژ است که انرژی آن را برای ۳۰ دقیقه حرکت مستمر تأمین می‌نماید [۱۴].



شکل ۱۳-۱: ربات سگ کوچک [۸]

¹Little Dog

²Locomotion

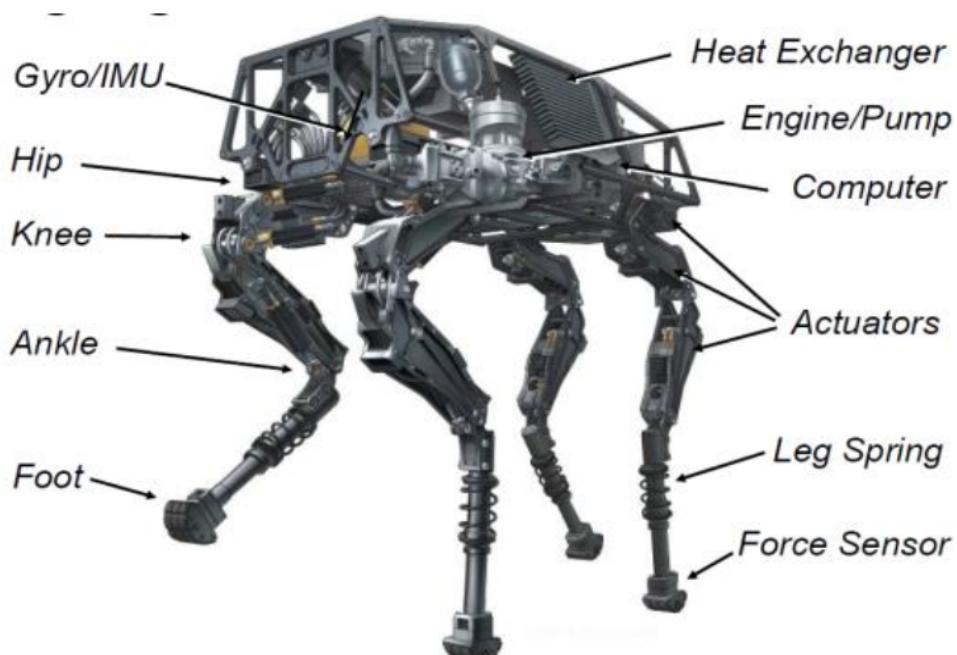
³Defense Advanced Research Projects Agency

⁴Proximity

⁵Inertial Measuring Unit

- سگ بزرگ^۱: ربات سگ بزرگ محصول شرکت بوستون داینامیکس^۲ است و دارای قابلیت‌های منحصر به فردی در مانور دادن در محیط‌های طبیعی و خشن می‌باشد. این ربات قادر است بارهای نسبتاً سنگین تا حدود دو برابر وزن خود را در زمین‌های شیب دار، برفی و یخ‌زده حمل نماید. ربات سگ بزرگ قادر به دویدن می‌باشد و در حال حاضر به عنوان پیشرفته‌ترین ربات چهارپا در جهان شناخته می‌شود. ساخت این ربات تحت حمایت ۳۰ میلیون دلاری آژانس طرح‌های پژوهشی پیشرفته دفاعی DARPA انجام گردیده است و در افغانستان نیز عملیات آزمایشی انجام داده است.

ربات سگ بزرگ ۱۰۹ کیلوگرم وزن دارد و ۱ متر ارتفاع، ۱/۱ متر طول و ۰/۳ متر عرض دارد. این ربات قادر است ۵۰ کیلوگرم بار را حمل نماید و نیز قادر است بدون بار مسافت ۱۰ کیلومتر را در ۲/۵ ساعت بدون سوخت‌گیری طی نماید. این چهارپا قادر به پرش تا ۱/۱ متر نیز می‌باشد. همان‌گونه که در شکل ۱۴-۱ مشاهده می‌شود، هر پای این ربات دارای ۴ درجه آزادی فعال (دو عدد در لگن، و یکی در زانو و یکی در مج) و یک درجه آزادی غیر فعال (در فتر مج پا) است. نیروی محرکه آن توسط یک موتور بنزینی تک سیلندر ۱۵ اسب بخار و سیستم هیدرولیک با فشار ۳۰۰۰ psi تأمین می‌شود. هر پا توسط چهار شیر سرو الکتروهیدرولیک دو مرحله‌ای کنترل می‌گردد [۱۶].



شکل ۱۴-۱: ربات سگ بزرگ [۱۶]

¹Big Dog

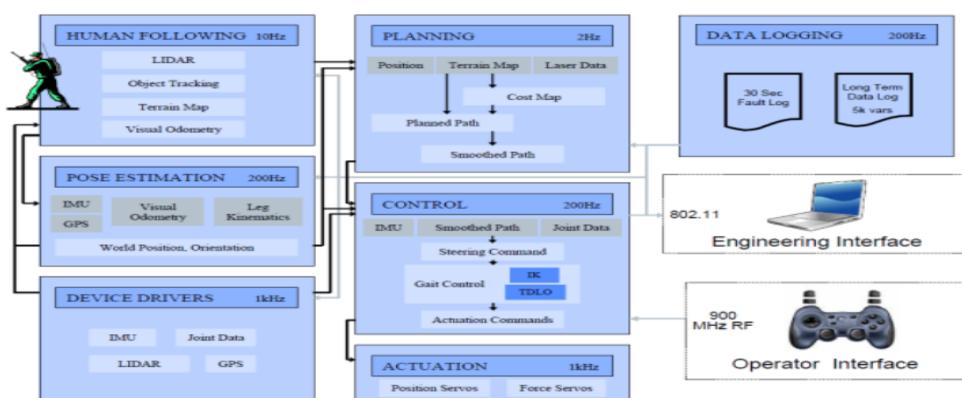
²Boston Dynamics

سگ بزرگ داری حسگرهای متعددی است که شامل: ۱۶ پتانسیومتر خطی برای اندازه‌گیری تغییر زوایای بندگاههای زانو، لگن، و مج پا، ۱۶ عدد حسگر بار^۱ برای پاها، ۱۶ عدد حسگر جریان برای پایش جریان شیرهای سرو، ۳ عدد دوربین برای بینایی سه بعدی، یک عدد رادار لیزری (لیدار)^۲، یک عدد ژایرسکوپ^۳ که تغییرات سه زاویه و شتاب خطی سه محور را می‌دهد، دو عدد حسگر ولتاژ باطری می‌باشد. شکل ۱۵-۱ محل قرار گیری حسگرهای سگ بزرگ نشان می‌دهد.



شکل ۱۵-۱: حسگرهای ربات سگ بزرگ و محل قرارگیری آنها

کامپیوتر روی ربات به صورت سفارشی تهیه شده است و دارای پردازشگر پنتیوم با سیستم عامل بلادرنگ QNX می‌باشد. کد برنامه به زبان C++ نوشته شده است که اطلاعات حسگرهای را دریافت و کنترل لازم به راهاندازها را اعمال می‌کند و نیز ارتباط با کاربر را نیز فراهم می‌نماید. معماری نرم افزار ربات سگ بزرگ در شکل ۱۶-۱ نمایش داده شده است.



شکل ۱۶-۱: معماری نرم افزار ربات سگ بزرگ

^۱Load cell

^۲Lidar

^۳Gyroscope

فرکانس نمونه برداری برای اطلاعات مربوط به مسیریابی ۲ هرتز، فاصله یابی و تشخیص موائع ۱۰ هرتز، تخمین موقعیت، کنترل و ثبت داده ۲۰۰ هرتز، حرکت‌ها و راه اندازها یک کیلو هرتز می‌باشد. همچنین جلیقه‌ای برای کاربر ربات طراحی گردیده است که با تجهیزات نصب شده بر روی آن کاربر می‌تواند از طریق یک ارتباط رادیویی بی‌سیم ۹۰۰ مگاهرتز اطلاعات دریافت شده توسط ربات را پایش کرده، و از طریق یک اهرم کنترل، حرکت ربات را کنترل کند.

نمونه پیشرفته تر ربات سگ بزرگ به نام سامانه پشتیبان گروهان (LS3)^۱ یا سگ آلفا و نیز گربه وحشی^۲ می‌باشد که اخیراً توسط همین شرکت و تحت حمایت وزارت دفاع آمریکا ساخته شده است که در شکل ۱۷-۱ و ۱۸-۱ قابل مشاهده‌اند [۴] [۱]. این ربات‌ها بیشتر جنبه نمایشی نظامی داشته و هنوز اطلاعات مستندی در مورد آنها در مقالات علمی موجود نیست. گربه وحشی نمونه پیشرفته تر ربات یوزپلنگ MIT است و هم اکنون مراحل آزمون و اصلاحات خود را طی می‌کند و قرار است تا سرعت ۵۰ کیلومتر در ساعت بدد.



شکل ۱۷-۱: دو تصویر از ربات سامانه پشتیبان گروه [۴]



شکل ۱۸-۱: گربه وحشی [۱]

^۱Legged Squad Support System

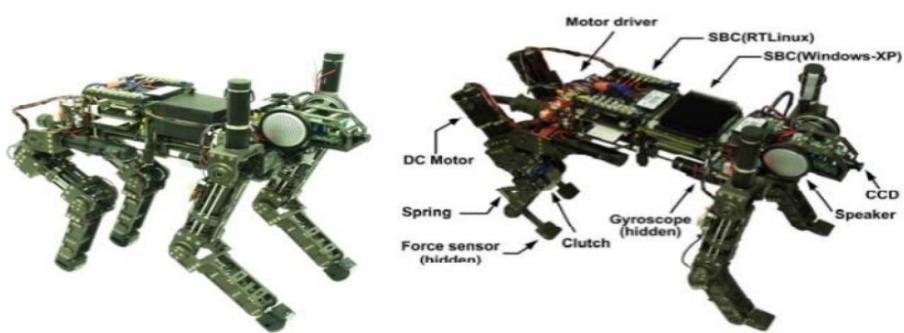
^۲WildCat

• **چهارپای پیرو:** ربات پیرو^۱ که به طول ۵۰ سانتیمتر می‌باشد، محصول استیتو کره‌ای پوهانگ^۲ است. ربات چهارپای پیرو برای انجام عملیاتی نظیر جستجو، کشف و حمل بار در شرایط طبیعی در نظر گرفته شده است (شکل ۱۹-۱). از آنجا که این ربات شباهت زیادی به ربات سگ بزرگ دارد، بسیاری آن را نمونه کره‌ای سگ بزرگ قلمداد کرده و این امر را نشانه علاقه کره به رقابت در عرصه رباتیک در تراز نخست می‌دانند. این ربات دارای سه درجه آزادی در هر پا می‌باشد^[۱۵].



شکل ۱۹-۱: ربات چهارپای پیرو[۱۵]

• **ربات‌های آیدین:** ربات آیدین ۱ و ۲ و ۳(AiDIN)^۳ ساخت آزمایشگاه رباتیک هوشمند و سیستم‌های مکاترونیک^۴ در کره شمالی در سال‌های ۲۰۰۷، ۲۰۰۸ و ۲۰۱۳ می‌باشد^{[۹][۱۰][۱۱][۱۲]}. شکل ۲۰-۱ نمایی از این دو نسل ربات را نشان می‌دهد. این ربات دارای ۱۰ درجه آزادی فعال و ۴ درجه آزادی غیر فعال است. همانطور که در شکل مشاهده می‌شود، پاهای جلو هر کدام دارای ۳ درجه آزادی فعال و یک غیر فعال به صورت فنر و پاهای عقب دارای ۲ درجه آزادی فعال و یک غیر فعال به صورت فنر می‌باشد. پاهای ربات آیدین ۲ دارای وزن کمتر و نسبت ساق به ران آن مناسب‌تر انتخاب شده است و نیز مکان مرکز ثقل پاهای برای پایداری بهتر اصلاح گردیده است.



شکل ۲۰-۱: نمای ربات آیدین (۱ راست) و آیدین (۲ چپ)[۱۲]

¹PQ1-PIRO

²Pohang Institute of Intelligent Robotics

³Artificial Digitigrade for Natural Environment

⁴Intelligent Robotics and Mechatronic Systems Laboratory

هر بندگاه فعال توسط یک موتور جریان ثابت ۲۰ وات با جعبه با نسبت دور ۱:۵۳ تحریک می‌گردد. ربات آیدین دارای دو کنترل گر درونی^۱ اصلی است که به صورت کامپیوترهای تک بُرد پنتیم ۳ با فرکانس ۸۰۰ مگاهرتز با دیسک فلاش و کنترل گر CAN^۲ تحت سیستم عامل بلادرنگ RTLinux می‌باشد. کامپیوتر تک بورد دیگر که پنتیوم ۱/۱ گیگاهرتز و دارای شبکه بی‌سیم است، وظیفه پردازش صدا و تصویر را دارد. انرژی ربات توسط یک سیم برق ۲۲۰ ولت تأمین می‌گردد ولی تمام تبادل اطلاعات ربات با کاربر از طریق شبکه بی‌سیم انجام می‌پذیرد. این ربات همچنین دارای ۱۰ کنترل گر کوچک از نوع MicroChip PCI 18f458 می‌باشد که هر کدام به طور محلی یکی از درجات آزادی را کنترل می‌کند.

- **چهارپای هانما:** چهارپای هانما^۳ در سال ۲۰۱۰ در پژوهشکده رباتیک دانشگاه شاندانگ (SUCRO)^۴ چین ساخته شده که دارای ۱۲ درجه آزادی و ۵۰ کیلوگرم وزن بود که البته برق آن از خارج از ربات تأمین می‌گردد [۱۳]. ارتفاع ربات در حالت اولیه ۶۷ سانتی‌متر و طول و عرض آن به ترتیب ۱۰۰ و ۴۰ سانتی‌متر است. برای حرکت بندگاههای ربات از محرک‌های هیدرولیکی استفاده شده است و قادر است ۸۰ کیلوگرم بار را با سرعت ۴۰ سانتی‌متر بر ثانیه منتقل نماید. شکل ۲۱-۱ دو نما از این ربات چهارپا را نشان می‌دهد.



شکل ۲۱-۱: ربات چهارپای هانما [۱۳]

¹Embedded Controller

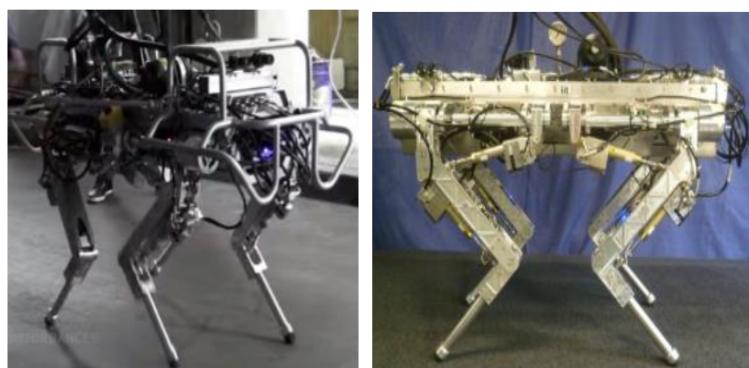
²Controller Area Network

³Hanma

⁴Shandong University Center for Robotics

- **چهارپای هایکیو:** ربات هایکیو^۱ ساخت انستیتو تکنولوژی ایتالیاست که دارای ۱۲ درجه آزادی است و موقعیت و گشتاور بندگاههای آن بهوسیله بازوهای هیدرولیکی کنترل می‌گردد [۱۸]. وزن ربات حدود ۷۵ کیلوگرم و دارای یک متر طول و نیم متر عرض است و جثه آن به اندازه یک بز می‌باشد.

سامانه کنترل این ربات بر روی یک کامپیوتر تک بورد PC104 تحت لینوکس زمان واقعی با پنج اجراء می‌گردد. این ربات می‌تواند علاوه بر قدم زدن، به صورت یورتمه نیز حرکت نماید. شکل ۲۲-۱ تصاویری از این ربات را نشان می‌دهد.



شکل ۲۲-۱: ربات چهارپای هایکیو [۱۸]

۴-۱ جمع بندی

زمینه رباتیک شامل طیف گسترده‌ای از انواع ربات‌ها است، هر کدام برای کاربردها و وظایف خاصی طراحی شده‌اند. از محیط‌های صنعتی تا بهداشت، از کاوش‌ها تا آموزش، ربات‌ها پتانسیل تحول در صنایع را دارند و به شکل‌های مختلفی به زندگی‌های ما اضافه ارزش می‌دهند. با پیشرفت فناوری، انتظار داریم که نوآوری‌های بیشتری را ببینیم و انواع جدیدی از ربات‌ها با قابلیت‌های پیشرفته‌تر پدیدار شوند.

¹HyQ

۱-۵ ساختار پایان نامه

پایان نامه شامل ۵ فصل است؛ در فصل اول، ربات و انواع آن با تاکید بر ربات‌های چهارپا، ویژگی‌ها، موارد کاربرد و نمونه‌های پیشین آورده شده است. در فصل دوم، به کمک نرم‌افزار سالیدورکس، قطعات ربات طراحی و بعد آن‌ها محاسبه شد. همچنین با بهبود نمونه اولیه سعی در رفع عیوب نموده تا به نمونه نهایی از ربات دست یافته شود. در فصل سوم، تمرکز بر سخت‌افزار ربات شده است و اجزائی مانند سرو موتور، میکروکنترلر، ماژول Arduino و دوربین ربات مورد بررسی قرار گرفته است. در فصل بعدی، پروتکل UART، تکنیک کنترلی PWM، تنظیمات لازم میکروکنترلر در نرم‌افزارهای Keil و CubeMX انجام شد. در نهایت در فصل پنجم نتیجه‌گیری و پیشنهادات لازم آورده شده است.

فصل دوم

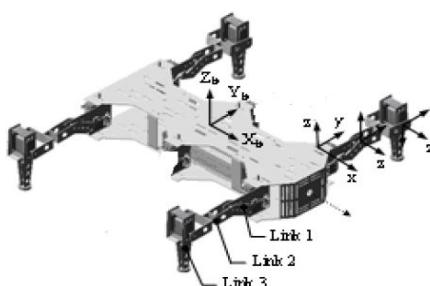
طراحی ربات

۱-۲ مقدمه

در این فصل شروع به طراحی ربات می‌کنیم که در وله اول باید طرح و مدل ربات را انتخاب کنیم یا از ابتدا مدل را بسازیم. با توجه به نمونه‌های پیشین که در فصل قبلی به آن‌ها اشاره شد، با الگو گرفتن از مدل‌های موجود، مدل ربات را طراحی و آن را می‌سازیم. در ادامه برای طراحی از نرم‌افزار سالیدورکس^۱ استفاده می‌کنیم که تمام جزئیات و نکات را می‌توان در آن اعمال کرد و درنهایت هم مدل نهایی را پرینت کرده و ربات را اسمبل می‌کنیم. در این مسیر با هربار پرینت قطعات و اسمبل ربات، تمام نکات و ایرادات آشکار شده و برای تست بعدی آن‌ها را برطرف می‌کنیم.

۲-۲ الگوبرداری و انتخاب مدل

همانطور که در قسمت بالا به آن اشاره شد؛ به دلیل وجود نمونه‌های مشابه و امکان الگوبرداری از آن‌ها، مدلی از ابتدا طراحی نشده و از مدل نمونه‌های ساخته شده، استفاده کردیم. به اصطلاح، ما مدل آزاد^۲ اقدام به طراحی ربات کردیم؛ به این معنا است که دیگر درگیر جزئیات و قیدهای مدل نشدهیم به این دلیل که هدف اصلی پروژه موضوعی دیگری می‌باشد. در این مسیر با تعدادی ربات چهارپا روبرو شدیم که با توجه به مشخصات مطلوب یعنی وزن مناسب، در حدی که موتورها قادر به بلند کردن و حرکت دادن ربات باشند، ابعاد مناسب، به شکلی که ربات قابلیت حرکت کردن در تمام محیط‌ها را داشته باشد و تنها محدود به محیط‌های باز و بزرگ نباشد، به دلیل امکان پرینت پلاستیکی از الگوهای با جنس فلزی و آهنی نمی‌توان استفاده کرد و درنهایت با توجه به محدودیت هزینه سعی بر ساده یازی مدل تا حد امکان کردیم. از چندین ربات الگو گرفتیم که همگی دارای چهارپا می‌باشند و هر پا دارای ۳ درجه آزادی^۳ است تا بتواند با آزادی عمل مناسبی به سمت هدف حرکت کند. الگو اصلی که در شکل ۱-۲ مشاهده می‌شود نزدیکترین نمونه به مشخصات مطلوب ما می‌باشد.



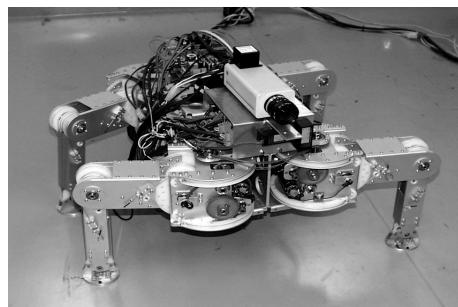
شکل ۱-۲: ربات چهارپا [۲]

¹SolidWorks

²Free-Model

³DOF(Degree Of Freedom)

یکی دیگر از نمونه‌هایی که از آن الگو گرفتیم، ربات Titan VIII می‌باشد که باز هم دارای چهارپا با سه درجه آزادی می‌باشد اما نحوه جایگیری موتورها با مدل قبلی کمی فرق دارند اما باز هم می‌تواند تمام حرکات لازم را انجام دهد^[۳].



[۳] ربات چهارپا Titan VIII

۳-۲ نرمافزار سالیدورکس [۲۰]

سالیدورکس یک نرمافزار مهندسی طراحی است که بر روی ویندوز اجرا می‌شود و توسط شرکت فرانسوی داسو سیستمز ارائه و همچنان توسعه داده می‌شود. این نرمافزار دارای سه محیط به نام‌های پارت^۱، اسembلی^۲ و دراوینگ^۳ می‌باشد. محیط اول برای رسم قطعه بوده، در محیط دوم قطعات یک مکانیسم بر روی هم سوار شده و در محیط آخر از آن‌ها نقشه مهندسی (معمولًاً برای نسخه چاپی) تهییه می‌شود. سالیدورکس یک مدل‌ساز برای مدل‌سازی جامدات است که مبتنی بر پارا سالید بوده و از رویکرد پارامتری مبتنی بر ویژگی برای ساخت مدل‌ها و مونتاژها استفاده می‌کند. پارامتر به ثابت‌هایی اطلاق می‌شود که مقدار آن‌ها، شکل یا هندسه مدل یا مونتاژ را تعیین می‌کند. پارامترها هم به صورت پارامترهای عددی نظیر طول خطوط یا قطر دایره بوده و هم به صورت قیدهای هندسی نظیر مماس، موازی، متقارب، هم مرکز و غیره هستند. پارامترهای عددی می‌توانند از طریق استفاده روابط با یکدیگر مرتبط بوده که امکان برآورده ساختن خواسته‌های طراحی را فراهم می‌کند. خواسته‌های طراحی به این معناست که طراح مایل است تا مدل نسبت به تغییرات و به روز آوری‌ها به چه صورت پاسخ دهد. مشخصات^۴ به عناصر اصلی سازنده قطعات اطلاق می‌شود. مشخصات اشکال و عملیاتی هستند که قطعه را به وجود می‌آورند. مشخصات، مبتنی بر شکل نظیر برآمدگی‌ها^۵، سوراخ‌ها^۶ و غیره معمولاً با یک نقشه دو بعدی یا سه بعدی آغاز می‌شوند.

¹Part

²Assembly

³Drawing

⁴Features

⁵Bosses

⁶Holes

نرمافزار سالیدورکس دارای ویژگی‌های خاصی می‌باشد که آن را از سایر نرمافزارهای CAD^۱ مانند کتیا^۲، پرو/اینجینیر^۳، یونیگرافیکس^۴، مکانیکال دسکتاپ^۵ و اینونتور^۶ شاخص و متمایز می‌نماید:

- افزایش سرعت و دقیقت در طراحی
- کاهش خطاهای و هزینه‌ها
- افزایش قابلیت اطمینان از محصول
- امکان طراحی پیچیده و ساختارهای مجتمع
- افزایش توانایی در حل مسائل
- سرعت بالاتر نسبت به سایر نرمافزارها
- سهولت کاربری و آموزش در مقایسه با سایر نرمافزارهای CAD
- قابلیت ارتباط با تمامی نرمافزارهای ماشین کاری (edge cam, master cam, power mill...) و نرمافزارهای تحلیل (Ansys, Adams, Abaqus, Cosmos...)
- قابلیت انجام تحلیل‌های مهندسی با اعمال پارامترها به صورت کاملاً ساده و کاربر پسند.
- امکان معادله‌نویسی (Equation) بین پارامترها و اندازه‌های مختلف در مدل

۴-۲ طراحی و محاسبه ابعاد قطعات

برای طراحی قطعات به دلیل مدل آزاد بودن، روی طراحی کامل قطعات تمرکز نشد بلکه روی جزئیات آنها و بهتر کردن آنها تمرکز کردیم. در طراحی بسیار مهم است که از چه موتوری استفاده می‌شود تا با توجه به آن، ابعاد قطعات را محاسبه کنیم و مشکلی برای جابه‌جایی هر پا و ربات نداشته باشیم. با توجه به اینکه سرموتور گشتاور ۲.۵ Kg/cm دارد، باید به این موضوع توجه شود که موتور اول هر پا باید بتواند تمام قطعات متصل به پا را یعنی دو موتور و سه لینک پا را جابه‌جا کند.

¹Computer Aided Design

²CATIA(Computer Aided Three-Dimensional Interactive Application)

³Pro/ENGINEER

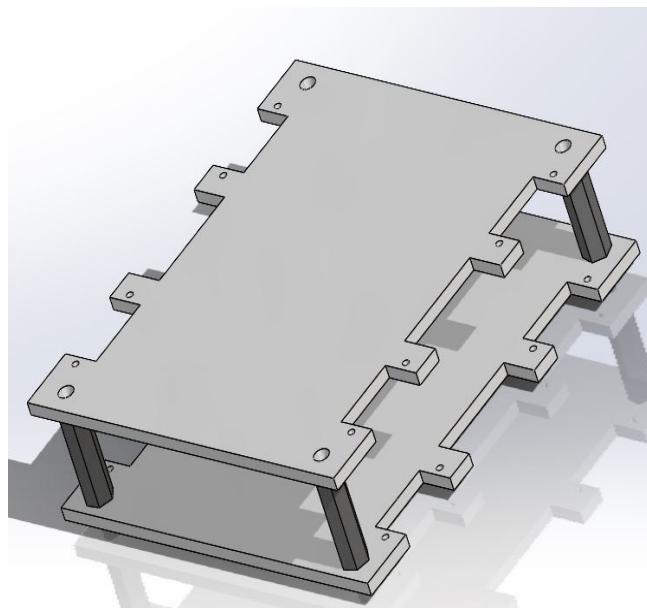
⁴Unigraphics

⁵Mechanical Desktop

⁶Inventor

۱-۴-۲ نمونه اولیه

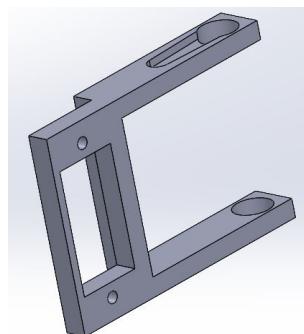
براساس همین مشخصات در ابتدا یک بدنه با ابعاد $140*120*4\text{ mm}$ طراحی کردیم که شامل دو مستطیل با همین ابعاد است که با چهارتا اسپیسر به هم متصل هستند. برای اینکه وزن موتور اول روی بقیه لینک‌ها ایجاد نشود، روی بدنه بالایی جا برای چهار موتور اول هر پا قرار دادیم که روی بدنه ثابت هستند و با استفاده از دیتاشیت^۱ ابعاد سروموتور را بدست آورده و به اندازه آن روی بدنه خالی می‌کنیم و در آخر هم جا برای پیچ‌های آن قرار میدهیم تا ربات موتور در جای خود ثابت شود و جابه‌جایی آن‌ها وابسته به بدنه است و اگر بدنه از موقعیت کنونی به نقطه‌ای دیگر برود آنگاه چهار موتور هم جابه‌جا می‌شوند. ملاحظاتی دیگر هم درنظر گرفته شده است مانند فاصله اسپیسراها از بدنه که خیلی نزدیک نباشد، چون اگر این اتفاق بیفتد خیلی شکننده شده و امکان باز شدن اسپیسراها وجود دارد. نکته دیگر هم فاصله بین دوتا موتور می‌باشد که باید به نهادی باشد که لینک اول به راحتی بچرخد و به بدنه ربات برخورد نکند. بدنه پایین هم برای قرارگیری قطعات الکتریکی ربات می‌باشد، به همین دلیل باید فاصله مناسبی بین دو بدنه باشد تا قطعات بدون مشکل قرار بگیرند و همچنین سیم ۱۲تا موتور هم باید به بردی که روی بدنه پایین قرار می‌گیرد متصل شود.



شکل ۲-۳: لایه بالایی و پایینی نمونه اولیه

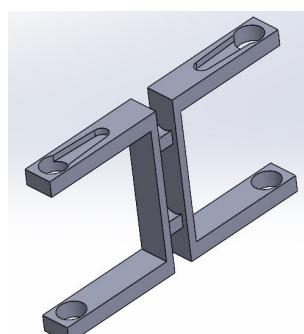
¹Datasheet

بعد از بدنه به لینک اول می‌رسیم که ابعاد $50*36*8\text{ mm}$ و تقریباً وزن ۳ گرمی دارد که برای سرو موتور هم مشکلی ایجاد نمی‌کند.(برای محاسبه وزن قطعات از نرم‌افزار Cura استفاده کردیم که می‌توان جنس موردنظر و بقیه جزئیات را در آن تنظیم کنیم) که مانند بدنه که موتور لینک اول وزنی روی لینک اضافه نمی‌کند، در اینجا هم وزن موتور دوم روی لینک اول قرار می‌گیرد و هیچ تاثیری روی لینک دوم ندارد. نکته بسیار مهمی که وجود دارد این است که روی لینک باید جایی برای قرارگیری شفت درنظر گرفته شود چونکه شفت موتور دندانه‌دار است و نمی‌توان به لینک اتصال داد که درنهایت بتواند لینک را جابه‌جا کند، پس از شفت‌های موجود در کنار موتور استفاده می‌کنیم تا این مشکل برطرف شود. باز هم مانند بدنه باید برای کوتور بعدی هم از لینک اول خالی کنیم تا در آن قرار بگیرد.



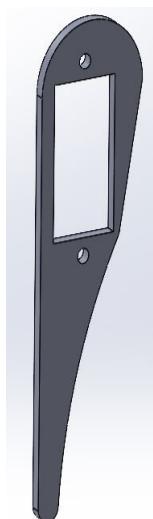
شکل ۲-۴: لینک اول نمونه اولیه

لینک بعدی هم مانند لینک قبل است فقط ابعاد و شکل آن متفاوت است. ابعاد این لینک $69*36*8\text{ mm}$ و وزن ۳ گرمی می‌باشد. تفاوت اصلی این لینک با لینک قبلی این است که، لینک دوم برای اتصال دو موتور دوم و سوم است و برای همین هم دو جا برای شفت در نظر گرفته شده است. مانند دو U است که به هم متصل شده‌اند. محور موتور اول و دوم باهم متفاوت و عمود برهم است. یکی از نکات مهم این لینک ابعاد هر U است چونکه وقتی موتور دوم جرکت می‌کند، لینک دوم جابه‌جا می‌شود و نباید ابعاد آن به شکلی باشد که به لینک اول گیر بکند یعنی عمق هر U باید از فاصله شفت موتور اول تا انتهای لینک اول بیشتر باشد.

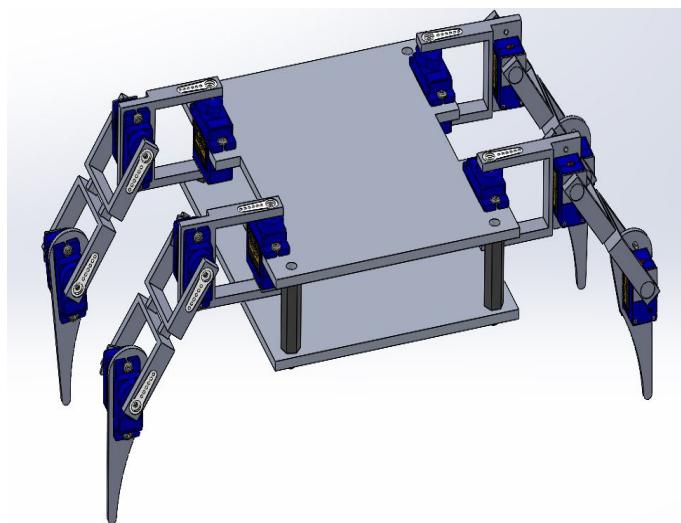


شکل ۲-۵: لینک دوم نمونه اولیه

در نهایت هم لینک آخر(سوم) را داریم که مشخصات آن $71*15*2$ mm و ۱۰۰ گرم است. چون آخرین لینک پا است دیگر نیازی به طراحی فضایی برای قرارگیری شفت نیست و تنها نیاز است فضایی برای پیچ‌های ربات تعبیه شود تا موتور ثابت در جای حود بماند و نلغزد. چالش موجود برای طراحی این قطعه در نقطه تماس آن با زمین است که باید به شکلی باشد که سطح مقطع مناسبی با زمین در تماس باشد چون اگر کمتر باشد باعث می‌شود ربات به درستی نایست و بلغزد و اگر بیشتر باشد، پاهای ربات اصطحکاک بیشتر از حد معمول خواهند داشت و به سختی جابه‌جا می‌شوند.



شکل ۲-۶: لینک سوم نمونه اولیه

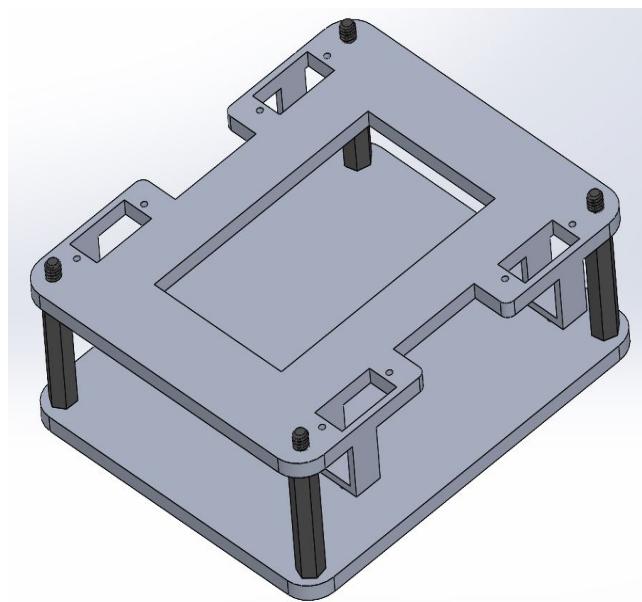


شکل ۲-۷: نسخه اولیه ربات

۲-۴-۲ بهبود مدل

در ابتدا نمونه اولیه قطعات طراحی و پرینت شدند. سپس با بررسی‌های پیاپی مشکلات و ایرادات فطعات پیدا شده و برای بهبود آن‌ها در طراحی تغییراتی ایجاد شدند.

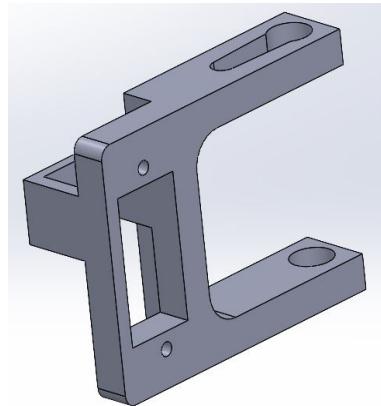
- در ابتدا به بدنه ربات می‌پردازیم؛ تقریباً ابعاد بدنه ثابت مانده است اما تغییراتی در آن اعمال شده است مانند قوسی^۱ کردن گوش‌های تیز بدنه که هم در استحکام و زیبایی آن تاثیرگذار است. دیگر تغییر مهم طراحی یک U است برای قرارگیری موتور در آن برای جایگیری دقیق موتور در مکان دلخواه و همچنین نیاز به یک قطعه مانند شفت موتور تا لینک بین این دو شفت ثابت بماند.
- و همچنین اگر سیم ۱۲ عدد موتور از پایین به برد وصل شود، ممکن است که پاهای به آن‌ها برخورد کرده و به درستی کار نکنند پس برای برطرف کردن این مشکل و همچنین کم کردن وزن بدنه، وسط بدنه بالایی را به اندازه‌ای خالی می‌کنیم تا هم سیم‌ها به راحتی و با قابلیت تفکیک عبور کنند هم اینکه قطعه محکم مانده و شکننده نشود.



شکل ۲-۸: لایه بالایی و پایینی نمونه نهایی

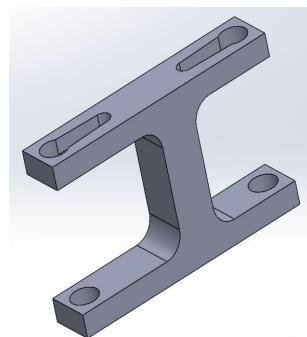
¹Curve

- اصلی‌ترین مشکلی که با آن مواجه شدیم، دقیق نبودن پرینتر هست. به عنوان مثال جایی که برای قرارگیری موتورها قرار دادیم کوچکتر از اندازه طراحی شده بود و نیاز بود کمی بزرگتر شود. دو روش برای حل این چالش وجود دارد که ابعاد موتور را خیلی بزرگتر وارد کنیم یا مقدار کمی افزایش دهیم. در حالتی که کمی افزایش دهیم شاید باز هم به راحتی موتور در مکان خود قرار نگیرد اما با کمی فشار یا تراشیدن قطعه بتوان آن را جا زد. اما اگر خیلی افزایش ابعاد را داشته باشیم امکان دارد ربات در مکان خود بلغزد.
- لینک اول بسیار نازک‌تر از قطعه مدنظر بوده است برای همین ابعاد آن تغییر کرده است. مکانی که شفت در آن قرار می‌گیر بزرگتر شده است به این دلیل که شفت فاصله کمی تا بدنه داشته و امکان آسیب دیدن و شکستن قطعه از آن ناحیه وجود داشته است و همچنین شفت به راحتی در آن قرار نمی‌گرفته و به دلیل فشار موتور کمی از جای خود بیرون زده و گاهی اوقات شفت بیرون از قطعه بوده و نتوانسته لینک را تکان دهد. در نهایت هم ماننده بدنه برای ثبات بهتر موتور روی لینک، موتور را در یک محفظه U شکل قرار داده شده است. همچنین فاصله سوراخ پیچ تا بدنه هم افزایش داده شده است تا باز هم اصطحکام بالاتر برود. آخرین نکته در این قطعه که در زیبایی و استقامت آن بسیار حائز اهمیت است، قوسی کردن گوشه‌های تیز موجود در لینک است که وظیفه بسیار مهمی در حفظ سلامت قطعه دارد؛ برای مثال قطعه اولیه اگر از بالا و پایین با دست فشرده شود به راحتی می‌شکند و همچنین قابلیت ارتعاشی دارد که اصلاً مطلوب و مناسب نیست.



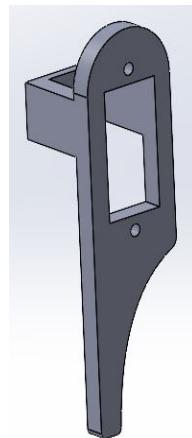
شکل ۹-۲: لینک اول نمونه نهایی

- همانند لینک اول، لینک دوم هم برای جای شفت تغییراتی کرده است تا دیگر از جای خود خارج نشده و به درستی عمل کند؛ همچنین گوشه‌های تیز هم قوسی شدند تا در استقامت لینک کمک کنند. سپس تغییری اساسی در آن مشاهده می‌شود که آن هم نحوه اتصال دو U می‌باشد که در نمونه اولیه با دو مستطیل کوچک به هم متصل شده‌اند که به شدت شکننده می‌باشد و برای برطرف کردن این مشکل، کل فاصله پر شده‌است و دیگر فضای خالی وجود ندارد. در آخر هم عمق دو تا U با هم متفاوت می‌باشد و طرفی که به سمت لینک سوم است کوچکتر از آن طرف قطعه است.

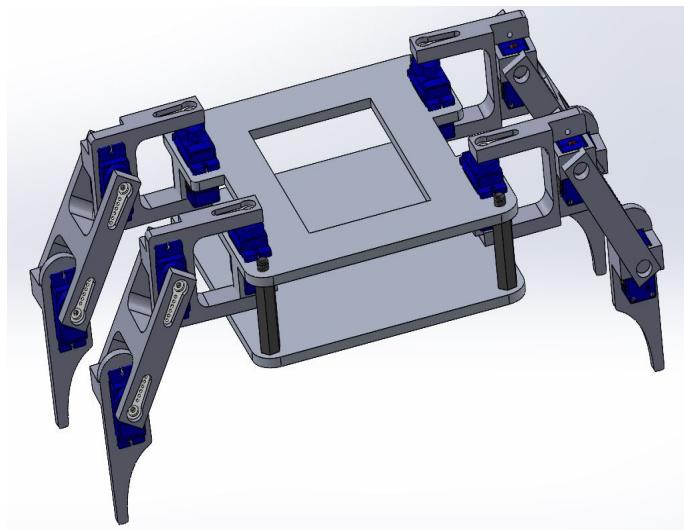


شکل ۲-۱۰: لینک دوم نمونه نهایی

- لینک آخر یا همان سوم، کمترین تغییر را نسبت به نمونه اولیه داشته است. ابعاد آن تغییر کرده است تا استحکام بیشتری داشته باشد و همچنین مانند لینک اول قسمتی برای قرارگیری موتور درنظر گرفته شده است.



شکل ۲-۱۱: لینک سوم نمونه نهایی



شکل ۱۲-۲: نسخه نهایی ربات

۳-۴-۲ نمونه نهایی

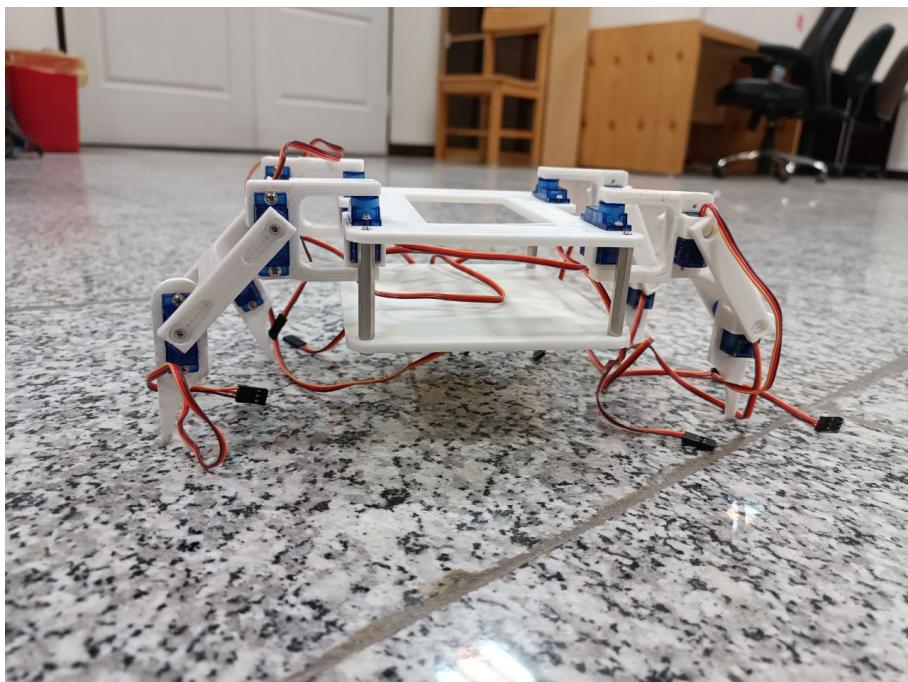
با بررسی‌های فراوان و بهبود مدل در نهایت قطعات پرینت و ربات اسمبل شده است. بدن ربات با ابعاد $140*120*4$ mm، لینک اول $140*120*4$ mm، لینک دوم $140*120*4$ mm و لینک سوم هم $140*120*4$ mm قطعی شده‌اند. بعد از چندین دفعه پرینت قطعات و تست آن‌ها به نسخه مطلوب و دلخواه رسیدیم که دیگر مشکل شفت، جای موتور، استقامت و ظاهر نامناسب ربات را نداشته باشیم.



شکل ۱۳-۲: قطعات پرینت شده ربات

۵-۲ جمع بندی

در کل، در این مسیر طراحی و ساخت ربات، با تمرکز بر جزئیات و بهبود مداوم در اجزاء و مراحل ساخت، به یک نسخه نهایی از ربات دست پیدا کردیم. با تغییر در ابعاد بدنه و لینک‌ها، حل مشکلاتی مانند جابجایی موتورها و تقویت ساختارها، به یک ربات عملی و قابل اعتماد دست یافتیم. این تجربه نشان می‌دهد که با توجه به دقیقیت در تکنیک‌های مهندسی ساده، می‌توان به اهداف خود در ساخت ربات چهارپا دست پیدا کرد؛ که با استفاده از نرم‌افزار سالیدورکس به این مهم دست پیدا کردیم.



شکل ۱۴-۲: ربات اسمنبل شده نهایی

فصل سوم

سخت افزار ربات

۱-۳ مقدمه

در این قسمت از پایان نامه، به بررسی اجزای اساسی ربات‌ها می‌پردازیم. این اجزا شامل سروموتورها، میکروکنترلرها، بردگاه‌های الکترونیکی و سایر قطعات مهم می‌شوند که در ساخت و کنترل ربات‌ها به کار می‌روند. این اجزا با تعامل و همکاری با یکدیگر، امکان حرکت، حسگری و کنترل دقیق را برای ربات‌ها فراهم می‌کنند. در ادامه به بررسی عمیق‌تر هر یک از این اجزا خواهیم پرداخت و نقش آنها در عملکرد کلی ربات‌ها را بررسی خواهیم کرد. همچنین، اهمیت انتخاب و استفاده صحیح از این اجزا را نیز بررسی می‌نماییم. این مقدمه به ما امکان می‌دهد تا به دنیای پیچیده و هیجان‌انگیز رباتیک در عمق بپردازیم و راهنمایی کنیم.

۲-۳ سرو موتور [۵]

سروموتورها از جمله اجزاء اساسی در رباتیک و بهویژه در ربات‌های چهارپا مورد استفاده قرار می‌گیرند. این اجزا جهت کنترل و حرکت اندام‌ها و پاهای ربات به کار می‌روند. سروموتورها معمولاً دارای یک شفت قابل چرخش هستند که از طریق یک سیگنال کنترلی به جلو و عقب حرکت می‌کنند. سروموتورها به دلیل دقت، سرعت و کاربردهای متعددشان، بخش مهمی از طراحی و ساخت ربات‌های چهارپا را تشکیل می‌دهند.

در ربات‌های چهارپا، سروموتورها به عنوان موتورهای اصلی برای حرکت پاهای و اندام‌ها عمل می‌کنند. این سروموتورها معمولاً دقیق و قابل کنترل با سرعت متغیر هستند، که این ویژگی‌ها امکان جابه‌جایی دقیق و پیچیده را فراهم می‌کنند. سروموتورها معمولاً با استفاده از میکروکنترلرها به راحتی کنترل می‌شوند. این اجزا به تنها یک یا به صورت گروهی در ربات‌های چهارپا استفاده می‌شوند تا حرکت و موقعیت دقیق در سیستم رباتیک تضمین شود.



شکل ۱-۳: سرو موتور SG90

۱-۲-۳ اجزای سرو موتور

یک سرو موتور از سه قسمت اصلی تشکیل شده است:

۱. **مотор:** موتور بسته به منبع تغذیه و نیازهای برنامه کاربردی، ممکن است یک موتور جریان مستقیم^۱ یا موتور جریان متناوب^۲ باشد که قدرت مکانیکی را برای چرخش یا حرکت شفت خروجی فراهم می‌کند.

۲. **حسگر:** می‌تواند یک پتانسیومتر، یک انکودر، یک رزولور یا دستگاهی دیگر باشد که موقعیت، سرعت یا گشتاور شفت خروجی را اندازه‌گیری کرده و سیگنال‌های بازخورد را به کنترل‌کننده ارسال می‌کند.

۳. **کنترل‌کننده:** می‌تواند یک مدار آنالوگ یا دیجیتال باشد که سیگنال‌های بازخورد از حسگر را با سیگنال‌های مقدار مورد نظر از یک منبع خارجی (مانند یک کامپیوتر یا یک جوی استیک^۳) مقایسه کرده و سیگنال‌های کنترلی را تولید کرده تا ولتاژ یا جریان موتور را به تناسب تنظیم کند. کنترل‌کننده از یک سیستم بازخورد حلقه بسته^۴ استفاده می‌کند تا حرکت موتور را تنظیم کرده و اطمینان حاصل کند که با مقدار مورد نظر تا حدود خطای مشخصی همخوانی دارد. کنترل‌کننده همچنین می‌تواند از انواع مختلفی از الگوریتم‌های کنترل استفاده کند، مانند کنترل PID، کنترل منطق فازی^۵، کنترل تطبیقی^۶ و غیره، به منظور بهینه‌سازی عملکرد موتور سرو.

۲-۲-۳ انواع سرو موتور

مоторهای سرو بر اساس منابع تغذیه، ساختار، مکانیزم بازخورد و کاربردهای خود به انواع مختلفی تقسیم می‌شوند.

۱-مоторهای سرو جریان متناوب (AC)

سرو مotorهای متناوب به دو نوع سنکرون و آسنکرون تقسیم می‌شوند:

- سرو مotorهای متناوب سنکرون: دارای یک روتور با آهنربای دائمی هستند که با سرعت مشابه میدان استاتور می‌چرخند. آن‌ها، دقیق‌تر و پاسخ‌گویی بهتری نسبت به مotorهای آسنکرون دارند، اما به کنترل‌کننده پیچیده‌تری و یک حسگر موقعیت نیاز دارند.

- سرو مotorهای متناوب آسنکرون: دارای یک روتور کلافی یا روتور سیم‌پیچی هستند که یک جریان القاء می‌کنند و یک میدان مغناطیسی دارند که عقب‌تر از میدان استاتور است. آن‌ها

¹DC motor

²AC motor

³Joystick

⁴Closed-Loop

⁵Fuzzy logic control

⁶Adaptive control

ساده‌تر، ارزان‌تر و مقاوم‌تر از موتورهای سنکرون هستند، اما کارایی، دقت و سرعت پایین‌تری دارند.

سرو موتورهای متناوب برای کاربردهایی با نیاز به توان بالا، سرعت و قابلیت اطمینان مناسب هستند و به طور معمول در دستگاه‌های صنعتی، رباتیک، دستگاه‌های CNC و غیره استفاده می‌شوند.

۲- موتورهای سرو جریان مستقیم (DC)

سرو موتورهای جریان مستقیم از جریان مستقیم (DC) برای تغذیه خود استفاده می‌کنند. آن‌ها دارای یک استاتور با آهنربای دائمی هستند که یک میدان مغناطیسی ثابت ایجاد می‌کند و یک روتور کلافی دارند که هنگامی که جریانی به آن داده می‌شود، می‌چرخد.

موتورهای سرو جریان مستقیم به دو نوع با جاروبک^۱ و بدون جاروبک^۲ تقسیم می‌شوند.

- سرو موتورهای جریان مستقیم با جاروبک: دارای یک کموتاتور^۳ و جاروبک‌هایی است که جهت جریان را در مارپیچ‌های روتور تغییر می‌دهند. آن‌ها ساده، ارزان و با کنترل آسان هستند، اما به دلیل اصطکاک و سایش تراشه‌ها کارایی، عمر مفید و سرعت پایین‌تری دارند.

- سرو موتورهای جریان مستقیم بدون جاروبک: دارای یک کنترل‌کننده الکترونیکی هستند که جهت جریان را در مارپیچ‌های استاتور تغییر می‌دهند. آن‌ها از لحاظ کارایی، مقاومت و سرعت، نسبت به موتورهای تراشه بهتر هستند، اما نیاز به کنترل‌کننده پیچیده‌تر و یک حسگر موقعیت دارند.

سرو موتورهای جریان مستقیم مناسب برای کاربردهایی با توان پایین که نیاز به دقت بالا، پاسخگویی و حرکت نرم دارند، هستند و به طور معمول در پروژه‌های سرگرمی، اسباب‌بازی‌های اتومبیل، پخش‌کننده‌های CD/DVD و غیره استفاده می‌شوند.

در این پروژه برای ساخت ربات چهارپا، با توجه به قابلیت‌ها و ویژگی‌ها از موتور جریان مستقیم بدون جاروبک استفاده شده است.

¹Brushed

²Brushless

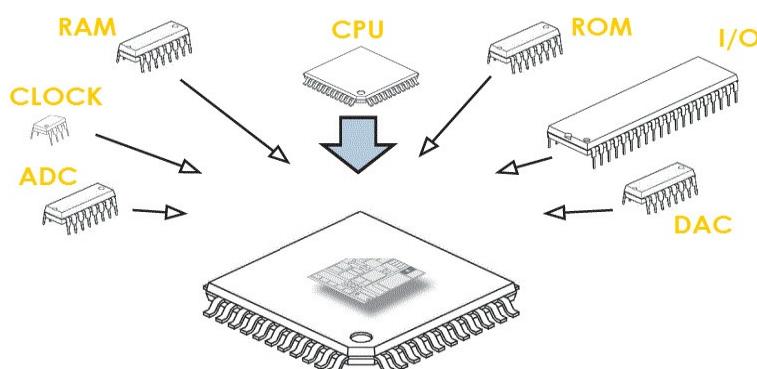
³Commutator

۳-۳ میکروکنترلر

میکروکنترلر^۱ در اصل یک چیپ الکترونیکی برنامه‌پذیر است که با اتصال قطعات مختلف در یک مدار الکترونیکی، اجزای یک کامپیوتر ساده را فراهم می‌کند. از میکروکنترلر برای ساخت، کنترل و مانیتورینگ انواع سیستم‌های الکترونیکی استفاده می‌شود که با برنامه‌ریزی واحدهای میکروکنترلر و تجهیزات جانبی فعال می‌گردد. میکروکنترلر شامل CPU، حافظه RAM/ROM، پورت‌های ورودی و خروجی (I/O)، تایмер، مبدل آنالوگ به دیجیتال^۲ و مبدل دیجیتال به آنالوگ^۳ است. CPU در واقع همان مغز میکروکنترلر است، که وظیفه‌ی استخراج و پردازش داده‌ها، انجام محاسبات و وظایف اختصاص‌داده شده را بر عهده دارد. تایمر برای تولید پالس، اندازه‌گیری فرکانس و ساخت نوسانات به کار می‌رود تا عملیات زمان‌بندی و شمارش را کنترل نماید.

برای برنامه نویسی میکروکنترلرها، نرم افزارهای خاصی وجود دارند که به آن‌ها کامپایلر^۴ می‌گویند. چند تا از کامپایلرهای محبوب کیل (Keil)، اتمل استودیو (Atmel Studio)، کدویژن (Codevision) و بسکام (Bascom) هستند. در هر کامپایلر برنامه نویسی به زبان‌های خاصی انجام می‌شود. به طور مثال در کامپایلر کیل از زبان‌های C و اسembly (Assembly) استفاده می‌شود^[۱۵].

برای استفاده مناسب و صحیح از این قطعه الکترونیکی، باید ابتدا با انواع میکروکنترلر و کاربردهای آن آشنا شد.



شکل ۳-۳: اجزای میکروکنترلر [۲۲]

^۱Microcontroller

^۲ADC

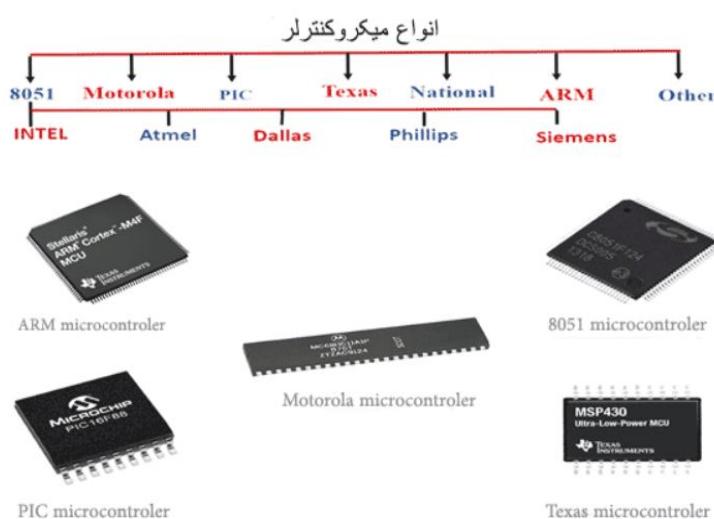
^۳DCA

^۴Compiler

۱-۳-۳ انواع میکروکنترلر

اغلب میکروکنترلرها ویژگی های مشترک زیادی دارند زیرا همهی آنها دارای یک حافظه درایو، پایه های ورودی و خروجی و توان مصرفی کم هستند. اما در جزئیاتی مانند تعداد پایه ها، ابعاد، قیمت تمام شده و غیره نیز با هم متفاوت اند. میکروکنترلرها براساس حافظه، معماری، بیت ها و مجموعه دستورالعمل ها به دسته بندی های مختلفی تقسیم می شوند. در اینجا انواع میکروکنترلرها براساس نوع کار کرد و مداری که در آن ها مورد استفاده قرار می گیرند به گروه های زیر دسته بندی می شوند:

- میکروکنترلرها AVR
- میکروکنترلرها ARM
- میکروکنترلرها XMEGA
- میکروکنترلرها PIC
- میکروکنترلرها ۸۰۵۱



شکل ۳-۳: انواع میکروکنترلر [۲۲]

۲-۳-۳ مزایا

میکروکنترلرها دارای مزایای متعدد هستند. مزیت فوق العاده میکروکنترلرها کم شدن تعداد آسی‌سی‌ها و قابلیت چند بار نوشتن و پاک کردن کد است. در ذیل چند مزیت مهم میکروکنترلرها آورده شده است:

۱. سهولت برنامه‌نویسی: میکروکنترلرها با زبان‌های برنامه‌نویسی ساده و قابل فهم برنامه‌ریزی می‌شوند.

۲. حجم کم: ابعاد کوچک میکروکنترلرها، امکان استفاده در دستگاه‌های با اندازه کوچک را فراهم می‌کند.

۳. مصرف انرژی پایین: میکروکنترلرها از انرژی کمی برای عملکرد خود استفاده می‌کنند.

۴. انعطاف‌پذیری: این دستگاه‌ها قابلیت تغییر برنامه‌های خود را دارا هستند و می‌توان آن‌ها را برای بسیاری از کاربردها برنامه‌ریزی کرد.

۵. هزینه پایین: میکروکنترلرها به صورت معمول به صورت اقتصادی قابل تهیه هستند و در اکثر موارد ارزان‌ترین گزینه برای کنترل دستگاه‌ها هستند.

۶. سرعت بالا: این دستگاه‌ها با سرعت بالایی عمل می‌کنند و گاهی نیاز به پردازش سریع دارند.

۷. قابلیت ارتباطات: میکروکنترلرها قابلیت ارتباط با سایر دستگاه‌های الکترونیکی را فراهم می‌کنند.

۸. پایداری: این دستگاه‌ها دارای پایداری و اطمینان بالایی در عملکرد خود هستند.

با توجه به این مزایا، میکروکنترلرها ابزاری قدرتمند برای کنترل و اتوماسیون در صنایع و کاربردهای الکترونیکی هستند.

۳-۳-۳ کاربرد

همان‌طور که اشاره شد، میکروکنترلر در اصل یک رایانه بسیار کوچک است که در هر پروژه و دستگاهی به طور همزمان نقش قلب و مغز مجموعه را ایفا می‌کند. بنابراین در حال حاضر لوازم خانگی و دستگاه‌های پزشکی و صنعتی مانند سیستم کنترل روشنایی، سیستم کنترل دما و آتش، سیستم‌های کنترل فرمان که در آنها اعمالی همچون اندازه‌گیری، ذخیره‌سازی، محاسبه، کنترل و نمایش اطلاعات انجام می‌شود، از میکروکنترلر استفاده شده است.



شکل ۳-۴: کاربردهای میکروکنترلر [۲۲]

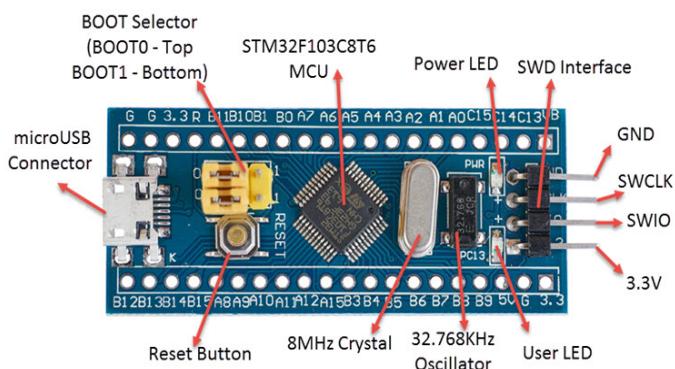
با توجه به تمام نکات در ابتدا قصد به استفاده از مژول Arduino Uno داشتیم اما به این دلیل که ما برای کنترل هر موتور به یک پایه با قابلیت PWM نیاز داریم تصمیم به استفاده از مژول STM32F103C8T6 گرفتیم که تمام خواسته‌های ما را برطرف می‌کند.

STM32F103C8T6 مژول ۴-۳

میکروکنترلر STM32F103C8T6 دارای هسته پردازنده ۳۲ بیتی ARM Cortex M3 توسط شرکت STMicroelectronics ساخته شده است. این برد شامل میکروکنترلر STM32F103C8T6، پورت RTC میکرو USB تغذیه، دکمه Reset، دو سلکتور BOOT0 و BOOT1، کریستال 8MHz، کریستال 32.768KHz، دو LED (یک LED نمایش دهنده اتصال تغذیه و دیگری متصل به پین C13 برای استفاده کاربر)، رگولاتور 3.3V و هدر برد پایه‌های SWI برای پروگرام کردن میکرو است.

مشخصات فنی مژول موردنظر:

- دارای هسته ۳۲ بیتی ARM Cortex M3
- ولتاژ کاری: ۲.۷ تا ۳.۶ ولت
- فرکانس: ماکزیمم ۷۲ مگاهرتز
- حافظه: ۶۴ کیلوبایت
- تعداد پین‌های ورودی/خروجی: ۳۷
- تعداد پین‌های PWM: ۴ + ۱۲
- تایمر: ۳+۱
- پروتکل‌های ارتباطی: I2C, SPI, UART/USART



شکل ۳-۵: مژول STM32F103C8T6

Arduino-Uno مژول ۵-۳

برد Arduino Uno، یک برد متن باز مبتنی بر میکروکنترلر Microchip ATmega328P است و توسط شرکت Arduino.cc ساخته شده است. این برد مجهز به مجموعه‌ای از پین‌های ورودی / خروجی دیجیتال و آنالوگ است که می‌تواند به راحتی با شیلد‌های آردوینو ارتباط برقرار کند. برد Uno اولین سری از سری برد‌های آردوینو مبتنی بر USB است. میکروکنترلر ATmega328 موجود در برد با یک بوت لودر از قبل برنامه ریزی شده است که اجازه می‌دهد بدون استفاده از پروگرم سخت‌افزاری خارجی برنامه‌ریزی شود. مژول آردوینو اونو در زیر آورده شده است:



شکل ۳-۶: مژول Arduino Uno

برای پروگرم کردن مژول ESP32-CAM نیاز داریم که با استفاده از پروتکل سریال این کار را انجام دهیم و برای انجام آن از FTDI استفاده می‌کنیم که رابط سریال به USB می‌باشد. اما ما به جای آن از مژول Arduino Uno استفاده می‌کنیم هم به دلیل دردسترس بودن، سادگی و توانایی مشاهده اطلاعاتی که از دوربین برای ما فرستاده می‌شود؛ مانند متصل بودن یا نبودن به روتور و IP که دوربین روی آن بالا آمده است.

ESP32-CAM مژول ۶-۳

ماژول ESP32-CAM یک مژول کامل با میکروکنترلری یکپارچه است که می‌تواند به طور مستقل کار کند. علاوه بر اتصال اینترنت و بلوتوث، دارای یک دوربین فیلمبرداری یکپارچه و یک کارت حافظه microSD برای ذخیره سازی است. این قطعه برای نظارت و شناسایی تصویر بسیار کاربردی است. مشخصات فنی این مژول به شرح زیر است:

- ارتباطات: بلوتوث BLE 4.2 به همراه WiFi 802.11b. که از بارگذاری تصویر از طریق WiFi پشتیبانی می‌کند.
- اتصالات: UART ، SPI ، I2C و دارای ۹ پایه GPIO و PWM است.

- فرکانس کاری: تا ۱۶۰ مگاهرتز
- حافظه: ۵۲۰ کیلوبایت SRAM + SD، چهار مگابایت حافظه کارت
- دوربین: از دوربین های OV2640 پشتیبانی می کند که دارای ۲ مگاپیکسل روی سنسور با اندازه آرایه 1622×1200 پیکسل هستند.



شکل ۷-۳: ماژول ESP32-CAM

۷-۳ منبع تغذیه دی سی

منبع تغذیه را می توان اینگونه تعریف کرد که یک وسیله الکتریکی است که برای تامین برق بارهای الکتریکی استفاده می شود. وظیفه اصلی این دستگاه تغییر جریان الکتریکی از منبع به ولتاژ، فرکانس و جریان دقیق برای تامین بار می باشد. منبع تغذیه دی سی^۱ منبع تغذیه ای است که ولتاژ دی سی ثابتی را برای بار خود فراهم می کند. منبع تغذیه DC که به عنوان منبع تغذیه میزی نیز شناخته می شود، نوعی منبع تغذیه است که ولتاژ جریان مستقیم (DC) را برای تغذیه یک دستگاه تامین می کند. در این پروژه، علت استفاده از منبع ولتاژ دی سی به جای باتری، کاهش وزن ربات، توان خروجی بالا و عملکرد پایدارتر است.



شکل ۸-۳: منبع ولتاژ DC

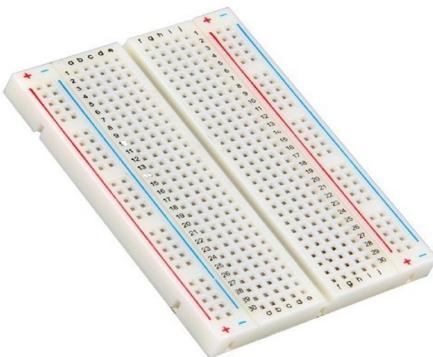
^۱Direct Current

۸-۳ برد الکترونیکی

بردهای الکترونیکی شامل Breadboard، برد مدار چاپی^۱ و برد ۱۰۰۰ سوراخ^۲ هستند. این ابزارها به عنوان پایه‌هایی برای ساخت و اتصال مدارهای الکترونیکی عمل می‌کنند و در جنبه‌های عملکردی متفاوتی دارند. در زیر توضیح مختصری از هر کدام ارائه شده است:

۱-۸-۳ Breadboard

بردبورد وسیله‌ای است که می‌توان با استفاده از آن بدون لحیم کاری نمونه‌های اولیه مدارهای الکترونیکی را طراحی و تست نمود. با قرار دادن پایه‌های قطعات الکترونیکی در سوراخ‌ها و سپس برقراری اتصال به وسیله سیم در مدارهای الکترونیکی به یکدیگر وصل می‌شوند. بردبورد نوارهای فلزی دارد که در زیر سوراخ‌های پلاستیکی قرار گرفته و آن‌ها را به هم متصل می‌کند. سوراخ‌های ردیفهای بالا و پایین به صورت افقی متصل هستند و به وسیله شیاری در وسط تقسیم می‌شوند در حالی که سوراخ‌های باقی مانده به صورت عمودی متصل هستند.



شکل ۹-۳: Breadboard

۲-۸-۳ برد PCB

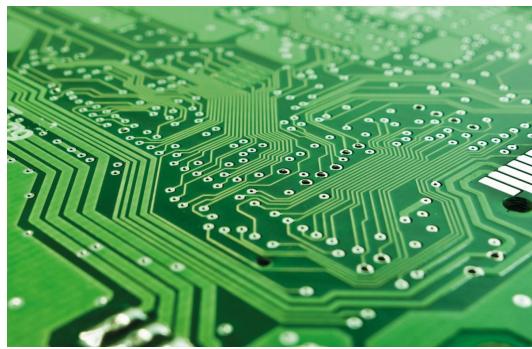
برد مدار چاپی یا PCB^۳ یک برد غیررسانا و چندلایه مسی است که تمامی قطعات الکتریکی و الکترونیکی از طریق یک برد مشترک و به وسیله تکیه‌گاه فیزیکی نصب شده در کف برد به یکدیگر متصل‌اند. قبل از رشد و گسترش بردهای مدار چاپی، مدارها به وسیله سیم‌کشی نقطه به نقطه ساخته می‌شوند که باعث افزایش پیچیدگی و عدم اطمینان در مدار می‌شود. در نتیجه هیچ‌گاه (بواسیله اتصال قطعات با سیم) قادر به ساخت مدار بزرگی نبودیم. در برد مدار چاپی تمامی قطعات بدون سیم و از داخل به یکدیگر متصل‌اند

¹PCB

²Stripboard

³Printed Circuit Board

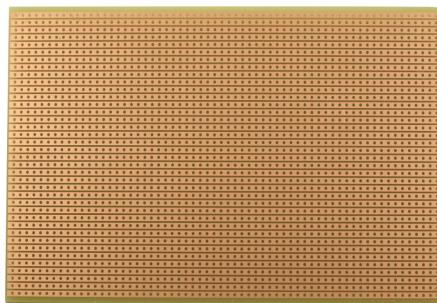
که این ویژگی منجر به کاهش پیچیدگی طرح کلی مدار می‌شود. روش طراحی PCB به گونه‌ای است که در فراهم کردن الکتریسیته و اتصال بین قطعات مدار کاربرد ویژه‌ای دارد.



شکل ۳-۱۰: یک نمونه برد مدار چاپی

۳-۸-۳ هزار سوراخ

برد هزار سوراخ^۱ گزینه دیگری است که می‌توان هنگام ساخت برد مدار برای پروژه الکترونیکی در نظر گرفت. این بردها برای نمونه‌سازی و تولید بردهای مدارهای کاربردی عالی هستند. بردهای هزار سوراخ با سوراخ‌های مشبک مستطیلی ۱۰۰ اینچی و نوارهای مسی موازی در یک طرف مشخص می‌شوند. قطعات الکترونیکی باید روی این بردۀا لحیم شوند. با توجه به سوراخ ۱۰۰ اینچی، به قطعاتی با پین‌های ۱۰۰ اینچی نیاز است. قطعات همیشه در یک طرف بورد قرار می‌گیرند، در حالی که انتهای پایه آن‌ها در انتهای دیگر بیرون می‌آید. سپس محل برخورد برد و پایه قطعات باید لحیم شوند. لحیم‌کاری باید به گونه‌ای باشد که اطمینان حاصل شود که پایه‌ها به مسیرهای مسی در طرف دیگر نوار وصل هستند. در مواردی که نیاز به اتصال سیم است، استفاده از سیم‌های ۱۰۰ اینچی توصیه می‌شود. بردهای هزار سوراخ برای ساخت مدارهای کوچک عالی هستند اما محدودیت‌هایی نیز دارند.



شکل ۱۱-۳: یک نمونه برد هزار سوراخ

در این پروژه، با توجه به بودجه مالی و امکانات، استفاده از برد هزار سوراخ بهترین گزینه ممکن است.

^۱Stripboard

۹-۳ جمع بندی

با بررسی تمام نکات و نیازهای ما برای انجام هرچه بهتر این پروژه، درنهایت بهترین انتخاب برای هر قطعه را با توجه به محدودیت مالی و زمانی انجام دادیم. نکته بسیار مهم که در آخر باید به آن اشاره شود این است که به این مسئله که آیا تمام قطعات با یکدیگر می‌توانند کار کنند و همان عملکری که از آن‌ها توقع داریم را برآورده کنند یا خیر. به عنوان مثال ما در ابتدا قصد استفاده از Arduino Uno داشتیم اما به دلیل نیاز به ۱۲ پایه با قابلیت PWM از مژول STM32F103C8T6 استفاده کردیم.

فصل چهارم

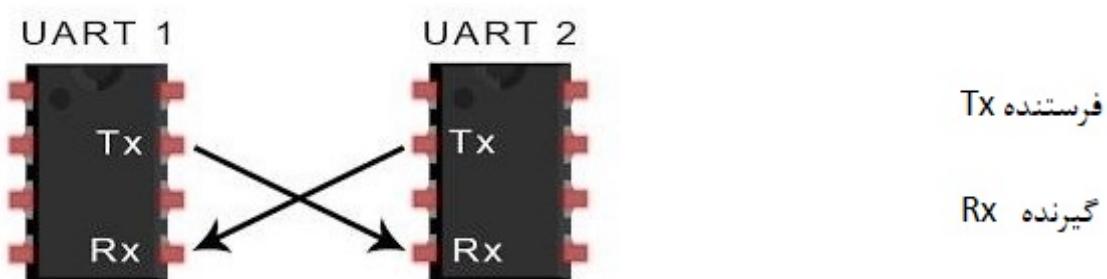
نرم‌افزار و الگوریتم‌های مورد استفاده

۱-۴ مقدمه

در این فصل بعد از طراحی و ساخت قطعات ربات و سپس اسمبل کردن آن به بحث نرم افزاری پروژه می پردازیم. در ابتدا با چند پروتکل و تکنیک آشنا می شویم که در مسیر انجام دقیق پروژه از آنها استفاده می کنیم که عبارت است از UART و PWM. به دلیل استفاده از میکروکنترلر STM32، تنظیمات اصلی و پایه های آن را در نرم افزار CubeMX انجام می دهیم و در ادامه در محیط Keil شروع به برنامه نویسی روی میکروکنترلر می کنیم. برای مرتب شدن کد سعی کردیم تا جایی که امکان دارد هر قسمت کد را جدا کرده و به صورت تابع بنویسیم. در نهایت هم با بررسی های فراوان، چهار حرکت اصلی ربات را بدست آوردهیم.

۲-۴ پروتکل UART

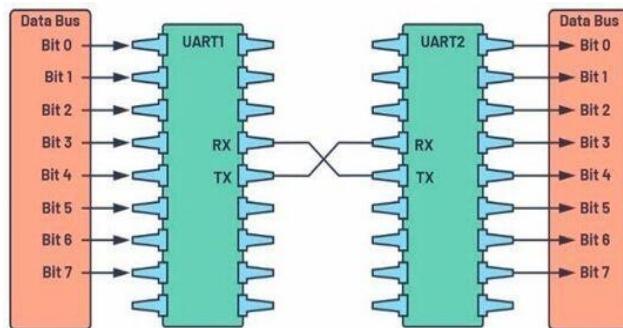
پروتکل های ارتباطی نقش مهمی را در سازماندهی ارتباطات بین دستگاه های مختلف بازی می کنند. این پروتکل ها بر اساس نیازهای هر سیستم در حالت های متفاوتی طراحی شده اند و با تعریف قوانین مشخص که در دستگاه های مختلف مشترک می باشد می توانند ضامن یک ارتباط موفق و پایدار در شبکه باشند. سیستم های جاسازی شده، میکروکنترلرها و کامپیوترها غالباً از UART به عنوان پروتکلی سخت افزاری برای برقراری ارتباط بین دستگاه های مختلف استفاده می کنند. در میان پروتکل های سخت افزاری موجود، UART برای ارسال و دریافت داده تنها از دو سیم بهره می برد. Universal Asynchronous Receiver-Transmitter مخفف UART به معنی فرستنده و گیرنده سریال ناهمزمان جهانی است. دو سیگنال در هر UART به صورت زیر نام گذاری می شود:



شکل ۴-۱: سیگنال های فرستنده گیرنده UART

بخش فرستنده UART به یک بس کنترل کننده داده ها متصل شده و اطلاعات به صورت موازی به کنترل کننده ارسال می شود، سپس داده ها به صورت سریال و بیت به بیت روی خط انتقال (سیم) برای گیرنده UART فرستاده می شوند. گیرنده UART نیز داده های سریال را پیش از انتقال به گیرنده اصلی به صورت موازی درمی آورد.

درواقع خطوط UART یک نوع واسط ارتباطی هستند که اطلاعات را از یک دستگاه می گیرند و به دستگاهی دیگر انتقال می دهند که پروتکل UART پین هایی منحصر به فرد را به انتقال یا دریافت داده اختصاص داده و یک پین نمی تواند هم خروجی و هم ورودی باشد.



شکل ۲-۴: رابط سریال UART

ما برای برقراری ارتباط بین STM32 و ESP32 از پروتکل UART استفاده می کنیم به این دلیل که ما از ۱۲ پایه که قابلیت PWM دارند استفاده کردیم و تنها پروتکل موجود برای برقراری ارتباط UART می باشد.

۳-۴ تکنیک کنترلی PWM

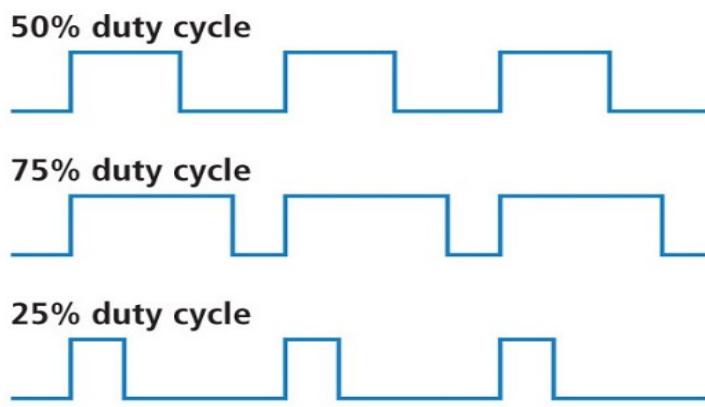
۱-۳-۴ مفهوم PWM

در بسیاری از موارد، ما نیاز به کنترل ولتاژ بر روی پایه های خروجی میکرو کنترلر را داریم. مثلاً اگر بخواهیم سرعت موتور را کنترل کنیم، باید ولتاژی که بر روی موتور اعمال می شود را کنترل کرد. در حقیقت سرعت موتور تقریباً تابع مستقیمی از ولتاژی است که بر روی آن اعمال می شود. یعنی اگر ولتاژ کاری موتوری (ولتاژ استاندارد برای فعال سازی موتور که بر روی بدنه آن نوشته می شود) ۱۲ ولت باشد، با اعمال ولتاژ ۶ ولت روی آن، می توانید سرعت چرخش آن (rpm^۱) را حدوداً به نصف کاهش دهید. مخفف PWM pulse width modulation یک فرآیند یا تکنیک مدولاسیون است که در اکثر سیستم های ارتباطی برای کد گذاری دامنه سیگنال در عرض پالس یا مدت زمان سیگنال، معمولاً سیگنال حامل، به منظور انتقال استفاده می شود.

PWM به ما این امکان را می دهد تا مدت زمان سیگنال را به صورت آنالوگ زیاد کنیم. در حالی که سیگنال در هر زمان فقط می تواند High (معمولأ ۵ ولت) یا Low (زمین) باشد، ما می توانیم نسبت زمان High بودن سیگنال را در مقایسه با زمانی که سیگنال Low است در یک فاصله زمانی ثابت تغییر دهیم.

^۱Rounds Per Minute

یکی از مفاهیم اساسی در فهم این که PWM چیست، چرخه کار یا همان duty cycle است. چرخه کار بر حسب درصد اندازه گیری می شود. چرخه کار به طور مشخص درصد زمانی را که سیگنال دیجیتال در یک بازه یا دوره زمانی روشن است، توصیف می کند. این دوره معکوس فرکانس شکل موج است. اگر سیگنال دیجیتالی نیمی از زمان را روشن و نیمی دیگر را خاموش باشد، ما می گوییم که سیگنال دیجیتال دارای چرخه کار 50% می باشد که شبیه یک موج مربعی ایده آل است. اگر درصد بیشتر از 50% باشد، سیگنال دیجیتال زمان بیشتری را در حالت بالا نسبت به حالت پایین می گذراند و برعکس اگر چرخه کار کمتر از 50% باشد، در اینجا یک نمودار وجود دارد که این سه سناریو را نشان می دهد:



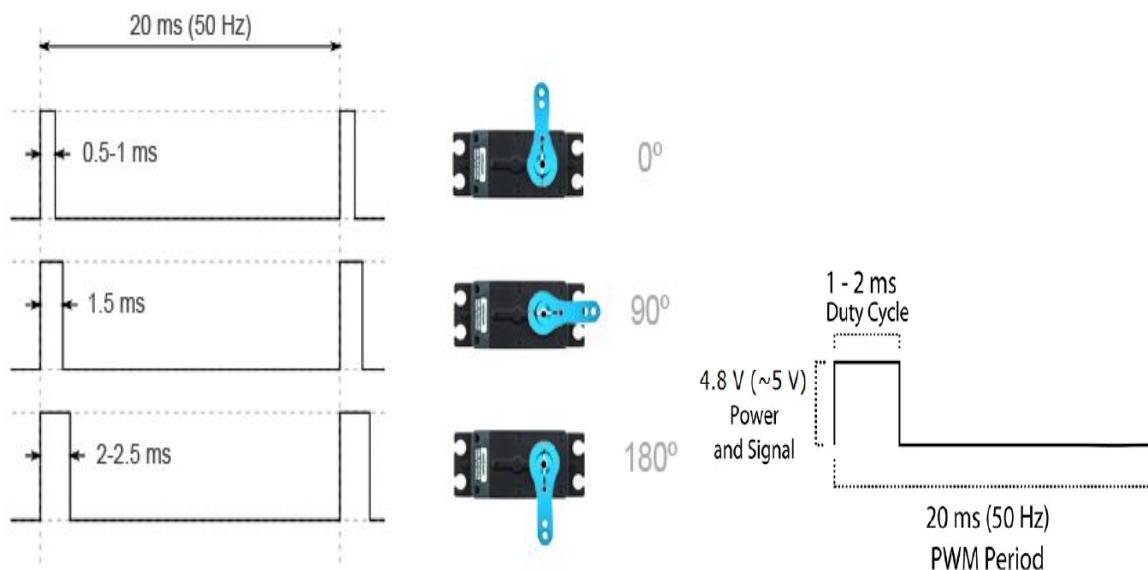
شکل ۴-۳: چرخه کار Duty Cycle

۲-۳-۴ کاربرد PWM در سروموتور

با این قابلیت می توان سرعت و یا مکان ربات را کنترل کرد. که در سروموتور با توجه به وجود برد کنترلی بر روی آن قادر هستیم زاویه چرخش شفت موتور را کنترل کنیم و با این کار می توانیم هر پا که دارای سه لینک است را به هر جهت دلخواه حرکت دهیم چون محور دو موتور موازی هستند اما محور موتور دیگر عمود بر بقیه هست و همین باعث می شود محدوده زیادی را پوشش دهد.

با توجه به دatasheet موتور SG90، این موتور دارای فرکانس 50 هرتز می باشد پس هر دوره تناوب آن برابر با 20 میلی ثانیه است. چرخه کاری این نوع موتور که در شکل ۴-۴ آمده است، بین 1 تا 2 میلی ثانیه (در بعضی مواقع 0.5 تا 2.5 ثانیه) است که باعث می شود شفت از زاویه 0° تا 180° تغییر کند. به این صورت که اگر چرخه کار کمتر از 1 میلی ثانیه یا بیشتر از 2 میلی ثانیه باشد هیچ تغییری ایجاد نمی کند.

برای شفاف شدن این موضوع باید به این مورد اشاره کنیم که به صورت خطی بین چرخه کار 1 تا 2 میلی ثانیه (در بعضی مواقع 0.5 تا 2.5 ثانیه) و زاویه 0° تا 180° رابطه ای وجود دارد. به این صورت که اگر چرخه کار 1 میلی ثانیه باشد موتور در 0° درجه خود قرار دارد و اگر 1.5 میلی ثانیه باشد در 90° درجه و در نهایت اگر در 2 میلی ثانیه باشد در 180° درجه قرار دارد (شکل ۴-۴)، پس ما به تمام زوایا دسترسی خواهیم داشت با توجه به این که تغییر هر زاویه تقریباً با 5.55 میکرو ثانیه معادل است.



شکل ۴-۴: فرکانس و دوره تناوب SG90

۴-۴ نرم افزار CubeMX

۱-۴-۴ مفهوم و کاربرد

نرم افزار STM32CubeMX که به اختصار به آن CubeMX -کیوب ام ایکس- نیز می‌گویند، به جهت ساده‌تر کردن و سرعت بخشیدن به برنامه نویسی میکروکنترلرهای STM32 ایجاد شده است. ایجاد پروژه و راهاندازی واحدهای مختلف میکروکنترلرهای STM32 به صورت گرافیکی از جمله مهم‌ترین وظایف این نرم افزار است. اگرچه کارکردهای دیگری چون تخمین میزان مصرف توان میکروکنترلر را نیز دارد.

۲-۴-۴ کتابخانه‌های برنامه نویسی

این نرم افزار، درون خود متناسب با هر میکروکنترلر شرکت ST، کتابخانه‌هایی چون HAL, LL, USB, TCP/IP و ... را دارا می‌باشد، که در صورت لزوم آن‌ها را به پروژه ما اضافه می‌کند. لایه HAL از یک سمت با سخت‌افزار و از یک سمت با سطوح بالاتر از خود که می‌تواند Middleware یا برنامه کاربر باشد ارتباط برقرار می‌کند. البته اگر بیان دقیق‌تر موردنظر باشد کتابخانه CMSIS نیز باید در نظر گرفته شود که خود رابط HAL با هسته ARM مورد استفاده در میکروکنترلر خواهد بود.

این نرم افزار علاوه بر فراهم کردن محیط تصویری و قابلیت فعال کردن و انجام اکثر تنظیمات به صورت گرافیکی این امکان را می‌دهد که کاربر یک دید کلی نسبت به میکروکنترلر خود داشته باشد و مناسب‌ترین میکروکنترلر برای کار خود را انتخاب کند.

با استفاده از این نرم افزار کتابخانه های HAL و LL به طور خودکار و بسته به انتخاب هایی که انجام شده باشد به کد اضافه می شود. همچنین مقداردهی های اولیه و بعضی از تنظیمات به صورت خودکار انجام می شود.

• **توابع Low Layer(LL):** در این روش دیگر به صورت مستقیم با رجیسترها کار نمی شود و فقط از توابعی به نام توابع LL استفاده خواهد شد که خود این توابع با توجه به آرگومان های ورودی شان، رجیسترها را مقداردهی می کنند. با فراخوانی هر کدام از این توابع عملاً بخشی از یک واحد جانبی راه اندازی می شود. برای اینکه یک واحد جانبی به طور کامل راه اندازی شود باید از یک توالی خاص از چندین توابع LL استفاده شود.

• **توابع Hardware Abstraction Layer(HAL):** در این نوع توابع که در بالاترین سطح ممکن است، با سطح رجیستر خیلی فاصله وجود دارد و عملاً به ندرت از رجیسترها استفاده می شود، اگرچه دسترسی به رجیسترها در این سطح بدون هیچ مشکلی ممکن است و با استفاده از ماکروهایی می توان به رجیسترها دسترسی داشت. وقتی که از یک تابع HAL استفاده می شود علاوه بر اینکه خودش رجیسترها یک بخش را تنظیم می کند، در اکثر موارد بسیاری از کارهایی که در توابع LL باید خود کاربر انجام می داد را نیز انجام می دهد.

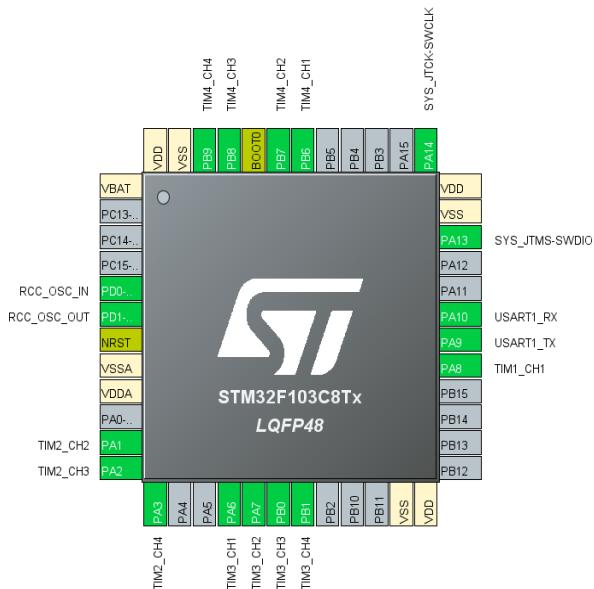
مثالاً اگر قرار بود در توابع LL، از ۵ تابع استفاده شود و کمی هم توابع زبان C به کار برده شود تا یک واحد جانبی راه اندازی شود، ممکن است تمامی این کارها با یک تابع HAL انجام شود.

• **توابع Standard peripheral libraries(SPL):** توابع SPL در سطح میانی قرار دارند و می توان گفت سطحی بین LL و HAL را دارا هستند. البته این توابع دیگر به روزرسانی نمی شوند. بنابراین امروزه به جای این توابع از توابع HAL که قابلیت های بسیار بهتری دارند، استفاده می شود.

توابع HAL به قدری همه چیز را آماده کرده اند که اگر به خوبی بررسی شوند، مشاهده می شود که در بعضی موارد اگر کاربر سهواً یا از روی ندانستن اشتباہی انجام داده باشد، آن اشتباہ را اگر ممکن باشد و تشخیص بدهد، تصحیح می کند. منطقاً هر کدام از این روش ها برای هدف خاصی مناسب هستند، و این گونه نیست که یک روش کاملاً ناکارآمد و به درد نخور باشد. به عنوان مثال اگر هدف سرعت برنامه باشد، توابع HAL اصلاً توصیه نمی شوند و رجیستری بهترین انتخاب است. در نقطه ای مقابل اگر هدف سرعت توسعه‌ی پروژه باشد و سرعت برنامه مدنظر نباشد، بهترین انتخاب توابع HAL هستند و رجیستری اصلاً توصیه نمی شود.

۴-۴-۳ کانفیگ پایه های میکروکنترلر

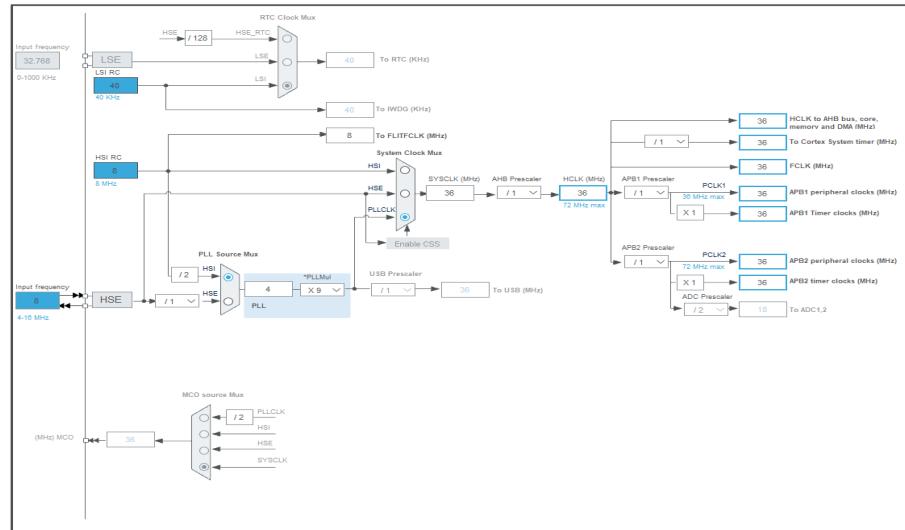
هر پین میکروکنترلر ممکن است چندین گزینه برای کانفیگ شدن داشته باشد. برای مثال اگر به دیتاشیت STMF103C8T6 مراجعه کنید. برای پین PB6 چندین کارکرد از جمله USART1-TX و Timer4-Channel1 و I2C1-SCL وجود دارد که تنها باید یکی از آنها را انتخاب نمود. ما نمی توانیم این پایه را هم به عنوان Tx واحد USART1 و هم به عنوان SCL پروتکل I2C تنظیم کنیم. چراکه این کار باعث به وجود آمدن تداخل می شود. از ویژگی های مفید CubeMX این است که در صورت امکان تداخل در کانفیگ پینهای میکروکنترلر، این موضوع را به ما نشان می دهد و مانع از بروز مشکل می شود.



شکل ۴-۵: کانفیگ میکروکنترلر

۴-۴-۴ تنظیم کلاک

به نوعی مشابه ویژگی قبلی، در قسمت تنظیم کلاک نرم افزار CubeMX امکانی وجود دارد که کمک می کند به تنظیم قسمت های مختلف کلاک پرداخته شود. در واقع در اینجا هم اگر مقداری خارج از حد مجاز به قسمتی از کلاک داده شود، با رنگ قرمز وجود مشکل را اعلام می کند. در صورت به وجود آمدن این وضعیت، به صورت اتوماتیک (توسط نرم افزار) یا دستی (توسط خودمان) مقادیر به حالت مجاز تغییر داده می شوند.



شکل ۴-۶: تنظیم کلاک

۴-۵ تنظیمات موردنیاز برای ربات چهارپا

کلاک داخلی میکروکنترلر ۸ مگاهرتز می باشد و با استفاده از کریستال خارجی می توان به ۷۲ مگاهرتز ارتقا داد، در این پروژه هم برای سرعت بالاتر از کریستال خارجی استفاده شده است و به این دلیل که به سرعت خیلی بالایی نیاز نیست ولی دقت پارامتر مهمی است پس کلاک را روی ۳۶ مگاهرتز تنظیم شده است.

در این پروژه همان طور که قبلاً به آن اشاره شده است، به ۱۲ پایه PWM نیاز است و در میکروکنترلر موردنیاز چهار تایمر وجود دارد که هر تایمر دارای ۴ کانال می باشد پس، از ۱۲ تا از این کانال ها استفاده می شود. برای درست کار کردن هر تایمر در ابتدا نیاز است که تنظیماتی اعمال شود. همانطور که در فصل قبل به آن اشاره شد، سرموتور به دامنه زمانی ۲۰ میلی ثانیه یا فرکانس ۵۰ هertz نیاز دارد. در این نرم افزار با دو پارامتر می توان به مقدار مطلوب رسید که فرمول به شکل زیر است:

$$F_{PWM} = \frac{F_{CLK}}{(ARR + 1) * (PSC + 1)} \quad (1-4)$$

پارامترهای PSC^۱ و ARR^۲ (counter period) به شکلی انتخاب می شوند تا به فرکانس ۵۰ هertz دست یافته شود. فرکانس کلاک برابر با ۳۶ مگاهرتز است پس مخرج که ضرب دوتا پارامتر می باشد باید برابر با ۷۲۰۰۰ شود؛ حال می توان هر مقداری برای این دو پارامتر در نظر گرفت ولی با توجه به چرخه کار مدنظر برای PSC مقداری خاص انتخاب می شود.

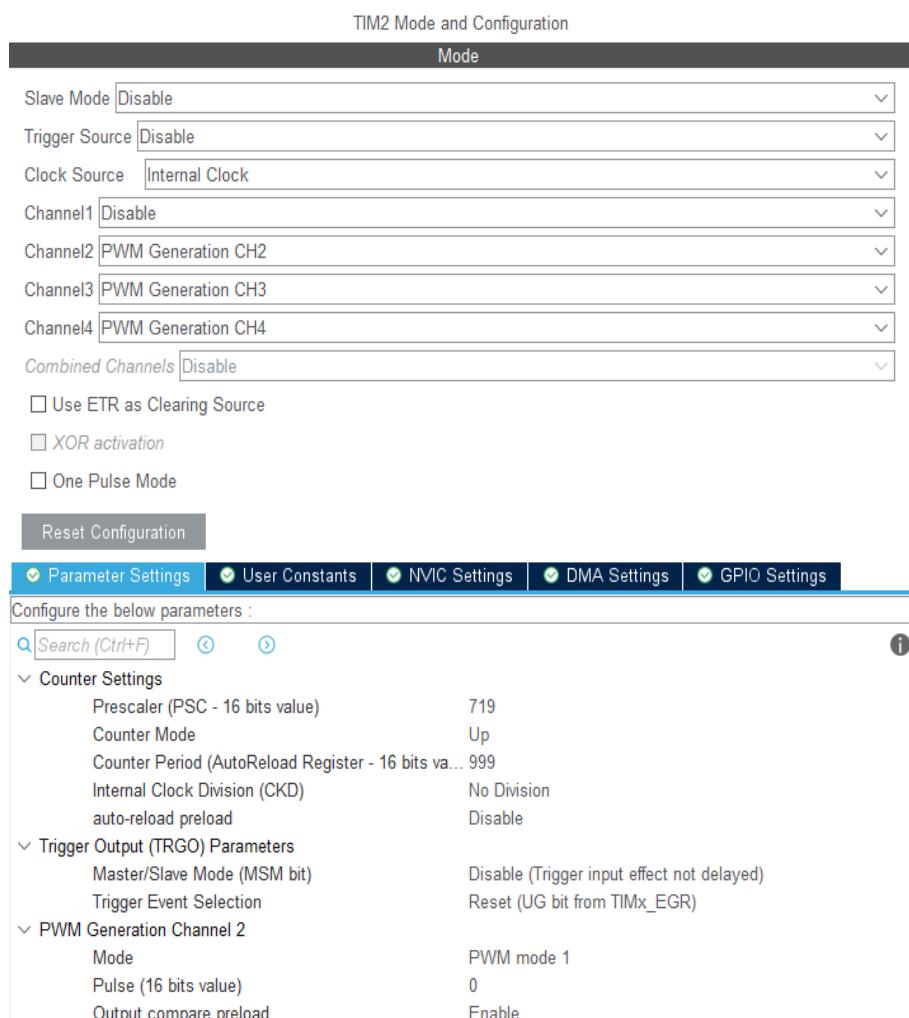
$$DutyCycle = \frac{CCR}{ARR} \quad (2-4)$$

^۱Prescale

^۲Auto Reload Register

نیاز است ابتدا و انتهای بازه برای مقداردهی PWM بدست آورده شود که مقداردهی آن به صورت رجیستری است:

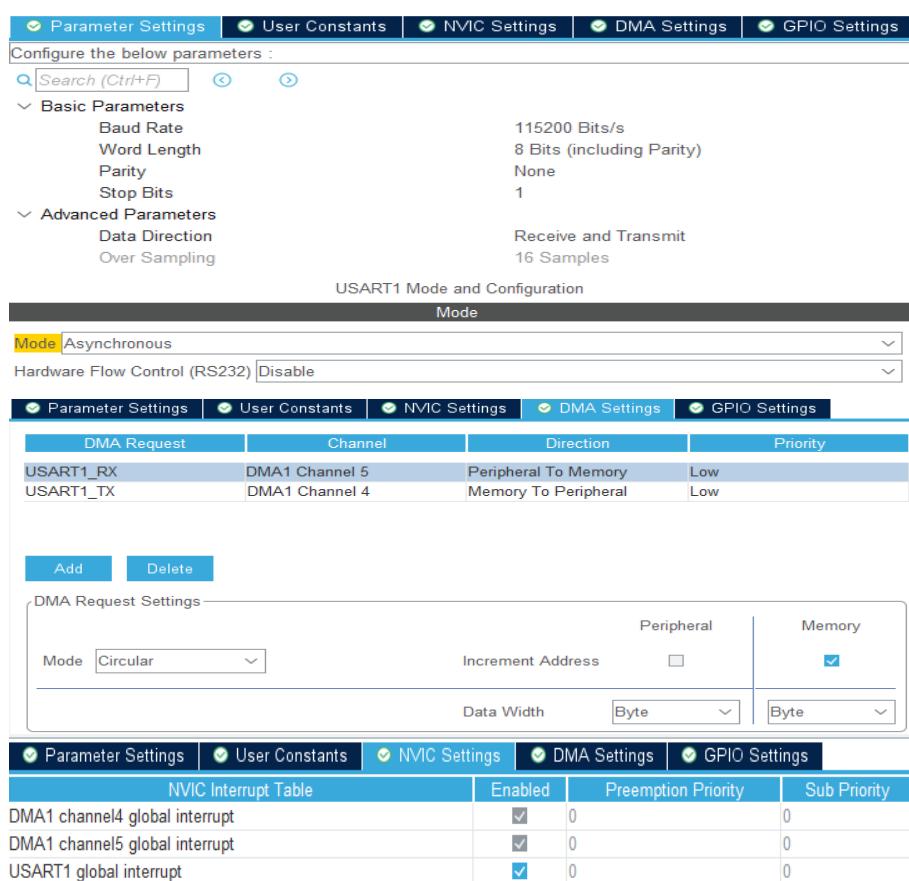
- زاویه ۰ درجه: همانطور که گفته شد بین بازه ۰.۵ تا ۱ میلی ثانیه سیگنال روشن است پس نسبت چرخه کار ۲.۵ درصد می باشد و برای راحتی در محاسبات، PSC ۹۹۹ در نظر گرفته می شود، درنهایت مقدار CCR ۲۵ بدست می آید.
- زاویه ۹۰ درجه: برای میانه بازه، نسبت چرخه کار ۱.۵ به ۲۰ می باشد که برابر با ۵.۷ درصد است و همانطور که PSC ۹۹۹ قرار داده شده است پس CCR باید ۷۵ مقداری دهی شود.
- زاویه ۱۸۰ درجه: در نهایت، برای بیشترین زاویه که ۱۸۰ درجه می باشد، سیگنال به اندازه ۲ تا ۲.۵ میلی ثانیه روشن می ماند پس نسبت چرخه کار برابر با ۱۲.۵ می شود و مقدار انتهایی CCR هم ۱۲۵ می باشد.



شکل ۷-۴: تنظیمات تایمر

برای تنظیمات پروتکل UART هم در ابتدا باید Baud Rate یکسانی برای دو پردازنده انتخاب شود تا با سرعت یکسانی کار کنند که در این پروژه بر روی ۱۱۵۲۰۰ بیت بر ثانیه تنظیم شده است. سپس در قسمت انتخاب مد، گزینه آسنکرون انتخاب شده است به این دلیل که به همزمانی نیاز نیست و همچنین حجم داده ارسالی خیلی زیادی وجود ندارد و سرعت انتقال داده این مد مناسب است و به سرعت بالاتری که منجر به استفاده از ارتباط سنکرون شود نیازی نیست.

دو نکته دیگر در کار با ارتباط سریال خیلی مهم هستند؛ زمانی که دوربین مانع را تشخیص دهد از طریق سریال به پردازنده اصلی داده ای ارسال می شود که کدام نوع مانع دیده شده است، چالشی که با آن مواجه شدیم این بود که ما نیاز به ارسال بلادرنگ^۱ داده است اما در سمت دریافت، داده فقط یکبار دریافت می شود پس ما مد گیرنده را روی Circular قرار می دهیم که در هر لحظه داده را بگیرد. در نهایت هم برای بهینه شدن کد و عملکرد آن، که در هر لحظه قصد خواندن داده نداشته باشد از وقهه^۲ استفاده شده است که باعث می شود همان لحظه که داده فرستاده می شود، داده را خوانده و آن را ذخیره کند.



شکل ۴-۸: تنظیمات ارتباط سریال

¹Real Time

²Interrupt

۵-۴ برنامه نویسی در نرم افزار Keil

۱-۵-۴ نرم افزار Keil چیست؟

نرم افزار Keil یا MDK (Microcontroller Development Kit) که ساخت شرکت Keil است، برای توسعه طیف وسیعی از میکروکنترلرهای ARM Cortex-M و میکروکنترلرهای دیگر است. این نرم افزار شامل یک IDE است که μVision نام دارد. همچنین شامل محیط ویرایشگر کد، پروگرامر و دیباگر، شبیه ساز، کامپایلر و ... است و این امکان را به کاربر می دهد که به راحتی بتواند برای میکروکنترلرها برنامه نویسی کند.

در ابتدا تنظیمات موردنیاز را در Cube MX انجام می دهیم و برای ما در Keil کد خام تولید می کند تا دیگر نیاز نباشد تمام کد را از ابتدا بنویسیم.

۲-۵-۴ تنظیم تایمرها

ما در ربات ۱۲ عدد موتور داریم که هر موتور با PWM کار می کند پس نیاز به ۱۲ پایه تایмер داریم. همانطور که در قسمتهای قبل به آن شاره شد در این میکروکنترلر ۴ عدد تایمر داریم که هر تایمر هم ۴ عدد کanal دارد پس در کل ۱۶ پایه با این قابلیت موجود است. به این دلیل که هر پایه در این میکروکنترلر چندین قابلیت دارد باید حتماً به آن دستور داده شود که از کدام قابلیت می خواهیم استفاده کنیم تا تصادفاً مشکلی پیش نیاید. هیچ تفاوتی وجود ندارد که از کدام ۱۶ پایه استفاده کنیم؛ فقط دو تا از پایه ها برای ارتباط سریال استفاده می شوند پس انتخاب از ۱۴ پایه دیگر است.

```
//Initialize all configured peripherals
MX_TIM1_Init();
MX_TIM2_Init();
MX_TIM3_Init();
MX_TIM4_Init();

//For leg1
HAL_TIM_PWM_Start(&htim1, TIM_CHANNEL_1); //Pin A8 => Motor Number 3
HAL_TIM_PWM_Start(&htim3, TIM_CHANNEL_4); //Pin B1 => Motor Number 2
HAL_TIM_PWM_Start(&htim4, TIM_CHANNEL_4); //Pin B9 => Motor Number 1

//For leg2
HAL_TIM_PWM_Start(&htim2, TIM_CHANNEL_2); //Pin A1 => Motor Number 3
HAL_TIM_PWM_Start(&htim2, TIM_CHANNEL_3); //Pin A2 => Motor Number 2
HAL_TIM_PWM_Start(&htim2, TIM_CHANNEL_4); //Pin A3 => Motor Number 1

//For leg3
```

```

HAL_TIM_PWM_Start(&htim3, TIM_CHANNEL_1); //Pin A6 => Motor Number 1
HAL_TIM_PWM_Start(&htim3, TIM_CHANNEL_2); //Pin A7 => Motor Number 2
HAL_TIM_PWM_Start(&htim3, TIM_CHANNEL_3); //Pin B0 => Motor Number 3
//For leg4

HAL_TIM_PWM_Start(&htim4, TIM_CHANNEL_1); //Pin B6 => Motor Number 1
HAL_TIM_PWM_Start(&htim4, TIM_CHANNEL_2); //Pin B7 => Motor Number 2
HAL_TIM_PWM_Start(&htim4, TIM_CHANNEL_3); //Pin B8 => Motor Number 3

```

۳-۵-۴ تابع حرکت موتور

در مرحله بعدی که تایمر را فعالسازی کردیم، توابع حرکت موتورها را می‌نویسیم. برای حرکت دادن هر موتور نیاز داریم که به صورت رجیستری به آن دستور بدھیم و همانطور که قبلاً به آن اشاره کردیم، مجاز هستیم در بازه ۲۵ تا ۱۲۵ فرمان بدھیم. برای انجام اینکار نیاز داریم که شماره تایمر، کانال و مقدار PWM را بدانیم. نکته مهم آن است که ما مقدار در لحظه هر موتور را در یک آرایه ذخیره می‌کنیم و این تابع دستور می‌دهیم که از زاویه قبلی به زاویه جدید و دلخواه برود.

```

//Moving the motor to a desired angle.

void Move(htim, timer, CCR, pwm, motor)

{
    Store the instantaneous value of the motor in variable (n)
    if(Is the previous value less than the new value or not)
    {
        switch(CCR)
        {
            case 1:
                for(;n <= pwm;n++)
                {
                    htim.Instance->CCR1 = n;
                    HAL_Delay(10);
                }
                break;
            case 2:
                for(;n <= pwm;n++)

```

```

{
    htim.Instance->CCR2 = n;
    HAL_Delay(10);
}

break;

case 3:
for(;n <= pwm;n++)
{
    htim.Instance->CCR3 = n;
    HAL_Delay(10);
}
break;

case 4:
for(;n <= pwm;n++)
{
    htim.Instance->CCR4 = n;
    HAL_Delay(10);
}
break;
}

}

Store the new value
}

```

نوعی دیگر از حرکت که برای موتورها درنظر گرفتیم، حرکت چهارپا همزمان است. به این صورت که چهارپای اول، دوم یا سوم به صورت همزمان حرکت می کنند که برای بلند شدن ربات استفاده می شود چون اگر هر پا مجزا حرکت بکنند ربات تعادل نخواهد داشت و ربات به یک سمت می افتد.

```

//Simultaneously moving all four motors in a row.

void Move_All(pwm_1, pwm_2, pwm_3, pwm_4, motor)
{
    int n, max;
    max = Find_max(pwm_1, pwm_2, pwm_3, pwm_4);
    switch(motor)

```

```
{
    case 1:
        n = 50;
        for(;n <= max;n++)
        {
            if(pwm_3 >= n)
                htim3.Instance->CCR1 = n;
            if(pwm_1 >= n)
                htim4.Instance->CCR4 = n;
            if(pwm_4 >= n)
                htim4.Instance->CCR1 = n;
            if(pwm_2 >= n)
                htim2.Instance->CCR4 = n;
            HAL_Delay(10);
        }
        break;
    }
}
```

۴-۵-۴ چهار حرکت اصلی ربات

بعد از آنکه تمام حرکت ها را نوشتهیم و بدست آوردهیم شروع به نوشتمن چهار حرکت اصلی می کنیم اما قبل از آن باید دستوری به ربات بدھیم که در ابتدا که شروع به حرکت می کند، به یک نقطه یا حالت اولیه برود که تعادل داشته باشد و به راحتی بدون لغزش بایستد.

در این تابع ابتدا به چهار موتور اول هر پا سپس چهار موتور دوم و در انتهای هم به چهار موتور سوم دستور می دهیم که به زاویه مطلوب برود. به اینصورت عمل می کند که تمام چهار موتور با یکدیگر حرکت می کنند و برای اینکار تابع Move All را فراخوانی می کنیم.

```
//Initial state of the robot in complete stability.

void Home()
{
    Delay 2 seconds
    Move_All(79, 75, 77, 78, 1);
```

```

    Move_All(125, 116, 117, 125, 2);

    Move_All(94, 86, 90, 85, 3);

    Delay 1 second

}

```

حرکت بعدی که آن را پیاده سازی کردیم، حرکت به سمت جلو می باشد که ربات با کمک اصطکاک و لغزش به سمت جلو حرکت می کند. در این حرکت ابتدا یکی از دو پای جلو باز شده تا مرکز ثقل به آن سمت برود و در این حالت پای مقابل آن در عقب ربات می تواند از زمین بلند شود و به سمت جلو برود و سپس پای جلو ربات که باز شده بود به حالت قبلی بر می گردد اما پای عقب جلو تراز حالت اولیه و صفر خود است پس وقتی به حالت قبلی بر می گردد، ربات به کمک اصطکاک پا به جلو حرکت می کند.

```

//Moving forward by the desired amount.

void Forward(int count)

{
    for(How many forward you want)

    {
        One of the front two legs opens.

        Then the robot tilts towards that leg.

        The opposite leg at the back of the robot has lifted off the
            ground and is moving forward.

        Then the front leg of the robot returns to its initial position.

        Afterwards, the rear leg, following contact with the ground, moves
            backward, causing the robot to move forward

        Delay 0.3 seconds
    }
}

```

حرکت به سمت عقب رفتن هم به همین شکل است فقط پاهای جلو و عقب عکس جلو رفتن عمل می کنند. ابتدا پای عقب باز شده و سپس پای مقابل آن در جلو ربات ابتدا از زمین بلند شده و به سمت عقب حرکت می کند؛ سپس پای عقب به حالت اولیه باز می گردد و پای جلو که به زمین رسیده است به سمت جلو حرکت می کند و با اصطکاک ایجاد شده ربات به سمت عقب می رود.

```

//Moving backward by the desired amount.

void Backward(int count)

{
    for(How many backward you want)

```

```
{
    One of the back two legs opens.

    Then the robot tilts towards that leg.

    The opposite leg at the front of the robot has lifted off the
        ground and is moving backward.

    Then the back leg of the robot returns to its initial position.

    Afterwards, the rear leg, following contact with the ground, moves
        forward, causing the robot to move backward

    Delay 0.3 seconds
}

}
```

در نهایت هم چرخش به سمت چپ و راست را داریم که به این صورت است که برای چرخش به چپ پای عقب راست باز شده تا پای جلو چپ از زمین جدا شده به سمت عقب حرکت کند و سپس بسته شده تا پای جلو چپ به زمین برسد؛ بعد از آن همان پا بسته می شود تا پای عقب راست از زمین جدا شده و جلو برود. در نهایت با برگشتن پای جلو چپ به حالت قبلی، دو پا همزمان به حالت اولیه حرکت می کنند و ربات به سمت چپ می چرخد.

```
/*Turning left by the desired amount.*/

void Turn_Left(int count)
{
    for(How many times should it turn left?)

    {
        The rear right leg has opened.

        The front left leg has lifted off the ground.

        It moves backward.

        The same leg has closed.

        The rear right leg moves forward.

        Then these two legs simultaneously return to their initial
            position.

    }
}
```

برای چرخش به راست هم به همین شکل فقط به جای پای عقب راست و جلو چپ، پای عقب چپ و جلو راست را داریم.

```
/*Turning right by the desired amount.*/

void Turn_Right(int count)
{
    for(How many times should it turn right?)

    {
        The rear left leg has opened.

        The front right leg has lifted off the ground.

        It moves backward.

        The same leg has closed.

        The rear left leg moves forward.

        Then these two legs simultaneously return to their initial
        position.

    }
}
```

۴-۵-۵ عملکرد کلی ربات

با همه نکاتی که گفتیم و توابعی که نوشتهیم باید از آنها استفاده کنیم به شکلی که ربات به سمت هدف حرکت کرده و از موانع دور شود. پس با استفاده از دوربین و تشخیص رنگ یک کاراکتر از طریق ارتباط سریال دریافت می کنیم و نسبت به رنگی که دارد دستور می دهیم که چه کاری انجام شود. ما کنار کاراکتر رنگ یک عدد سه رقمی از مساحت مانع هم می گیریم که هر چه به مانع نزدیکتر شویم این عدد بزرگتر شده پس می دانیم کی باید حرکت دیگری انجام دهیم که برخورد نداشته باشیم.

```
switch(Which color detected?)

{
    if(The obstacle is Pink)

        if(The area of the obstacle has exceeded the limit or not?)

            Forward(1);

        else

    {
        Turn_Right(5);
}
```

```
    Forward(11);

    Turn_Left(5);

    Forward(11);

    Turn_Right(5);

}

if(The obstacle is Blue)

    if(The area of the obstacle has exceeded the limit or not?)

        Forward(1);

    else

        Forward(20);

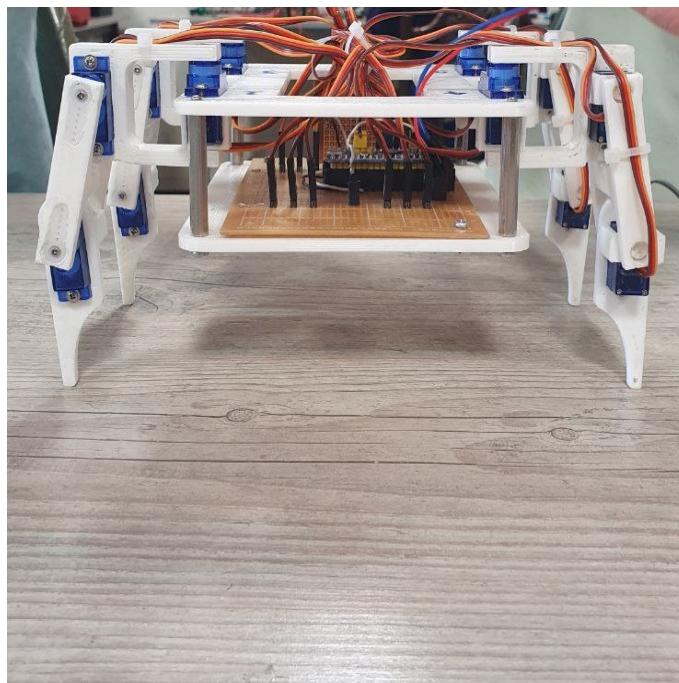
}
```

فصل پنجم

نتیجه‌گیری و پیشنهادات

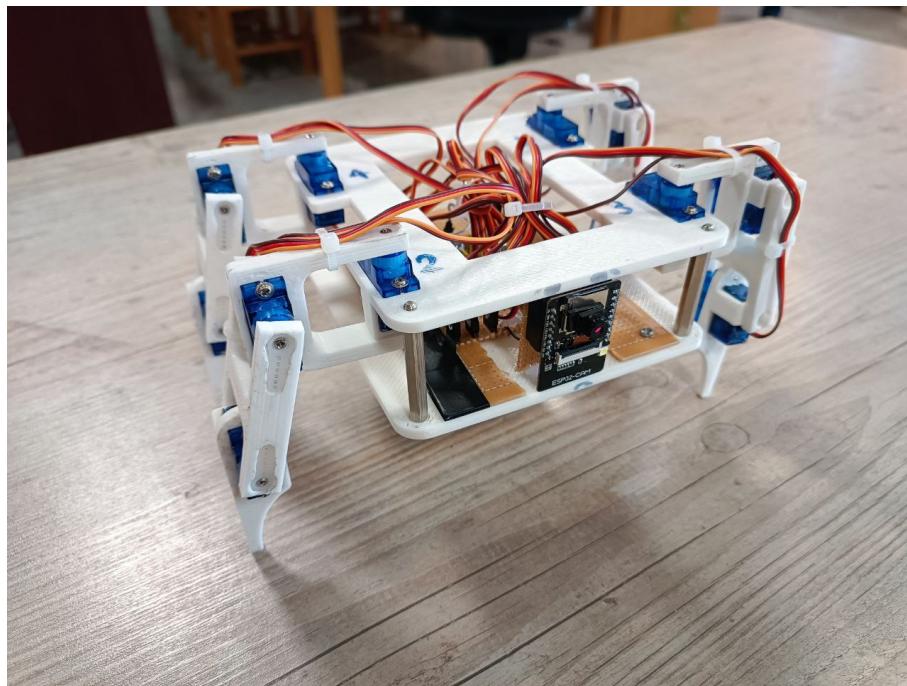
۱-۵ نتیجه‌گیری

در فصول پیشین با توجه به مزیت‌های ذکر شده برای ربات چهارپا و مشخصات مطلوب مدنظر، مدل نهایی طراحی شد. با انجام اصلاحاتی روی ربات، قطعات جدید پرینت شد و مدل نهایی در عمل پیاده سازی شد. در قسمت سخت‌افزار ربات، با توجه اینکه سرورموتور SG90 نیاز ما از لحاظ سبکی و گشتاور لازم برای بلند کردن و حرکت ربات و مقرون به صرفه بودن بر طرف نمود از آن استفاده شد، همچنین کم بودن پایه‌های Arduino Uno برای استفاده از PWM منجر به استفاده از میکروکنترلر STM32F103C8T6-ESP32 شد. برای تشخیص موانع موجود به دلیل قیمت مناسب، موجود در بازار و کیفیت مطلوب مژول CAM به کار برد شد. به این دلیل که وزنی به ربات اضافه نشود به جای آنکه باتری بر روی ربات قرار بگیرد و موتورها را برای حرکت دادن ربات دچار مشکل کند، از منبع ولتاژ DC استفاده شد و همچنین ۱۲ موتور در بعضی مواقع مانند بلند شدن ربات از زمین، جریانی زیادی را می‌کشند که ممکن است باتری نتواند آن را تامین کند.



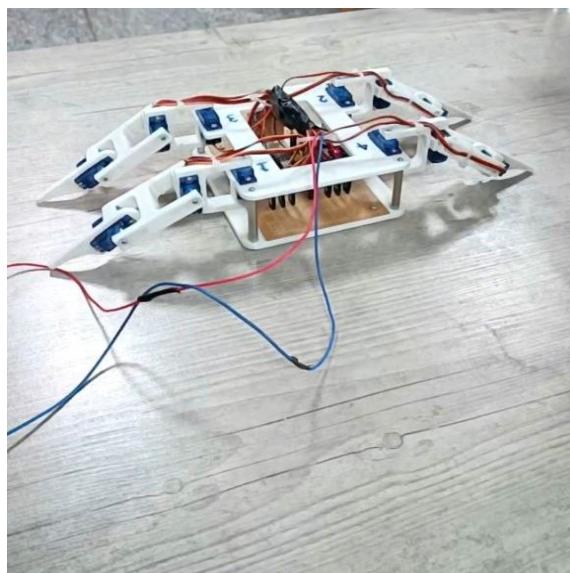
شکل ۱-۵: ربات نهایی

برای قسمت نرمافزار ربات، برای برقراری ارتباط بین میکروکنترلر و دوربین، از پروتکل UART استفاده شد. برای حفظ تعادل و حرکت در مسیر درست، نیاز است که هر موتور دقیقاً به موقعیت مطلوب برود پس برای کنترل موقعیت و زاویه هر موتور تکنیک کنترلی PWM به کار بردشده است. برای انجام تنظیمات مورد نیاز میکروکنترلر و به جهت ساده‌تر کردن و سرعت بخشیدن به برنامه‌نویسی میکروکنترلر از نرمافزار CubeMX بهره برده شد. درنهایت به دلیل رایج بودن محیط برنامه‌نویسی Keil، برای فعال‌سازی و راهاندازی تایمراها و نوشتن توابع حرکتی ربات از آن استفاده شد.

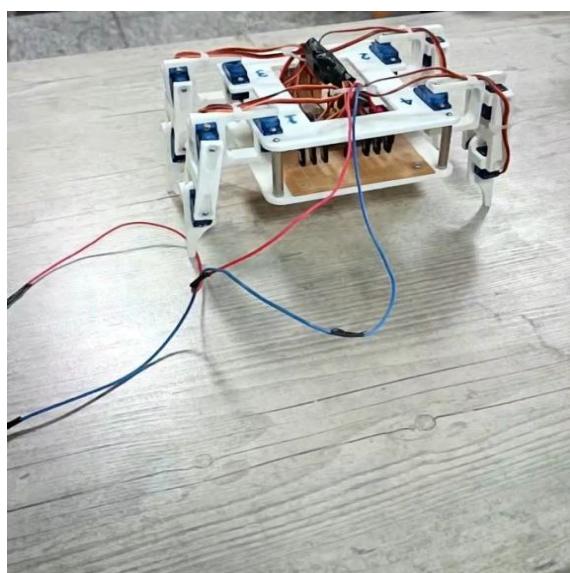


شکل ۵-۲: ربات نهایی

برای جابه‌جایی ربات از نقطه‌ای به نقطه دیگر از اصطکاک پاها کمک گرفته شده است و در تمام حرکت‌های ربات اعم از حرکت رو به جلو و عقب، به موتورها فرمانی داده شده است که با استفاده از اصطکاک و عکس العمل آن ربات به سمت دلخواه حرکت کند. همچنین برای بلند شدن ربات و قرارگیری در حالت اولیه، به موتورهای اول، دوم یا سوم هر پا همزمان فرمان داده می‌شود تا ربات در هنگام بلند شدن تعادل خود را نگه دارد و بدون لعزم بایستد. در شکل ۳-۵ ربات بدون هیچ دستور و ولتاژی به موتورهایش روی زمین قرار گرفته است اما با اعمال ولتاژ به موتورها و میکروکنترلر، دستور رفتن به حالت اولیه (۴-۵) اتفاق می‌افتد.



شکل ۳-۵: ربات بدون هیچ فرمان



شکل ۴-۵: ربات در حالت اولیه

۲-۵ پیشنهادات

با توجه به تجربیاتی که از ساخت ربات بدست آورده‌یم، بهتر است از سروموتورهای قوی‌تر و بهتری استفاده شود که در حرکت دادن پاهای ربات مشکلی وجود نداشته باشد و همچنین بهتر است پای دوم کوتاه‌تر از پای سوم باشد و به شکلی ربات قرار بگیرد که در حالت اولیه ربات که ربات ایستاده است؛ بدنه از مفصل پای سوم پایینتر باشد و به زمین نزدیک باشد. مهمترین نکته هم آن است که چرخ‌دنده‌های موتورها پلاستیکی نباشد چون با کوچکترین فشاری از خارج به ربات باعث شکستن چرخ‌دنده می‌شود.

اگر از سروموتور قوی‌تری استفاده شود، می‌توان بدن را کمی سبکتر ساخت و دیگر نیازی به منبع ولتاژ ثابت نباشد و به جای آن از باتری استفاده گردد که جابه‌جایی ربات آسان‌تر انجام شود چون وجود دو سیم منبع ولتاژ می‌تواند مانع انجام درست حرکت‌ها شود (فقط به این توجه شود که باتری توان جریان کشی زیاد موتورها را داشته باشد).

به جای حرکت به کمک اصطکاک بهتر است که طراحی به شکلی صورت گیرد که ربات قادر به گامبرداری باشد به صورتی که اگر یک پا از زمین جدا شود و به سمتی حرکت کند ربات تعادل داشته باشد و واژگون نشود. به این ترتیب ربات وابستگی به جنس محیط برای داشتن اصطکاک مطلوب ندارد.

منابع و مراجع

- [1] Ackerman, Evan. Whoa: Boston dynamics announces new wildcat quadruped robot.
<https://spectrum.ieee.org/whoa-boston-dynamics-announces-new-wildcat-quadruped>,
2013.
- [2] Agarwal, Saurav, Mahapatra, Abhijit, and Roy, Shibendu Shekhar. Dynamics and optimal feet force distributions of a realistic four-legged robot. *International Journal of Robotics and Automation (IJRA)*, 1(4):223–234, December 2012.
- [3] Chen, Xue Dong, Watanabe, Keigo, and Izumi, Kiyotaka. Kinematic solution of a quadruped walking robot-posture analysis of titan-viii-. *IFAC 14th Wprld Congress*, pp. 827–832, 1999.
- [4] Darpa. Legged squad support system(ls3)
<https://www.darpa.mil/program/legged-squad-support-system>, 2012.
- [5] Electrical4U. What is servomotor. <https://www.electrical4u.com/what-is-servo-motor/>,
2023.
- [6] Fermoso, Jose. G-dog robot moves swiftly, scares others easily
<https://www.wired.com/2008/11/g-dog-robot/>, 2008.
- [7] Fujita, Masahiro. Aibo: Toward the era of digital creatures. *The International Journal of Robotics Research*, 20(10):781–794, October 2001.

- [8] Greenemeier, Larry. Darpa pushes machine learning with legged littledog robot .
<https://www.scientificamerican.com/article/littledog-darpa/>, 2008.
- [9] Koo, Ig Mo, Kang, Tae Hun, Vo, GiaLoc, Trong, Tran Duc, Song, Young Kuk, and Choi, Hyouk Ryeol. Biologically inspired control of quadruped walking robot. *ICROS International Journal of Control, Automation, and Systems*, 7(4):577–584, 2009.
- [10] Koo, Ig Mo, Tran, Duc Trong, Lee, Yoon Haeng, Moon, Hyungpil, Koo, Ja Choon, Park, Sangdeok, and Choi, Hyouk Ryeol. Development of a quadruped walking robot aidin-iii using biologically inspired kinematic analysis. *ICROS International Journal of Control, Automation, and Systems*, 11(6):1276–1289, 2013.
- [11] Koo, Ig Mo, Tran, Duc Trong, Lee, Yoon Haeng, Moon, Hyungpil, Koo, Ja Choon, Park, Sangdeok, and Choi, Hyouk Ryeol. Biologically inspired gait transition control for a quadruped walking robot. *Springer Science+Business Media New York*, January 2015.
- [12] Koo, Ig Mo, Trong, Tran Duc, Kang, Tae Hun, Vo, GiaLoc, Song, Young Kuk, Lee, Chang Min, and Choi, Hyouk Ryeol. Control of a quadruped walking robot based on biologically inspired approach. *IEEE/RSJ International Conference on Intelligent Robots and Systems San Diego*, pp. 2969–2974, October 2007.
- [13] Li, Yibin, Li, Bin, Ruan, Jiuhong, and Rong, Xuewen. Research of mammal bionic quadruped robots: a review. *IEEE 5th International Conference on Robotics, Automation and Mechatronics*, pp. 166–171, November 2011.
- [14] Pongas, Dimitris, Mistry, Michael, and Schaal, Stefan. A robust quadruped walking gait for traversing rough terrain. *IEEE International Conference on Robotics and Automation*, pp. 1474–1479, April 2007.

- [15] Puig, Alejandro Alonso. Robot world 2009 the south korean robotics exhibition. <http://automacomp.blogspot.com/2009/09/robot-world-2009-south-korean-robotics.html>, 2009.
- [16] Raibert, Marc, Blankespoor, Kevin, Nelson, Gabriel, and Playter, Rob. Bigdog, the rough-terrain quadruped robot. *IFAC The International Federation of Automatic Control*, pp. 10822–10825, July 2008.
- [17] Scherz, Paul and Monk, Simon. *Practical Electronics for Inventors*. McGraw-Hill, 4th ed. , 2016.
- [18] Ugurlu, Barkan, Havoutis, Ioannis, Semini, Claudio, and Caldwell, Darwin G. Dynamic trot-walking with the hydraulic quadruped robot - hyq: Analytical trajectory generation and active compliance control. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, November 2013.
- [۱۹] جوهری مجید، وحید. بررسی پیشرفت‌های فناوری ربات‌های چهارپا. انجمن مهندسان کنترل و ابزار دقیق ایران، ۸(۳):۵۱ تا ۶۸، پاییز ۱۳۹۳.
- [۲۰] حیدری، میلاد. معرفی کامل نرم افزار سالیدورک .۱۴۰۱، <https://sariasan.com/solidworks-tutorial/solidworks-introduction-and-timing/>
- [۲۱] فرهبد، احمد رضا. ربات چیست و چه کابردی دارد؟ .۱۴۰۱، <https://digiato.com/article/2022/11/17/what-is-robot>
- [۲۲] پارس، الکترونک. میکروکنترلر چیست؟ کاربرد و انواع آن .۱۴۰۱، <https://parselectronic.com/what-is-microcontroller-and-types/>

پیوست

کد کامل شامل ۱۳۰۰ خط است که به چندین فایل تقسیم کردیم که ما داخل فایل Main تغییرات را ایجاد می‌کنیم و در اینجا توابع و دستورات مهم را آورده‌یم.

```
/* Includes -----*/
#include "main.h"
#include "dma.h"
#include "tim.h"
#include "usart.h"
#include "gpio.h"

/* USER CODE BEGIN PV */
/*Setting an initial value for the angle of each motor.*/
int Motor_Current_Position[4][5] = {{0, 0, 0, 0, 0}, {0, 79, 75, 77, 78},
{0, 125, 116, 117, 125}, {0, 94, 86, 90, 85}};
int space = 0, count = 0;
char symbol;
uint8_t RxBuffer[4];
/* USER CODE END PV */

/* Private function prototypes -----*/
void SystemClock_Config(void);
/* USER CODE BEGIN PFP */
int Find_max(int pwm_1, int pwm_2, int pwm_3, int pwm_4);
void Move_All(int pwm_1, int pwm_2, int pwm_3, int pwm_4, int motor);
void Move(TIM_HandleTypeDef htim, int timer, int CCR, int pwm, int motor);
void Home(void);
void Forward(int count);
void Backward(int count);
void Turn_Left(int count);
void Turn_Right(int count);
void deCode(void);
void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart);
```

```

MX_GPIO_Init();
MX_DMA_Init();
MX_USART1_UART_Init();
MX_TIM1_Init();
MX_TIM2_Init();
MX_TIM3_Init();
MX_TIM4_Init();
/* USER CODE BEGIN 2 */
/////////////////////////////// Leg1
HAL_TIM_PWM_Start(&htim1, TIM_CHANNEL_1); //A8 => M3
HAL_TIM_PWM_Start(&htim3, TIM_CHANNEL_4); //B1 => M2
HAL_TIM_PWM_Start(&htim4, TIM_CHANNEL_4); //B9 => M1
/////////////////////////////// Leg2
HAL_TIM_PWM_Start(&htim2, TIM_CHANNEL_2); //A1 => M3
HAL_TIM_PWM_Start(&htim2, TIM_CHANNEL_3); //A2 => M2
HAL_TIM_PWM_Start(&htim2, TIM_CHANNEL_4); //A3 => M1
/////////////////////////////// Leg3
HAL_TIM_PWM_Start(&htim3, TIM_CHANNEL_1); //A6 => M1
HAL_TIM_PWM_Start(&htim3, TIM_CHANNEL_2); //A7 => M2
HAL_TIM_PWM_Start(&htim3, TIM_CHANNEL_3); //B0 => M3
/////////////////////////////// Leg4
HAL_TIM_PWM_Start(&htim4, TIM_CHANNEL_1); //B6 => M1
HAL_TIM_PWM_Start(&htim4, TIM_CHANNEL_2); //B7 => M2
HAL_TIM_PWM_Start(&htim4, TIM_CHANNEL_3); //B8 => M3

while (1)
{
    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */
    switch(symbol)
    {
        case 'P':
            if(space < 110)
                Forward(1);
            else
            {
                Turn_Right(5);
                Forward(11);
                Turn_Left(5);
                Forward(11);
                Turn_Right(5);
            }
            break;
        case 'B':
            if(space < 105)
                Forward(1);
            else
                Forward(20);
            break;
    }
}

```

```

/*Finding the maximum value among the four motors.*/
int Find_max(int pwm_1, int pwm_2, int pwm_3, int pwm_4)
{
    int max = pwm_1;
    if(max < pwm_2)
        max = pwm_2;
    if(max < pwm_3)
        max = pwm_3;
    if(max < pwm_4)
        max = pwm_4;
    return max;
}

/*Simultaneously moving all four motors in a row.*/
void Move_All(int pwm_1, int pwm_2, int pwm_3, int pwm_4, int motor)
{
    int n, max;
    max = Find_max(pwm_1, pwm_2, pwm_3, pwm_4);
    switch(motor)
    {
        case 1:
            n = 50;
            for(;n <= max;n++)
            {
                if(pwm_3 >= n)
                    htim3.Instance->CCR1 = n;
                if(pwm_1 >= n)
                    htim4.Instance->CCR4 = n;
                if(pwm_4 >= n)
                    htim4.Instance->CCR1 = n;
                if(pwm_2 >= n)
                    htim2.Instance->CCR4 = n;
                HAL_Delay(10);
            }
            break;
        case 2:
            n = 70;
            for(;n <= max;n++)
            {
                if(pwm_3 >= n)
                    htim3.Instance->CCR2 = n;
                if(pwm_1 >= n)
                    htim3.Instance->CCR4 = n;
                if(pwm_4 >= n)
                    htim4.Instance->CCR2 = n;
                if(pwm_2 >= n)
                    htim2.Instance->CCR3 = n;
                HAL_Delay(10);
            }
            break;
    }
}

```

```

case 3:
    n = 45;
    for(;n <= max;n++)
    {
        if(pwm_3 >= n)
            htim3.Instance->CCR3 = n;
        if(pwm_1 >= n)
            htim1.Instance->CCR1 = n;
        if(pwm_4 >= n)
            htim4.Instance->CCR3 = n;
        if(pwm_2 >= n)
            htim2.Instance->CCR2 = n;
        HAL_Delay(10);
    }
    break;
}
}

/*Moving the motor to a desired angle.*/
void Move(TIM_HandleTypeDef htim, int timer, int CCR, int pwm, int motor)
{
    int n;
    if( timer == 3 && CCR == 3)
        n = Motor_Current_Position[motor][1];
    else if( timer == 4 && CCR == 4)
        n = Motor_Current_Position[motor][1];
    else
        n = Motor_Current_Position[motor][timer];
    if(n <= pwm)
    {
        switch(CCR)
        {
            case 1:
                for(;n <= pwm;n++)
                {
                    htim.Instance->CCR1 = n;
                    HAL_Delay(10);
                }
                break;
            case 2:
                for(;n <= pwm;n++)
                {
                    htim.Instance->CCR2 = n;
                    HAL_Delay(10);
                }
                break;
            case 3:
                for(;n <= pwm;n++)
                {
                    htim.Instance->CCR3 = n;
                    HAL_Delay(10);
                }
                break;
            case 4:
                for(;n <= pwm;n++)
                {
                    htim.Instance->CCR4 = n;
                    HAL_Delay(10);
                }
                break;
        }
    }
}

```

```

else
{
    switch(CCR)
    {
        case 1:
            for(;n >= pwm;n--)
            {
                htim.Instance->CCR1 = n;
                HAL_Delay(10);
            }
            break;
        case 2:
            for(;n >= pwm;n--)
            {
                htim.Instance->CCR2 = n;
                HAL_Delay(10);
            }
            break;
        case 3:
            for(;n >= pwm;n--)
            {
                htim.Instance->CCR3 = n;
                HAL_Delay(10);
            }
            break;
        case 4:
            for(;n >= pwm;n--)
            {
                htim.Instance->CCR4 = n;
                HAL_Delay(10);
            }
            break;
    }
}
if( timer == 3 && CCR == 3)
    Motor_Current_Position[motor][1] = pwm;
else if( timer == 4 && CCR == 4)
    Motor_Current_Position[motor][1] = pwm;
else
    Motor_Current_Position[motor][timer] = pwm;
}

/*Initial state of the robot in complete stability.*/
void Home()
{
    HAL_Delay(2000);
    Move_All(79, 75, 77, 78, 1);
    Move_All(125, 116, 117, 125, 2);
    Move_All(94, 86, 90, 85, 3);
    HAL_Delay(1000);
}

```

```

/*Moving forward by the desired amount.*/
void Forward(int count)
{
    for(int i = 1;i <= count;i++)
    {
        /////////////////
        Move(htim3, 3, 2, 107, 2);
        Move(htim3, 3, 3, 80, 3);
        Move(htim4, 4, 1, 103, 1);
        Move(htim3, 3, 2, 117, 2);
        Move(htim3, 3, 3, 90, 3);
        Move(htim2, 2, 3, 111, 2);
        Move(htim2, 2, 2, 76, 3);
        Move(htim4, 4, 1, 78,1);
        Move(htim2, 2, 3, 116, 2);
        Move(htim2, 2, 2, 86, 3);
        HAL_Delay(300);
        ///////////////
        Move(htim2, 2, 3, 106, 2);
        Move(htim2, 2, 2, 76, 3);
        Move(htim4, 4, 4, 54, 1);
        Move(htim2, 2, 3, 116, 2);
        Move(htim2, 2, 2, 86, 3);
        Move(htim3, 3, 2, 112, 2);
        Move(htim3, 3, 3, 85, 3);
        Move(htim4, 4, 4, 79, 1);
        Move(htim3, 3, 2, 117, 2);
        Move(htim3, 3, 3, 90, 3);
        HAL_Delay(300);
    }
}

/*Moving backward by the desired amount.*/
void Backward(int count)
{
    for(int i = 1;i <= count;i++)
    {
        ///////////////
        Move(htim4, 4, 2, 115, 2);
        Move(htim4, 4, 3, 75, 3);
        Move(htim3, 3, 1, 102, 1);
        Move(htim4, 4, 2, 125, 2);
        Move(htim4, 4, 3, 85, 3);
        Move(htim3, 3, 4, 120, 2);
        Move(htim1, 1, 1, 89, 3);
        Move(htim3, 3, 1, 77,1);
        Move(htim3, 3, 4, 125, 2);
        Move(htim1, 1, 1, 94, 3);
        HAL_Delay(300);
        ///////////////
        Move(htim3, 3, 4, 115, 2);
        Move(htim1, 1, 1, 84, 3);
        Move(htim2, 2, 4, 50, 1);
        Move(htim3, 3, 4, 125, 2);
        Move(htim1, 1, 1, 94, 3);
        Move(htim4, 4, 2, 120, 2);
        Move(htim4, 4, 3, 80, 3);
        Move(htim2, 2, 4, 75, 1);
        Move(htim4, 3, 2, 125, 2);
        Move(htim4, 3, 3, 85, 3);
        HAL_Delay(300);
    }
}

```

```

/*Turning left by the desired amount.*/
void Turn_Left(int count)
{
    Move(htim2, 2, 3, 111, 2);
    for(int i = 1;i <= count;i++)
    {
        /////////////////////////
        Move(htim4, 4, 2, 110, 2);
        Move(htim4, 4, 3, 70, 3);
        Move(htim3, 3, 1, 107, 1);
        Move(htim4, 4, 2, 125, 2);
        Move(htim4, 4, 3, 85, 3);
        /////////////////////////
        Move(htim3, 3, 3, 70, 3);
        Move(htim4, 4, 1, 108, 1);
        Move(htim3, 3, 3, 90, 3);
        for(int p = 1;p <= 30;p++)
        {
            htim3.Instance->CCR1 = 107 - p;
            htim4.Instance->CCR1 = 108 - p;
            HAL_Delay(10);
        }
        Motor_Current_Position[1][3] = 77;
        Motor_Current_Position[1][4] = 78;
        HAL_Delay(300);
    }
    Move(htim2, 2, 3, 116, 2);
}

/*Turning right by the desired amount.*/
void Turn_Right(int count)
{
    Move(htim3, 3, 2, 114, 2);
    for(int i = 1;i <= count;i++)
    {
        /////////////////////////
        Move(htim2, 2, 3, 101, 2);
        Move(htim2, 2, 2, 71, 3);
        Move(htim4, 4, 4, 49, 1);
        Move(htim2, 2, 3, 116, 2);
        Move(htim2, 2, 2, 86, 3);
        /////////////////////////
        Move(htim1, 1, 1, 74, 3);
        Move(htim2, 2, 4, 45, 1);
        Move(htim1, 1, 1, 94, 3);
        for(int p = 1;p <= 30;p++)
        {
            htim4.Instance->CCR4 = 49 + p;
            htim2.Instance->CCR4 = 45 + p;
            HAL_Delay(10);
        }
        Motor_Current_Position[1][1] = 79;
        Motor_Current_Position[1][2] = 75;
        HAL_Delay(300);
    }
    Move(htim3, 3, 2, 117, 2);
}

```

```

void deCode()
{
    int index = 5;
    if(RxBuffer[0]==66 || RxBuffer[0]==78 || RxBuffer[0]==80 || RxBuffer[0] ==89)
    {
        index = 0;
        symbol = RxBuffer[0];
    }
    else if(RxBuffer[1]==66 || RxBuffer[1]==78 || RxBuffer[1]==80 || RxBuffer[1]==89)
    {
        index = 1;
        symbol = RxBuffer[1];
    }
    else if(RxBuffer[2]==66 || RxBuffer[2]==78 || RxBuffer[2]==80 || RxBuffer[2]==89)
    {
        index = 2;
        symbol = RxBuffer[2];
    }
    else if(RxBuffer[3]==66 || RxBuffer[3]==78 || RxBuffer[3]==80 || RxBuffer[3]==89)
    {
        index = 3;
        symbol = RxBuffer[3];
    }
    switch(index)
    {
        case 0:
            space = (RxBuffer[3] - 48) + (RxBuffer[2] - 48) * 10 + (RxBuffer[1] - 48) * 100;
            break;
        case 1:
            space = (RxBuffer[0] - 48) + (RxBuffer[3] - 48) * 10 + (RxBuffer[2] - 48) * 100;
            break;
        case 2:
            space = (RxBuffer[1] - 48) + (RxBuffer[0] - 48) * 10 + (RxBuffer[3] - 48) * 100;
            break;
        case 3:
            space = (RxBuffer[2] - 48) + (RxBuffer[1] - 48) * 10 + (RxBuffer[0] - 48) * 100;
            break;
    }
}
.....
void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart)
{
    if(huart == &huart1){
        deCode();
    }
}

```

Abstract

In today's world, robots play a significant role in human life. Mobile robots, due to their interaction with the environment, differ from other machines. They can create changes in their surroundings based on their actions and respond to the world around them. The diverse applications of robots in simplifying human life have led engineers to pay special attention to robots.

Among all mobile robots, quadruped robots are a type of legged robot that, due to their ability to explore all types of terrains like humans and animals, outperform wheeled robots. The advantage of quadruped robots is that they maintain their stability when standing still and require control when walking.

In this thesis, we first introduced robots, their types, with a focus on constructed quadruped robots, and their applications. Then, we delved into the discussion of designing a model and building a robot using SolidWorks software. In this design process, an initial model was created, and efforts were made to improve its performance after addressing its flaws, resulting in a refined final version. Subsequently, we discussed the hardware and components used in the robot, including motors, microcontrollers, and various modules. Finally, we described the algorithms used and the programming on the robot's microcontroller.

Key Words:

Mobie Robots, Quadrupedal Robot, Legged Robot, SolidWorks, Algorithm, Microcontroller



**Amirkabir University of Technology
(Tehran Polytechnic)**

Electrical Engineering Department

B. SC. Thesis

Design and Manufacture of Four-Legged Robot

By
Mohammad Barabadi

Supervisor
Dr. Mohammad Azam Khosravi

Advisor
Dr. Masoud Shafiee

September 2023