



دانشگاه صنعتی امیرکبیر

(پلی‌تکنیک تهران)

دانشکده برق

پروژه کارشناسی

گرایش کنترل

بهبود عملکرد ربات اسپایدر و پیاده سازی الگوریتم برنامه
ریزی مسیر برای آن

نگارش

سجاد قدیری

استاد راهنما

دکتر محمد اعظم خسروی

تابستان ۱۴۰۲

بِسْمِ اللّٰهِ الرَّحْمٰنِ الرَّحِيْمِ

اکنون که به یاری پروردگار و راهنمایی اساتید بزرگ موفق به اتمام این بخش از مسیر علمی خود شده
ام وظیفه خود دانسته که نهایت سپاسگزاری را از تمامی عزیزانی که در این راه به من کمک کرده اند را
به عمل آورم:

در آغاز از استاد بزرگوارم جناب دکتر محمد اعظم خسروی که راهنمایی این پایان نامه را به عهده
داشته اند کمال تشکر را دارم.
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 محمد برابادی که همواره یار و یاوری صبور بود کمال تشکر را دارم.
در آخر نیز خالصانه از تمامی عزیزان اعم از هم دانشگاهیان، همراهان عزیز و دوستان خوبم که در مقاطع
مختلف تحصیلی به هر نحوی بنده را یاری کرده نهایت سپاس را دارم.

چکیده

ربات‌ها از جدیدترین دستاوردهای انسان و تلفیقی از فناوری در حوزه‌های مختلفی همچون کنترل، هوش مصنوعی و اتوماسیون و... هستند. امروزه هوشمندسازی فناوری‌های پیشین موردتوجه بوده و ربات‌ها نیز از آن مستثنی نیستند. از میان انواع ربات‌ها آن دسته‌ای که در ساخت آنها از طبیعت الهام گرفته شده است دارای چالش‌های نوینی بوده‌اند. ربات‌های عنکبوتی چهارپا نمونه‌ای از این دسته هستند. این ربات‌ها از لحاظ مکانیزم حرکتی و همچنان کاربرد دارای تفاوت‌هایی با ربات‌های چرخ‌دار و... هستند. این پژوهش به تلاش‌هایی که در جهت بهبود عملکرد و مسیریابی ربات عنکبوتی چهارپا انجام شده می‌پردازد. بدین منظور قابلیت‌هایی به ربات اضافه گردیده و سپس به بهره‌گیری از این قابلیت‌ها در جهت بهبود عملکرد ربات تلاش شده است. در ابتدا یک مارژول چندجانبه که شامل دوربین است به ربات اضافه شده و به طراحی یک سرور باهدف تبادل اطلاعات بهصورت بیسیم میان خود و ربات پرداخته می‌شود. از این ویژگی برای نظارت و ارزیابی صحت اطلاعات و همچنین تصمیم‌گیری‌های ربات در محیط استفاده خواهد شد. در ادامه تشخیص موانع موجود در محیط به کمک تشخیص رنگ و روش‌های پردازش تصویر با استفاده از زبان پایتون ارائه می‌گردد. بهمنظور موقعیت‌سنجی ربات در محیط از کتابخانه‌ای که اساس کار آن بهره‌گیری از تگ‌های حاوی اطلاعات است، استفاده شده است. همچنین برای هدف نهایی ربات یعنی طی کردن مسیری از نقطه شروع به نقطه پایان، یک الگوریتم مسیریابی شبیه‌سازی شد. همچنین در جهت تلفیق این الگوریتم با ویژگی‌های ربات برای پیاده‌سازی عملی، تغییرات موردنیاز مطرح و بررسی می‌گردد. در انتهای آزمایش‌های عملی بر روی ربات با دو سناریو مختلف صورت گرفته و نتایج آن ارائه می‌گردد. ارائه چالش‌ها و پیشنهاداتی برای کارهای پیش رو نیز مختصراً انجام گردیده است.

واژه‌های کلیدی:

ربات عنکبوتی چهارپا ، دوربین، تشخیص موانع، مسیریابی با روش RRT ، مکانیابی با استفاده از تگ‌های Aruco

فهرست مطالب

صفحه

عنوان

۱	۱ مقدمه
۲	۱-۱ مقدمه
۲	۲-۱ ربات و انواع آن
۲	۱-۲-۱ ربات‌های سری
۳	۲-۲-۱ ربات‌های موازی
۴	۳-۲-۱ ربات‌های متحرک
۴	۳-۱ ربات عنکبوتی چهارپا
۴	۱-۳-۱ ربات‌های عنکبوتی
۵	۱-۱-۳-۱ پایداری استاتیکی
۵	۱-۲-۳-۱ ربات‌های عنکبوتی چهارپا
۶	۴-۱ تاثیر هوش مصنوعی بر رباتیک
۷	۱-۵ مروری بر الگوریتم‌های مسیریابی
۸	۶-۱ اهداف و نواوری
۸	۷-۱ ساختار پایان‌نامه
۹	۸-۱ جمع‌بندی
۱۰	۲ مسیریابی و الگوریتم‌های پیاده سازی شده
۱۱	۱-۲ مقدمه
۱۱	۲-۲ پردازش تصویر
۱۱	۱-۲-۲ مقدمه
۱۲	۲-۲-۲ تصویر چیست؟
۱۲	۳-۲-۲ پردازش تصویر چیست؟
۱۳	۳-۲ تشخیص موائع
۱۳	۱-۳-۲ تشخیص رنگ
۱۴	۲-۳-۲ تشخیص لبه‌ها
۱۴	۳-۳-۲ شبکه‌کد و تشریح جزئیات آن
۱۵	۴-۲ الگوریتم درخت جستجوی تصادفی
۱۶	۱-۴-۲ شبیه‌سازی دوبعدی با پایتون
۱۶	۱-۱-۴-۲ شبکه‌کد و تشریح جزئیات برنامه
۱۸	۲-۱-۴-۲ ادغام الگوریتم با قابلیت‌های ربات به منظور پیاده‌سازی
۱۸	Localization ۵-۲
۱۹	۱-۵-۲ استفاده از تگ

۱۹	۱-۱-۵-۲ مفاهیم اصلی
۱۹	۲-۱-۵-۲ شبکه کد و تشریح جزئیات برنامه
۲۳	۶-۲ جمع بندی
۲۴	۳ نرم افزار و ارتباطات
۲۵	۱-۳ ارتباطات و سرور
۲۵	۱-۱-۳ اتصال بیسیم دوربین به سرور
۲۵	۱-۱-۱-۳ مفاهیم اصلی
۲۵	۲-۱-۱-۳ شبکه کد و تشریح جزئیات برنامه
۲۹	۲-۱-۳ Handlers
۲۹	۱-۲-۱-۳ نحوه کارکرد سرور و نقش هندرلرها
۲۹	۲-۲-۱-۳ شبکه کد و تشریح جزئیات برنامه
۳۲	۳-۲-۱-۳ کوئری استرینگ
۳۴	۴-۲-۱-۳ تابع هندرلر
۳۵	۳-۱-۳ ارتباط USART
۳۶	۲-۳ اتصال دو برد به یکدیگر
۳۷	۳-۳ جمع بندی
۳۸	۴ ساخت افزار الکترونیکی ربات
۳۹	۱-۴ مقدمه
۳۹	۲-۴ پردازنده ARM سری F103C8T6
۴۰	۳-۴ ماژول ESP32-CAM
۴۱	۴-۴ دوربین OV2640
۴۳	۵-۴ سرموتور
۴۳	۱-۵-۴ روش PWM
۴۳	۱-۱-۵-۴ مدولاسیون چیست؟
۴۴	۲-۱-۵-۴ مدولاسیون پهنهای پالس
۴۴	۶-۴ تجهیزات مکانیکی
۴۴	۱-۶-۴ پین هدر
۴۵	۷-۴ جمع بندی
۴۶	۵ نتایج و پیشنهادات
۴۷	۱-۵ نتایج
۴۷	۱-۱-۵ نتایج شبیه‌سازی الگوریتم درخت جستجوی تصادفی
۴۸	۲-۱-۵ نتیجه تشخیص مانع

فهرست مطالب

۴۹	۳-۱-۵ نتایج پیاده‌سازی عملی
۴۹	۴-۱-۵ نتایج مکان‌یابی
۵۰	۲-۵ پیشنهادات
۵۱	کتابنامه
۵۳	پیوست
۶۴	۱- کد Localization

صفحه	فهرست تصاویر	شكل
۳	۱-۱ ربات سری
۳	۲-۱ ربات موازی
۴	۳-۱ ربات کشاورزی
۵	۴-۱ ربات عنکبوتی
۶	۵-۱ پایداری استاتیکی
۶	۶-۱ نسخه اولیه طراحی شده
۹	۷-۱ نسخه اسمبل شده
۱۶	۳-۲ الگوریتم RRT
۱۸	۴-۲ مراحل تولید گره جدید و تشکیل مسیر
۱۹	۵-۲ آزمایشگاه اپریل میشیگان
۲۵	۱-۳ نقطه دسترسی و مُد ایستگاهی
۳۲	۲-۳ ساختار کوئری استرینگ
۳۶	۳-۳ نحوه اتصال دو برد به یکدیگر
۴۱	۲-۴ مژول ESP32CAM
۴۲	۳-۴ دوربین OV2640
۴۳	۴-۴ سروموتور SG90
۴۴	۵-۴ پین هدر نر
۴۵	۶-۴ پین هدر ماده
۴۷	۱-۵ شبیه سازی الگوریتم RRT
۴۸	۲-۵ مانع مکعب مستطیل با رنگ صورتی
۴۹	۳-۵ تشخیص رنگ آبی
۴۹	۴-۵ زوایا و فوائل از تگ

فهرست نمادها

فصل اول

مقدمه

۱-۱ مقدمه

با پیشرفت روزافزون دانش بشری در دهه اخیر و تلفیق هرچه بیشتر شاخه‌های علمی با یکدیگر و همچنین همگام‌شدن و بهره‌مندی از فناوری‌های موجود شاهد ساخته شدن محصولات جدیدی هستیم که ربات‌ها یکی از محبوب‌ترین این محصولات هستند.

در این پایان‌نامه، به اختصار به طراحی و پیاده‌سازی ربات چهارپایی عنکبوتی را که ساخته شده است، اشاره شده و سپس به تفصیل درباره انتخاب و کالیبره‌سازی دوربین، نحوه ارتباطات بین قطعات الکترونیکی و مکانیکی، الگوریتم تشخیص موانع و نحوه اجرای و ارزیابی آزمایش‌ها خواهیم پرداخت. همچنین، نتایج و تحلیل‌های حاصل از آزمایش‌ها به منظور ارزیابی کارایی و کاربردی بودن این روش در محیط‌های واقعی را ارائه خواهیم داد.

در انتهای نیز به بررسی چالش‌های پیاده‌سازی پرداخته و پیشنهاداتی برای کارهای آینده خواهیم پرداخت.

۲-۱ ربات و انواع آن

۱-۲-۱ ربات‌های سری

به طور معمول ربات‌ها با توجه به نوع کاربردشان، به شکل سری، موازی و یا ترکیبی از هر دو ساخته می‌شوند. همان‌طور که در شکل ۱-۱ نشان‌داده شده است، ربات‌های سری از یک زنجیره سینماتیکی تشکیل شده‌اند که در آن مفاصل در یک ارتباط سری باهم قرار می‌گیرند. این ربات‌ها به دلیل ویژگی داشتن فضای کاری بزرگ‌تر در صنعت بسیار پرکاربرد هستند. در این ربات‌ها عموماً عملگرها^۱ با قرارگرفتن به صورت سری در هر مفصل قرار یک درجه آزادی ایجاد می‌کنند. علی‌رغم مزیت‌های ذکر شده برای این دسته از ربات‌ها باید توجه داشت که ربات‌های سری لزوماً برای تمامی کاربردهای صنعتی و حتی غیرصنعتی، بهترین انتخاب نبوده و معایبی نیز به همراه دارند. از جمله این معایب آنکه خطاهای موقعیتی در آنها جمع‌شونده هستند. از دیدگاه انرژی میزان مصرف انرژی در این ربات‌ها بالا است؛ زیرا هر کدام از مفاصل تحریک‌شده نه تنها بار را حمل می‌کنند بلکه باید بازوها ماقبل خود را نیز جابه‌جا نماید. در نتیجه برای جابه‌جایی بارهای سنگین بازوها قوی‌تری موردنیاز است. در ربات‌های سری تمامی مفاصل باید کار کنند، اگر یک مفصل از کار بیفتد یا آزاد گذاشته شود ساختار ربات به هم ریخته و در این حالت ربات قادر به حفظ موقعیت خود و انجام وظیفه^۲ محوله نخواهد بود.

¹Actuators

²Task



شکل ۱-۱: ربات سری [۱]

۲-۲-۱ ربات‌های موازی

به مرور زمان احساس نیاز به ساختارهایی که محدودیت‌های ربات‌های سری را نداشتند باشند به وجود آمد. همان‌گونه که ما انسان‌ها برای حرکت‌های دقیق و یا برداشتن اجسام سنگین از هر دوست خود استفاده می‌کنیم، می‌توان چنین تصور کرد که استفاده از زنجیره‌های سینماتیکی بسته که چند بازو باهم و به صورت موازی در حرکت دخیل باشند می‌توانند پاسخی درخور برای این مشکل باشد.

همان‌طور که در شکل ۲-۱ مشاهده می‌شود ربات‌های موازی از زنجیره‌های سینماتیکی ساخته می‌شوند که شامل یک تکیه‌گاه^۱ و یک صفحه متحرک^۲ که به وسیله تعدادی عملگر یا رابط به یکدیگر متصل شده‌اند، است. مهم‌ترین ضعف ربات‌های موازی محدودیت در فضای کاری آنهاست.



شکل ۱-۲: ربات موازی [۲]

^۱Base

^۲Moving Platform

۱-۲-۳ ربات‌های متحرک

در دهه‌های اخیر، ربات‌های متحرک^۱ به عنوان ابزاری چندمنظوره و قدرتمند در دنیای مدرن از ماشین‌های خودکار تلقی می‌شوند. ربات‌های متحرک، از جمله دستاوردهای بزرگ در زمینه‌های مختلف از تحقیقات تا کاربردهای عملی، توانسته‌اند نقش مهمی را ایفا کنند. از تعقیب کاربردهای صنعتی تا خدمات انسانی و حتی مسائل محیط‌زیستی، علی‌رغم تنوع گسترده شکل‌ها و ساختارها، ربات‌های متحرک، توانایی انجام وظایف متنوع در محیط‌های مختلفی مانند صنایع کشاورزی، خدماتی و... را دارا هستند.



شکل ۱-۳: ربات کشاورزی [۳]

ربات‌های متحرک در انواع مختلف طراحی می‌شوند که هر یک برای انجام وظایف خاصی از مکانیزم‌های متمایز استفاده می‌کنند. ربات‌های چرخ دار^۲، به عنوان مثال، از چرخ‌های محرک دارای موتورها برای حرکت و جابه‌جایی استفاده می‌کنند. این‌گونه ربات‌ها از استیکرهای تفاضلی بهره می‌برند که چرخ‌های موتورها در سمت‌های مختلف با سرعت‌های متفاوت می‌چرخند تا امکان تغییر جهت را فراهم کنند. ربات‌های پادار^۳ تلاش می‌کنند تا حرکت حیوانات را تقلید کنند و از پاهای برای حرکت استفاده می‌کنند. کنترل ربات‌های پادار به دلیل درجات آزادی متعدد در مفاصل پیچیده است. ربات‌های هوایی مانند پهپادها، با استفاده از پره‌ها برای تولید لیفت و تراکم برای پرواز بهره می‌برند. این ربات‌ها نیاز به الگوریتم‌های کنترل پیچیده دارند تا پایداری و مسیر طی شده را مدیریت کنند [۱۷].

۱-۳ ربات عنکبوتی چهارپا

۱-۳-۱ ربات‌های عنکبوتی

راه‌رفتن با چهارپا برای اکثر حیوانات رایج است و دلیل خوبی برای تکرار آن در ربات‌ها وجود دارد. ربات‌های عنکبوتی^۴، از جمله انواع پیشرفته ربات‌ها، طراحی شده‌اند تا با تقلید از ساختار حرکت و عملکرد

¹Mobile Robots

²Wheeled Robots

³Legged Robots

⁴Spider Robots

عنکبوت‌ها، قابلیت‌های منحصر به فردی در زمینه حرکت و کنترل را ارائه دهند. این ربات‌ها با دارابودن چندین پا و قابلیت تحرک انعطاف‌پذیر، می‌توانند در محیط‌های مختلف و متغیر عملکرد خوبی داشته باشند. در میان ربات‌های پادار، ربات‌های چهارپا به علت پایداری مناسب‌تر نسبت به ربات‌های دوپا و نیز تعداد پاهای کمتر نسبت به ربات‌های شش پا، پیچیدگی کمتری را در طراحی و عمل دارند. ربات‌های چهارپا دارای پایداری استاتیکی هستند و الگوی راه‌رفتن یک ربات چهارپا را می‌توان به روش‌های مختلف طراحی کرد.



شکل ۱-۴: ربات عنکبوتوی [۴]

در این پروژه، به بررسی ربات عنکبوتوی چهارپا می‌پردازیم که با تشکیل یک ساختار چهارپا و مکانیزم‌های تحرک پیچیده، قادر به مسیریابی و انجام وظایف متنوع در محیط آزمایشگاهی است.

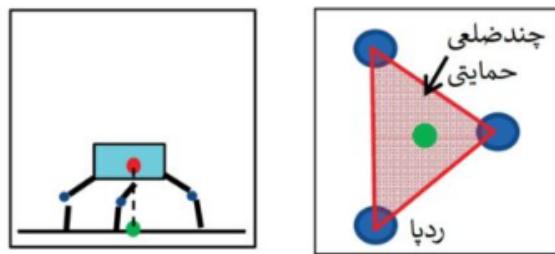
۱-۳-۱ پایداری استاتیکی

یک ربات با پایداری استاتیک تعادل خوبی دارد و در هنگام ایستادن به زمین نمی‌خورد. این بدین معنی است که مرکز ثقل ربات درون پایه تماس با زمین قرار دارد. فرض کنید یک ربات سه پا داریم که پاهای به صورت مثلث تنظیم شده‌اند. این ربات تا زمانی که مرکز ثقل داخل مثلث قرار دارد، به هیچ نوع جایه‌جایی برای ثابت ایستادن نیاز ندارد. این مثلث را «چندضلعی حمایتی» می‌نامند که ناحیه‌ای افقی بالای مکان مرکز ثقل است تا پایداری استاتیک به دست آید. اگر جملات قبلی نامفهوم هستند، فقط کافی است متوجه شوید که چندضلعی حمایتی همان سطحی است که ربات روی آن ایستاده و درون نقاط حمایتی قرار دارد.

۲-۳-۱ ربات‌های عنکبوتوی چهارپا

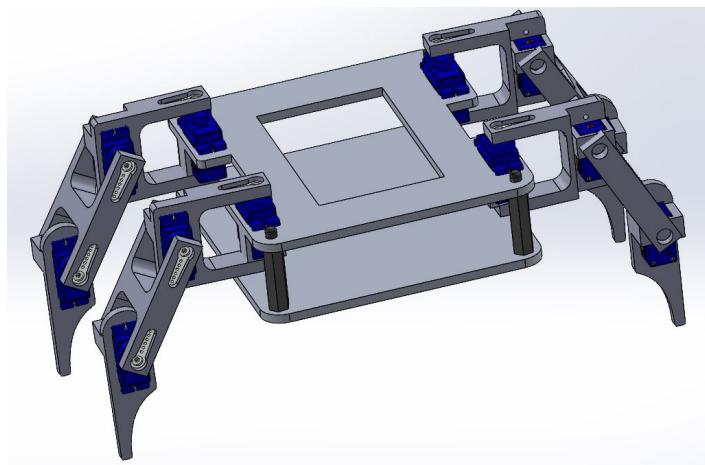
ربات‌های عنکبوتوی چهارپا^۱، با شباهت به حرکت عنکبوت‌های طبیعی، به کمک چهارپا محرک تحرک کرده و قادر به تغییر شکل در محیط‌های مختلف هستند. این انعطاف‌پذیری در حرکت، به ربات‌ها

^۱Quadruped Spider Robots



شکل ۱-۵: چندضلعی حمایتی [۵]

امکان حرکت در مسیرهای مختلف، شکل‌گیری برای عبور از موانع و تطابق با محیط را می‌دهد. علاوه بر این، استفاده از حسگرهای چندگانه مانند دوربین‌ها، ژیروسکوپ‌ها و... به ربات‌ها امکان مشاهده و تجزیه و تحلیل محیط را می‌دهد. این قابلیت‌ها به عنوان اصول مهم در ناوبری، پیمایش محیط و انجام وظایف متنوع در برنامه‌ریزی و کنترل ربات‌های عنکبوتی چهارپا مورداستفاده قرار می‌گیرد. برای درک بهتر یک نمونه از ربات‌های عنکبوتی چهارپا که در واقع همان ربات پایان‌نامه پیش روست و طراحی اولیه آن در محیط سالیدورک انجام شده است، در شکل ۱-۶ قابل مشاهده می‌باشد.



شکل ۱-۶: نسخه اولیه طراحی شده

۴-۱ تاثیر هوش مصنوعی بر رباتیک

در دهه‌های اخیر، پیشرفت‌های چشمگیری در زمینه هوش مصنوعی و رباتیک ایجاد شده است که به طور گسترده تأثیرات قابل توجهی را در صنعت و علم رباتیک ایجاد کرده است. هوش مصنوعی به عنوان یکی از شاخه‌های علوم کامپیوتر که در تلاش برای تجسم و شبیه‌سازی هوش انسانی بوده و همچنین توانایی یادگیری و تصمیم‌گیری در ماشین‌ها را دارد.

ترکیب هوش مصنوعی با رباتیک، باعث پدیدآمدن ربات‌های هوشمند و تعاملی شده است که

توانایی‌هایی انسان‌نمایانه از جمله تشخیص محیط، پردازش اطلاعات، تصمیم‌گیری و انجام وظایف پیچیده را اجرا می‌کنند. این ترکیب موجب ایجاد امکانات جدید در زمینه‌های مختلف مانند صنعت، پژوهشکی، کشاورزی هوشمند، خودروهای خودران و بسیاری دیگر شده است.

در این پایان‌نامه به بررسی اثرات هوش مصنوعی بر رباتیک در یک نمونه خاص از ربات‌ها (ربات چهارپا) و همچنین کاربردها و چالش‌های متنوعی که این ترکیب مهارت‌ها ایجاد می‌کنند، پرداخته خواهد شد. این نمونه خاص از ربات‌ها مثال خوبی از تعامل موثر هوش مصنوعی و رباتیک در مختلف زمینه‌ها می‌پردازد.

۱-۵ مروری بر الگوریتم‌های مسیریابی

امروزه با ساخت ربات‌های مختلف، برنامه مساله‌ریزی مسیر ربات، بسیار مورد توجه قرار گرفته است. به طور خلاصه، مساله برنامه‌ریزی مسیر، یافتن مسیری است که ربات را از نقطه شروع به نقطه هدف هدایت می‌کند [۱۸]. در بسیاری از کاربردها برای اینکه ربات بتواند وظیفه خود را بدرسی انجام دهد، نیاز به حرکت در محیط دارد. الزام حرکت در محیط موجب پرسیدن این سوال می‌شود که، ربات برای انجام وظیفه خود چه مسیری را طی کند که علاوه بر این بودن آن، با کمترین هزینه به هدف خود برسد. با توجه به محدودیت انرژی، تسریع در انجام کارها و عدم برخورد ربات با موانع موجود در محیط، اهمیت این سوال بهتر درک می‌شود. پاسخگویی به این پرسش، که پرسش اساسی در زمینه تحقیقاتی مهمی تحت عنوان برنامه‌ریزی مسیر است، از دیدگاه‌های مختلف صورت گرفته است.

رویکردهای حل برنامه مسئله ریزی مسیر را می‌توان به چند دسته کلی تقسیم کرد:

۱. بلاذرنگ: اگر ربات از واقعی محیطی، مکان و حرکت موانع، از قبل اطلاع نداشته باشد، آنگاه به یافتن مسیری ایمن به سوی هدف مبادرت کند، در این حالت برنامه‌ریزی بلاذرنگ نامیده می‌شود [۱۸].

۲. کامل و کامل احتمالاتی: ارائه یک مسیر، در صورت وجود، کامل بودن الگوریتم نامیده می‌شود. کامل بودن احتمالاتی نیز به این معنی است که اگر الگوریتم تا زمان طولانی اجرا شود، در صورت وجود یک مسیر، آن را خواهد یافت [۱۸].

از جمله روش‌های مطرح برای حل برنامه مساله‌ریزی مسیر، الگوریتم‌های مبتنی بر نمونه‌گیری هستند. در این روش‌ها با نمونه‌برداری از فضای پیکربندی، به ساخت درختی از نقطه شروع، مبادرت می‌شود. با رسیدن یکی از شاخه‌های درخت به نقطه هدف الگوریتم خاتمه می‌یابد. این روش در ابتدا توسط Lavalle در سال ۱۹۹۱ تحت عنوان درخت جستجوی سریع تصادفی RRT مطرح گردید.

۶-۱ اهداف و نوآوری

با نگاهی به پژوهش‌های انجام شده در زمینه ربات‌های عنکبوتی چهارپا مشاهده می‌شود که در اکثر پژوهش‌های صورت گرفته به دیدگاه تئوری بیشتر اهمیت داده شده است. از این‌رو در پژوهش پیش‌رو تلاش شده است تا جنبه‌های عملی ربات‌های پادار مورد توجه قرار گرفته و همچنین تمرکز بر روی بهبود عملکرد و افزایش قابلیت‌های آنها گردد. در نتیجه این‌هدف، با انتخاب مکانیزمی سهل‌تر برای حرکت در محیط، به انجام وظایف محوله توسط ربات پرداخته شده است. از مهمترین نوآوری این پژوهش، می‌توان به موارد زیر اشاره کرد:

- افزودن دوربین به ربات به منظور تشخیص موانع مختلف موجود در محیط
- مانیتورینگ بلاذرنگ^۱ تصویر محیط
- مدیریت برخط^۲ اطلاعات ارسالی از محیط به ربات و بالعکس

۷-۱ ساختار پایان‌نامه

در این پایان‌نامه به بهبود عملکرد ربات عنکبوتی چهارپا ساخته شده از طریق افزودن یک دوربین و طراحی یک سرور برای تبادل اطلاعات به منظور مسیریابی در محیط پرداخته می‌شود. بدین منظور در فصل ۱ و در مقدمه‌ای که مطرح شد، مروری بر انواع ربات‌ها، الگوریتم‌های مسیریابی و تأثیر هوش مصنوعی بر رباتیک ارائه گردید. در ادامه در فصل ۲ مفاهیم اولیه پردازش تصویر معرفی و به بررسی نحوه تشخیص موانع و شبیه‌سازی الگوریتم مسیریابی درخت جستجوی تصادفی و ارائه راهکاری برای مکان‌یابی ربات پرداخته شده است.

در ادامه و در فصل ۳ به تشریح نرم‌افزار و ارتباطات موجود با ربات پرداخته شده است. بدین منظور پس از بیان مفاهیم اولیه شبکه، در بخش ۲-۱-۱-۳ نحوه اتصال بیسیم دوربین ربات به سرور ارائه گردیده است. در ادامه این فصل، نقش Handler‌ها در دریافت و ارسال داده‌ها تبیین گردیده و به بهره‌گیری از روش‌هایی همچون Query String‌ها اشاره شده است. در فصل ۴ مروری بر سخت‌افزار به کار برده شده در ربات انجام شده است. در این فصل، به تشریح مشخصات دقیق قطعات الکترونیکی، موتورهای ربات و نحوه عملکرد آن‌ها پرداخته شده و به برخی از تجهیزات مکانیکی اشاره گردیده است.

در انتها و در فصل ۵ به بررسی نتایج فصل‌های قبل پرداخته و سپس با ارائه پیشنهاداتی در راستای بهبود روند کارهای آینده تلاش شده است.

¹Real Time

²Online

۱-۱ جمع بندی

با توجه به طراحی اولیه برای ربات و بررسی قابلیت‌های قابل ارتقا، تمرکز اصلی بر روی بهبود مداوم و حداکثری ربات (از لحاظ نکات طراحی و ساخت و همچنین بخش نرم‌افزاری آن) در طی مراحل مختلف قرار گرفت. در نهایت با ساخت نمونه اولیه و اسambil^۱ شدن آن به شروع جدی‌تر بخش نرم‌افزار و برطرف کردن چالش‌های عملی آن پرداخته شد. نمونه اسambil شده اولیه ربات در شکل ۷-۱ قابل مشاهده می‌باشد.



شکل ۷-۱: نسخه اسambil شده

^۱Assemble

فصل دوم

مسیریابی و الگوریتم های پیاده سازی شده

۱-۲ مقدمه

در این فصل پایان نامه به پردازش تصویر و ارتباط بین ربات و سرور می‌باشد. این فصل به معرفی و بررسی اصول و مبانی پردازش تصویر و نیز نحوه ارتباط و تبادل اطلاعات بین ربات چهارپا و سرور می‌پردازد. در دنیای امروز که هوش مصنوعی و فناوری‌های رباتیک و در حال پیشرفت های صنعتی می‌باشند، تلفیق پردازش تصویر به عنوان یک افزونه از ربات و ارتباط میان ربات و سرور، برای انجام وظایف پیچیده و کاربردهای متنوع بسیار حائز اهمیت می‌باشد.

این فصل به بررسی مفاهیم پایه پردازش تصویر، تکنیک‌های استخراج و تحلیل اطلاعات از تصاویر، و همچنین نحوه ارتباط بین ربات و سرور اختصاص دارد. از جمله مسائل مورد بررسی در این فصل، می‌توان به پیش‌پردازش تصاویر، تشخیص الگوها، تصویرسازی داده‌ها و ارسال اطلاعات بین ربات و سرور اشاره کرد.

به عنوان پایه‌ای برای فصل‌های بعدی، این فصل به ما امکان می‌دهد تا اصول پایه پردازش تصویر را برای کنترل و هدایت ربات‌ها بهره‌برداری کنیم. همچنین، اهمیت برقراری ارتباط مؤثر بین ربات و سرور را برای جمع‌آوری داده‌ها و انتقال دستورات کنترلی تأکید می‌کند.

در این فصل، ابتدا به مقدمه‌ای کوتاه درباره موضوع، اهمیت و اهداف این فصل می‌پردازیم. سپس به توضیح دقیق‌تر مفاهیم پایه پردازش تصویر و ارتباطات ربات و سرور می‌پردازیم. در پایان نیز ساختار و مرور مطالب فصل‌های آینده را معرفی خواهیم کرد.

با توجه به اهمیت این فصل در ساختار کلی پایان نامه و کاربردهای آینده، مطالب ارائه شده در این فصل بهترین پایه‌ها و مفاهیم را برای فهم بهتر و تحلیل دقیق‌تر موضوعات بحرانی در پژوهش ارائه خواهد داد.

۲-۲ پردازش تصویر

۱-۲-۲ مقدمه

از مهمترین موضوعات هوش مصنوعی که در سال‌های اخیر حوزه‌های مختلف به ویژه مهندسی را تحت تاثیرات بسزایی قرار داده است، می‌توان به پردازش تصویر و بینایی ماشین اشاره کرد. کاربردهای کنترلی همچون ماشین‌های خودران، دوربین‌های کنترل جرایم رانندگی، سیستم‌های تشخیص چهره و... از پردازش تصویر بهره می‌برند.

۲-۲-۲ تصویر چیست؟

تصویر، نوعی نمایش بصری از داده هاست که معمولاً توسط دوربین ها یا سنسورهای تصویربرداری ثبت می شود. تصویر به صورت مجموعه ای از نقاط کوچک به نام پیکسل ها تشکیل می شود که هر کدام حاوی اطلاعاتی چون رنگ و روشنایی می باشند. از طریق ترکیب مختلف پیکسل ها، تصویرهایی با پیچیدگی، اشکال و جزئیات مختلف ایجاد می شوند.

تصاویر می توانند در انواع مختلف از جمله تصاویر رنگی، تصاویر سیاه و سفید و... باشند. از تصاویر به عنوان یک ابزار قوی برای انتقال اطلاعات و نمایش داده ها در زمینه های مختلف از جمله در پزشکی، هنر، رباتیک و... استفاده می شود.

۳-۲-۲ پردازش تصویر چیست؟

پردازش تصویر یک زیرشاخه از پردازش سیگنال است که به تحلیل، تغییر، و بهبود تصاویر دیجیتالی می پردازد. در این فرایند، تصاویر از دوربین ها یا سنسورهای تصویربرداری گرفته می شوند و سپس توسط الگوریتم ها و روش های پردازشی مختلف تجزیه و تحلیل می شوند.

هدف اصلی پردازش تصویر بهینه سازی و بهبود کیفیت تصاویر است. این امر می تواند شامل کاهش نویز، تعدیل رنگ و کنتراست، تشخیص الگوها، تشخیص ویژگی ها و استخراج اطلاعات از تصاویر باشد. در زمینه های مختلف از پردازش تصویر استفاده می شود که از جمله آن ها می توان به پزشکی (تشخیص بیماری ها از تصاویر پرتونگاری)، رباتیک (شناسایی محیط توسط ربات ها)، و امنیت (تشخیص چهره ها و اجسام در تصاویر نظارتی) اشاره کرد.

پردازش تصویر شامل مراحل مختلفی مانند پیش پردازش، تبدیلات، تشخیص الگوها، و استخراج ویژگی ها است. الگوریتم های پردازش تصویر معمولاً بر اساس مفاهیم ریاضی، آمار و هوش مصنوعی طراحی می شوند. از این رو می توان گفت پردازش تصویر در حوزه های مختلف از مهندسی، علوم رایانه، و علوم پایه کاربردهای متعددی داشته و از اهمیت بالایی در تحلیل و انتقال اطلاعات تصویری برخوردار است.

۳-۲ تشخیص موانع

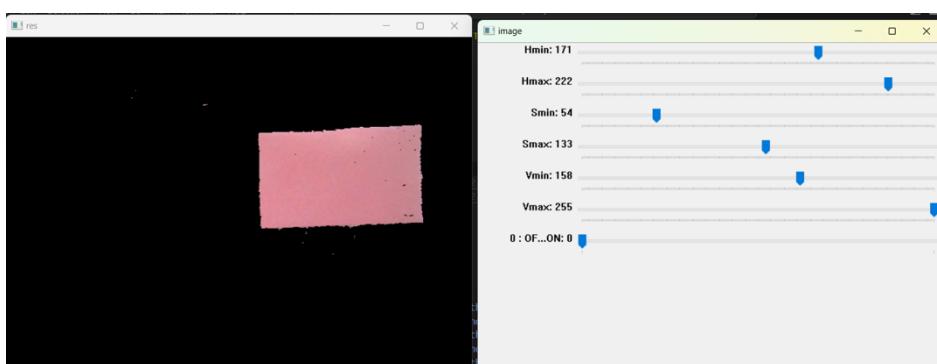
در حوزه رباتیک و بخصوص در توسعه ربات های چهارپا، تشخیص موانع یکی از مسائل بنیادین و حیاتی است که ابزارهای پیشرفته تصویربرداری و پردازش تصویر را به کار می گیرد. در این بخش به تشخیص موانع با تمرکز بر روش های تشخیص رنگ و تشخیص لبه پرداخته خواهد شد. هدف اصلی این بخش، توضیح نحوه استفاده از این روش ها به منظور تشخیص و تجزیه و تحلیل موانع است. این تکنیک ها به ربات ها اجازه می دهند تا محیط خود را بشناسند و در مسیر حرکت خود موانع را شناسایی و از آن ها دور بشوند یا با آن ها تعامل کنند. بررسی این روش ها نقطه عطفی مهم در توسعه ربات های پویا و هوش مصنوعی موجود است.

ددر ادامه، به بررسی مفاهیم تشخیص رنگ و تشخیص لبه خواهیم پرداخت تا درک عمیق تری از این مفاهیم و نحوه اجرای آن ها در مسائل تشخیص موانع داشته باشیم.

۱-۳-۲ تشخیص رنگ

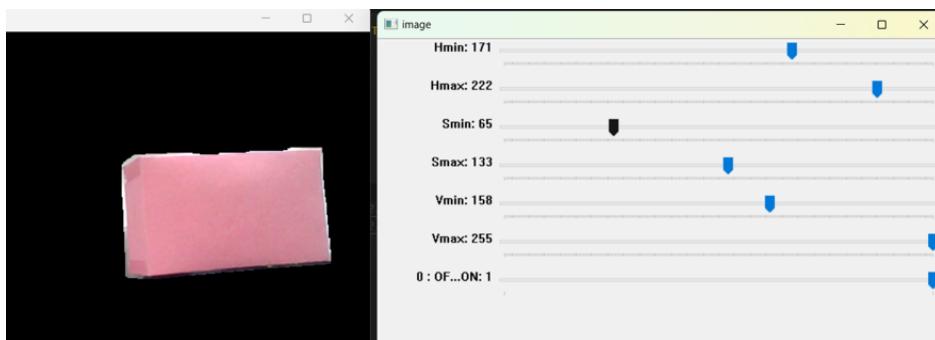
در فرایند تشخیص موانع، یکی از روش های مورد استفاده، تشخیص رنگ مowanع است. استفاده از تشخیص رنگ به عنوان یک ابزار قدرتمند در مسائل تشخیص موانع، به ربات ها امکان می دهد محیط خود را بادقت بالا کنترل کنند و به تصمیم گیری های بهتری دست یابند. برای انجام این کار، ابتدا باید محدوده رنگی موانع را مشخص کرد. به منظور این کار، از یک قطعه برنامه پایتون با استفاده از کتابخانه OpenCV استفاده می شود. همان طور که در شکل ۱-۲ مشاهده می شود این قطعه برنامه به شما اجازه می دهد تا با تصاویری از محیط خود، محدوده های رنگی (مثل محدوده رنگ قرمز، آبی، زرد و...) که موانع دارای آن رنگ هستند را تعیین کنید.

بدین منظور چند نوار برای هر یک از سه پارامتر در روش HSV طراحی شده است که با تنظیم بازه های بالا و پایین برای آنها می توان رنگ موردنظر خود را از محیط ایزوله و آن را تشخیص داد.



شکل ۱-۲

همچنین با استفاده از روشی تحت عنوان Morphological Transformations که از مفهوم کرنل بهره می گیرد، می توان نویزهای موجود در تصویر را از بین برد.



شکل ۲-۲

تصویر ۲-۲ حاصل استفاده از این روش بر روی تصویر ۱-۲ می‌باشد. برای این ویژگی نیز یک نوار در انتهای قرار داده شده که با دستور روشن و خاموش می‌توان آن را فعال و غیرفعال کرد. از این محدوده‌های رنگی به عنوان پارامترهای ورودی در برنامه اصلی استفاده خواهد شد که جزئیات آن در قسمت‌های آتی بیان می‌گردد. در نتیجه اطلاعات تشخیص موانع از این مرحله استخراج شده و ربات می‌تواند بر اساس آن‌ها تصمیمات خود را در مورد حرکت و انجام وظایف مختلف بگیرد.

۲-۳-۲ تشخیص لبه‌ها

در مرحله تشخیص لبه‌ها، از تکنیک‌های پردازش تصویر برای تعیین مرزها و لبه‌های اشیا (موانع) استفاده می‌شود. این مرحله بهوسیله کانتورها آغاز می‌شود که در مرحله قبلی به دست آمده‌اند. با استفاده از کانتورها، می‌توان مرزهای شیء را تشخیص داد و اطلاعاتی مانند عرض و ارتفاع شیء (موانع) را به دست آورد. این اطلاعات می‌توانند برای موارد مختلف مورداستفاده قرار گیرند. به عنوان مثال می‌توانند یک معیار برای تعیین فاصله از شیء موردنظر استفاده شوند. در نهایت با استفاده از بازه‌های تعیین شده برای رنگ موانع و لبه‌ها قادر به تشخیص موانعی از آن دست در محیط باشیم.

نتیجه تشخیص موانع در فصل آخر بررسی شده است. شکل ۲-۵ نمونه‌ای از تشخیص مانعی به رنگ صورتی است.

۳-۳-۲ شبکه‌کد و تشریح جزئیات آن

در زیر شبکه‌کد این بخش مختصررا قرار داده شده و پس از آن به تشریح کامل‌تر آن پرداخته خواهد شد.

```
#####
# Pseudocode #####
#####
Start
```

```

1. import necessary libraries
2. Determine boundries of colors
3. Define function to detect colors
    3.1.Create mask with boundries
    3.2.Convert frame to hsv representations
    3.3.Use threshold for mask
4. Find all contours
5. Choose biggest object among contours detected
6. Return width and Heigth of object
7. Draw shape around obstacle for visualization
End

```

۴-۲ الگوریتم درخت جستجوی تصادفی

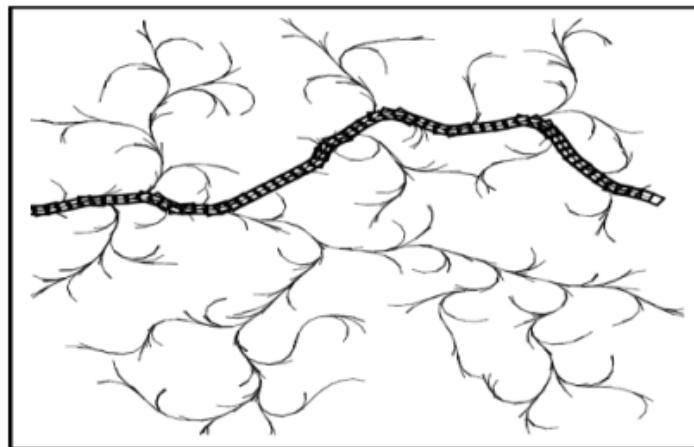
در بخش ۱-۵ مروری بر الگوریتم های مسیریابی انجام گرفت و به اهمیت این موضوع در رباتیک اشاره گردید. پیشرفت های قابل توجه در الگوریتم های برنامه ریزی مسیر به ربات های خودکار امکان می دهد تا به طور کارآمد در محیط های پیچیده حرکت کنند. از میان این الگوریتم ها، الگوریتم درخت جستجوی تصادفی^۱ یا به اختصار RRT به عنوان یک رویکرد نوآورانه شناخته می شود که به حل چالش یافتن مسیر های قابل اجرا در فضاهای تنظیم با بعد بالا می پردازد.

این الگوریتم با استفاده از جستجوی تصادفی فضای پیکربندی به حل برنامه مساله ریزی مسیر می پردازد. در این روش، ریشه درخت در پیکربندی اولیه قرار دارد. ابتدا یک نقطه به صورت تصادفی با توزیع یکنواخت^۲ انتخاب می شود. اگر نقطه انتخاب شده به فضای موائع تعلق داشته باشد، نقطه دیگری انتخاب می شود. با انتخاب نقطه از فضای آزاد، در راستای خط واصل بین این نقطه و ریشه درخت، نقطه جدیدی تحت عنوان X_{new} که به اندازه گام ربات از ریشه فاصله دارد، انتخاب می شود. اگر بتوان X_{new} را با پاره خطی که تماماً به فضای آزاد تعلق دارد، به ریشه وصل کرد، X_{new} به عنوان گره جدید به درخت اضافه می شود.

¹Rapidly Exploring Random Tree

²Uniform distribution

نمایی از درخت توسعه یافته با این روش در شکل ۳-۲ نمایش داده شده است.



[۱۸] شکل ۳-۲: الگوریتم RRT

۱-۴-۲ شبیه‌سازی دوبعدی با پایتون

۱-۱-۴-۲ شبکه کد و تشریح جزئیات برنامه

در زیر شبکه کد این بخش مختصراً قرار داده شده و پس از آن به تشریح کامل‌تر آن پرداخته خواهد شد.

```
#####
# Pseudocode #####
#####

Start

1. import necessary libraries
2. import classes from modules which we wrote before
3. Define Start and goal points and obstacles dimention
4. Define Main function and write below details in it
    4.1 Create Map and Graph object with init method
    4.2 Create obstacles and draw on Map
    4.3 while(not graph."path_to_goal()" is True):
        use bias or expand approach for optimization
        update graph by drawing nodes and lines
        draw path to goal untill last node
```

```

5. call Main function and define a Flag

6. if(last node reached to goal point area):

    assign True to our Flag

    else:

        assign False to our Flag

        Main function will call again

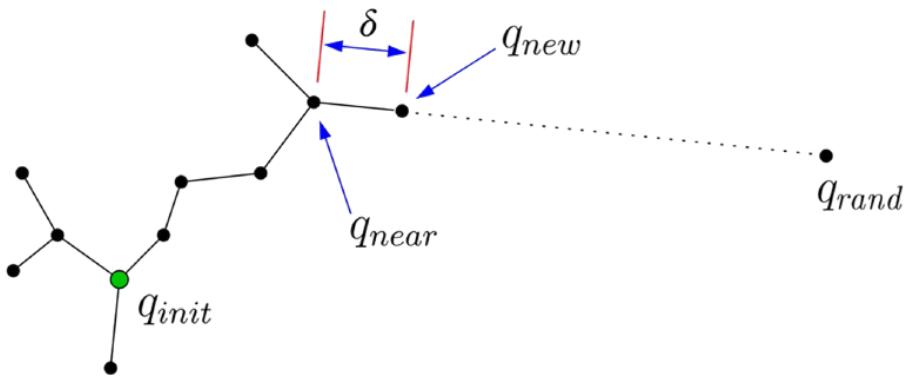
End

```

همان‌طور که در شبکه‌کد بالا مشخص است برنامه تا زمانی که آخرین گره گراف به ناحیه نزدیک نقطه هدف نرسد ادامه خواهد داشت. جزئیات و نحوه تولید مسیر در قطعه برنامه‌ای در پیوست موجود است. نحوه عملکرد توابع اصلی برنامه برای تولید مسیر به شرح زیر است:

۱. تولید یک گره تصادفی با فاصله مشخص از گره شروع
۲. بررسی معتبر بودن گره جدید (معتبر بودن به معنای آنکه آیا گره جدید بر روی موقعیت یکی از موانع نبوده باشد).
۳. در صورت معتبر بودن گره جدید با فواصل از پیش تعیین شده توسط چند قدم یال‌هایی برای گراف ایجاد می‌شود.
۴. اگر در طی این چند قدم یکی از یال‌ها با یکی از موانع برخورد داشته باشد، قدم دیگری برای رسیدن به گره جدید انتخاب می‌شود. در غیر این صورت برنامه تا رسیدن به گره جدید ادامه داده و دوباره به مرحله اول (تولید گره تصادفی) برگشت داده خواهد شد.
۵. گراف این روند را تا زمانی که به نقطه هدف نزدیک نشود ادامه داده و در نهایت با توجه به دانستن موقعیت این نقطه، مسیر بهترین مسیر از بین مسیرهای ساخته شده توسط گره‌های خود را انتخاب می‌کند.

برای درک بهتر مراحل بالا می توان شکل ۴-۲ را در نظر گرفت. همچنین قابل ذکر است که برای هر یک از مراحل بالا یک یا دو روش^۱ در مژول نوشته شده در نظر گرفته شده است و این روشها با یکدیگر در ارتباط بوده و یا به صورت تو در تو^۲ در یکدیگر استفاده شده‌اند.



شکل ۴-۲: مراحل تولید گره جدید و تشکیل مسیر [۶]

نتایج حاصل از این شبیه‌سازی در شکل ۱-۵ قابل مشاهده می‌باشد.

۲-۱-۴-۲ ادغام الگوریتم با قابلیت‌های ربات به منظور پیاده‌سازی

همان‌طور که مشخص است استفاده از الگوریتم پیشنهادی به تنها‌یی مفید واقع نخواهد شد و بایستی از قابلیت‌های موجود در ربات برای بهینه‌سازی این روش استفاده شود. از قابلیت‌های ربات که می‌تواند به کار آید، دوربین استفاده شده می‌باشد. با استفاده از دوربین موجود موانع و فاصله‌ی ربات تا موانع تشخیص داده شده و سپس با توجه به این اطلاعات ربات تصمیم می‌گیرد تا نقاط تصادفی خود را در کدام جهت تولید کند و یا در کدام جهت به تولید ادامه مسیر نپردازد. بدین منظور در بخش‌های آتی به تشریح موارد ذکر شده پرداخته خواهد شد.

Localization ۵-۲

مکان‌یابی یکی از جنبه‌های حیاتی در علم رباتیک است که به ربات‌ها امکان می‌دهد تا مکان و موقعیت دقیق خود را در محیط تعیین کنند. برای ربات‌ها، داشتن اطلاعات صحیح و دقیق در مورد مکان خود بسیار مهم است تا بتوانند به درستی عمل کرده و وظایف مختلفی را انجام دهند. در این بخش به مکان‌یابی در ربات عنکبوتی چهارپا پرداخته خواهد شد. بدین منظور به انتخاب یکی از روش‌های نوین در این حوزه دست زده شد. در سال‌های اخیر برخی از آزمایشگاه‌های رباتیک در دانشگاه‌های معتبر جهان که از میان آنها می‌توان به آزمایشگاه رباتیک اپریل^۳ در دانشگاه میشیگان اشاره کرد، روش‌هایی

¹Method

²Nested

³APRIL robotics lab

برای مکان یابی با استفاده از تگ هایی مشابه به تگ های معروف که با نام کیو آر کد^۱ شناخته شده اند، معرفی کرده اند.



شکل ۲-۵: آزمایشگاه اپریل میشیگان [۷]

۱-۵-۲ استفاده از تگ

۱-۱-۵-۲ مفاهیم اصلی

نحوه استفاده از تگ های این چنینی بر اساس روابط مثلثاتی و هندسه اقلیدسی موجود در فضا است و با داشتن اطلاعاتی همچون فاصله کانونی دوربین و مرکز آن، اطلاعاتی همچون فاصله در هر سه محور و یا زاویه دوربین نسبت به تگ را برمی گردانند. بدین منظور از یکی از مازول های موجود در کتابخانه OpenCV که Aruco نام دارد، استفاده خواهد شد.

۲-۱-۵-۲ شبکه کد و تشریح جزئیات برنامه

در زیر شبکه کد این بخش مختصراً قرار داده شده و پس از آن به تشریح کامل تر آن پرداخته خواهد شد.

```
##### Pseudocode #####
Start
1. import necessary libraries
```

^۱QR code

```

2. Get camera access with IP

3. Define mtx matric and dist array

4. while(camera is available and send validate frame):

    Convert frame to grayscale
    Use gray frame as Aruco methods
    Output of methods include corners
    { Find Transition and Rotation matrices
        by predefined variables and corners }

    plot frame in a monitoring window

5. Print Transition and Rotation matrices

6. Release camera

7. Destroy all monitoring windows

End

```

مکان‌یابی در رباتیک به وسیله مختصات مکانی (مثلًاً نقاط (x, y, z) یا مختصات خروجی (طول، عرض، ارتفاع) صورت می‌پذیرد. این اطلاعات توسط تگ‌های Aruco به ربات ارائه می‌شود. این تگ‌ها اطلاعاتی را شامل می‌شوند برای مثال سه زاویه (roll, pitch, yaw) برای جهت‌یابی و فاصله‌ها در سه محور مختلف می‌شوند. برای دقیق‌تر در محاسبات، ما یک ماتریس را تعریف می‌کنیم که حاوی فاصله‌های مرکزی دوربین و فوکوس آن است و این ماتریس را به کد مکان‌یابی ارائه می‌دهیم. این اطلاعات به ربات امکان می‌دهند تا موقعیت و جهت خود را با دقیق‌تری تعیین کنند و وظایف خود را انجام دهند.

```

import numpy as np
import cv2
import PIL
import os
from cv2 import aruco
from mpl_toolkits.mplot3d import Axes3D
import matplotlib.pyplot as plt
import matplotlib as mpl

```

```
import pandas as pd

BASE_URL = "http://172.21.232.26"
camera = cv2.VideoCapture('http://172.21.232.26:81/stream')

mtx = np.array([[1.78952655e+03, 0.00000000e+00, 9.69572430e+02],
               [0.00000000e+00, 1.78952655e+03, 5.64872516e+02],
               [0.00000000e+00, 0.00000000e+00, 1.00000000e+00]])

dist = np.array([[5.33659854e+00], [-1.67904382e+02], [3.32943561e-03],
                [-4.67385863e-03], [9.75622127e+02], [5.14691206e+00],
                [-1.66105367e+02], [9.69643912e+02], [0.00000000e+00],
                [0.00000000e+00], [0.00000000e+00], [0.00000000e+00],
                [0.00000000e+00], [0.00000000e+00]])

dist.reshape(1,14)

while 1 :
    try:
        _, frame = camera.read()

        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

        aruco_dict = aruco.Dictionary_get(aruco.DICT_6X6_250)

        parameters = aruco.DetectorParameters_create()

        corners, ids, rejectedImgPoints = aruco.detectMarkers(gray,
                                                               aruco_dict,parameters=parameters)

        # SUB PIXEL DETECTION

        criteria = (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER
                    ,
                    100,0.0001)

        for corner in corners:

            cv2.cornerSubPix(gray, corner, winSize=(3, 3),
                             zeroZone=(-1, -1), criteria=criteria)
```

```
frame_markers=aruco.drawDetectedMarkers(frame.copy(),corners,
                                         ids)

size_of_marker = 0.0285 # side lenght of the marker in meter

rvecs, tvecs, _ = aruco.estimateSingleMarkers(corners,
                                                size_of_marker,mtx, dist)

length_of_axis = 0.1

#print("hi")

imaxis = aruco.drawDetectedMarkers(frame.copy(), corners, ids)
#print(imaxis)

for i in range(len(tvecs)):
    imaxis = aruco.drawAxis(
        imaxis, mtx, dist, rvecs[i], tvecs[i], length_of_axis)

cv2.imshow("hi",frame)

if (cv2.waitKey(1) & 0xFF) == ord('q'):
    break

cv2.imshow("hii", frame_markers)
cv2.imshow("hiii", imaxis)

data = pd.DataFrame(data=rvecs.reshape(len(rvecs), 3),
                     columns=["tx", "ty", "tz"],
                     index=ids.flatten())

data.index.name = "marker"

data.sort_index(inplace=True)

print(data)

except:

    print("an error occured in while loop")

camera.release()
```

```
|| cv2.destroyAllWindows()
```

باتوجه به شبکه کد و توضیحات ارائه شده، نتایج مطلوب برای مکان‌یابی ربات به دست آمد. جزئیات بیشتر آن در بخش ۴-۱-۵ قابل مشاهده است.

۶-۲ جمع بندی

در این فصل با تشریح مفاهیمی از تصویر و پردازش آن و کاربردهای آن در زمینه‌های مختلف همچون رباتیک آغاز و با بررسی الگوریتم درخت جستجوی تصادفی و شبیه‌سازی آن در محیط پایتون ادامه یافت. همچنین در انتهای آن به بررسی مفهومی بسیار مهم به نام مکان‌یابی که برای پیاده‌سازی عملی الگوریتم بر روی ربات در محیط آزمایشگاهی ضروری است اشاره و از راهکاری نوین که در سال‌های اخیر مورد توجه قرار گرفته است، استفاده شد.

فصل سوم

نرم افزار و ارتباطات

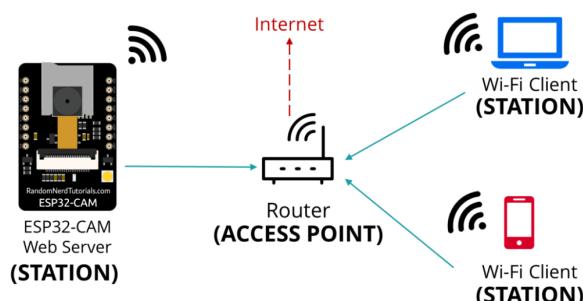
۱-۳ ارتباطات و سرور

۱-۱-۳ اتصال بیسیم دوربین به سرور

۱-۱-۳ مفاهیم اصلی

در این بخش به تبیین مفاهیمی از شبکه^۱ که مورد استفاده قرار گرفته اند و تشریح فرآیندهای انجام شده برای برقراری ارتباط بین ماژول ESP32 CAM و سرور پرداخته شده است. در شبکه‌های اینترنتی دو مفهوم پرکاربرد با کلیدواژه نقطه دسترسی^۲ و مُد^۳ ایستگاهی^۴ (یا به اختصار STA) وجود دارد. طبق تعریف، نقطه دسترسی [۵] یک دستگاه سخت افزاری شبکه است که به سایر دستگاه‌ها اجازه می‌دهد به یک شبکه سیمی متصل شوند و به عنوان یک دستگاه مستقل، ممکن است یک اتصال سیمی به یک روتر^۶ داشته باشد. اگرچه در یک روتر بی‌سیم، می‌تواند جزء جدایی ناپذیر خود روتر نیز باشد.

احتمالاً تجربه متصل کردن تلفن همراه و یا سایر وسایل الکترونیکی خود به مودم خانگی را داشته اید. در این حالت تلفن همراه در مُد ایستگاهی جهت دریافت آی پی^۵ از مودم و همچنین مودم نیز در حالت نقطه دسترسی قرار گرفته است. به همین طریق ماژول ESP32-CAM نیز به عنوان یک دستگاه الکترونیکی و به دلیل نوع طراحی و قابلیت ارتباط بیسیم، می‌تواند به مودم و حتی تلفن همراه در حالت نقطه دسترسی متصل گردد.



شکل ۳: نحوه ارتباط اجزای شبکه [۶]

۲-۱-۳ شبکه و تشریح جزئیات برنامه

در ادامه، به طور دقیق‌تر، فرآیند اتصال ماژول به مودم با استفاده از برنامه نوشته شده در محیط آردوبینو را تشریح خواهیم کرد. سپس نحوه استفاده از کتابخانه‌های ارتباط بیسیم برای اتصال به شبکه و اخذ

¹Network

²Access Point

³Mode

⁴Station

⁵Router

⁶Internet Protocol address

آدرس پروتکل اینترنت (IP) به عنوان یک مشخصه‌ی مهم برای دستگاه خواهد آمد.

```
#####
# Pseudocode #####
#####

Start
1. Import necessary libraries
2. Define SSID and PASSWORD of router
3. Define some configs for camera
4. Start serial port
5. Try to connect module to router
6. if (there are no error):
    Print IP which assigned to module in serial monitor
End
```

در ابتدا به نحوه تنظیم پارامترهای شبکه اعم از نام شبکه (SSID) و رمز عبور (Password) اشاره می‌کنیم. نخست باید مدل دوربین با توجه به مژول مورد استفاده در برنامه تعريف می‌شود. در این بین دوربین OV2640 در دسته AI THINKER قرار می‌گیرد. سپس نام شبکه و رمز عبور مودم مورد استفاده را در برنامه قرار داده و به تنظیم پارامترهای مهمی در بخش Setup در محیط آردوبینو پرداخته می‌شود. نخست مشخصه Buad rate را که بیانگر تعداد تغییرات در سطح سیگنال در یک ثانیه است، برابر ۱۱۵۲۰^۱ قرار داده شده و سپس پارامترهای تصویر اعم از اندازه تصویر^۱، کیفیت تصویر (Jpeg quality) و... تنظیم گردید.

```
1 #include "esp_camera.h"
2 #include <WiFi.h>
3 #include "camera_pins.h"
4
5 #define CAMERA_MODEL_AI_THINKER // Has PSRAM
```

¹Frame size

²Jpeg quality

```

6 const char* ssid = "IRLab";
7 // The password is hide for personal reason :)
8 const char* password = "#####";
9 void startCameraServer();
10 void setup() {
11     Serial.begin(115200);
12     Serial.setDebugOutput(true);
13     camera_config_t config;
14     if(psramFound()){
15         config.frame_size = FRAMESIZE_QXGA; // UXGA
16         config.jpeg_quality = 10;
17         config.fb_count = 2;
18     } else {
19         config.frame_size = FRAMESIZE_QVGA; // SVGA
20         config.jpeg_quality = 12;
21         config.fb_count = 1;
22     }

```

در ادامه پیکربندی تعیین شده برای دوربین بررسی گردیده و در صورت عدم وجود خطا در پیکربندی اتصال به روتر شروع می‌گردد. همانطور که در قطعه کد زیر مشخص است با قرار دادن دستورات چاپ در خطوط مختلف برنامه از اتصال کامل ماژول به روتر اطمینان حاصل می‌گردد.

```

1 // camera init
2 esp_err_t err = esp_camera_init(&config);
3 if (err != ESP_OK) {
4     Serial.printf("Camera init failed with error 0x%x", err);

```

```

5      return; }

6  sensor_t * s = esp_camera_sensor_get(); 

7 // drop down frame size for higher initial frame rate

8 s->set_framesize(s, FRAMESIZE_QVGA); // QVGA

9 WiFi.begin(ssid, password);

10 while (WiFi.status() != WL_CONNECTED) {

11   delay(500);

12   Serial.print(".");

13   Serial.println("WiFi connected");

14 void startCameraServer();

15 Serial.print("Camera Ready! Use 'http://");

16 Serial.print(WiFi.localIP());

17 Serial.println("' to connect");

18 }

19 void loop() {}

```

در انتهای نیز آدرس محلی^۱ که روتر به ماژول اختصار می‌دهد، در بخش سریال مانیتور^۲ آردینو چاپ می‌گردد.

¹LocalIP

²Serial Monitor

Handlers ۲-۱-۳**۱-۲-۱-۳ نحوه کارکرد سرور و نقش هندرلرها**

در سامانه‌های وب، ارتباط و تبادل اطلاعات بین دستگاه‌ها و سرور^۱ از طریق ارسال درخواست^۲ و دریافت پاسخ^۳ به وسیله پروتکل‌های HTTP و HTTPS و... صورت می‌گیرد. وقتی که یک دستگاه یا مرورگر اینترنتی درخواستی به سرور ارسال می‌کند، این درخواست اطلاعاتی از قبیل نوع عملیات (DELETE، PUT، POST، GET و...) و آدرس مورد نظر (URL)^۴ را شامل می‌شود. سرور در این زمان تجزیه و تحلیل درخواست را انجام می‌دهد و با توجه به محتوا و ماهیت درخواست، یک منطق مناسب را فعال می‌سازد. این عملیات‌ها ممکن است شامل دسترسی به پایگاه داده، پردازش اطلاعات، تولید پاسخ‌های مناسب و دیگر عملیات باشند.

در اینجا نقش هندرلر^۵‌ها به عنوان یکی از اجزای اصلی سیستم به چشم می‌خورد. هندرلرها مسئول پردازش درخواست‌ها و انجام عملیات‌های مختلف میان سرور و دستگاه‌ها هستند. هندرلرها اطلاعات درخواست را تجزیه و تحلیل می‌کنند و بر اساس آن‌ها عملیات‌های مورد نیاز را انجام می‌دهند. به عبارت دیگر می‌توان گفت که هندرلرها، ارتباط بین دستگاه‌ها و سرور را به صورت منظم و سازماندهی شده‌ای صورت داده و در نتیجه به بهبود عملکرد و کارایی سامانه کمک می‌کند.

۲-۲-۱-۳ شبکه کد و تشریح جزئیات برنامه

در زیر شبکه کد^۶ این بخش مختصرًا قراره داده شده و پس از آن به تشریح کامل‌تر آن پرداخته خواهد شد.

```
#####
# Pseudocode #####
#####

Start

1. Add some handlers definition to predefined ones
2. Declaration of "current_color_uri" with "/currentcolor" path
3. Define same configs for camera
4. if (there is any request):
    Check it with all handlers
```

¹Server

²Request

³Respond

⁴Uniform Resource Locator

⁵Handler

⁶Pseudo code

```

    if(any handler were called (except "current_color_uri")):
        respond properly
End

```

در قطعه برنامه زیر چند هندر برای مدهای مختلف احتمالی تعریف شده است. برخی از آنها برای ماژول ESP32-CAM از پیش تعریف شده اند. مهمترین هندری که با هدف ارسال و مانیتورینگ اطلاعات بین سرور و ماژول و به طبع آن برد STM32 پیاده سازی شد، current_color_uri نام دارد. برای این هندر مقدار uri که بیانگر مسیر^۱ در آدرس است، /currentColor تعریف شد. تابع هندر آن در ادامه بطور کامل تر تعریف شده است. هدف از تعریف این هندر آن است که درخواست ارسال شده به سرور که حاوی اطلاعاتی اعم از رنگ مانع تشخیص داده شده توسط ربات، فاصله ربات تا مانع و... است را دریافت کرده و این اطلاعات از طریق ارتباط USART، بصورت برخط و با کمترین تاخیر به برد STM32 فرستاده شود.

```

1 void startCameraServer(){
2     httpd_config_t config = HTTPD_DEFAULT_CONFIG();
3     httpd_uri_t index_uri = {
4         .uri      = "/",
5         .method   = HTTP_GET,
6         .handler  = index_handler,
7         .user_ctx = NULL
8     };
9     httpd_uri_t capture_uri = {
10        .uri      = "/capture",
11        .method   = HTTP_GET,
12        .handler  = capture_handler,
13        .user_ctx = NULL
14    };
15     httpd_uri_t stream_uri = {
16        .uri      = "/stream",
17        .method   = HTTP_GET,
18        .handler  = stream_handler,
19        .user_ctx = NULL
20    };
21 }

```

^۱Path

```

15 httpd_uri_t current_color_uri = {
16     .uri      = "/currentColor", .method   = HTTP_GET,
17     .handler = current_color_handler, .user_ctx = NULL
18 };
```

در ادامه چند کانفیگ برای دوربین تنظیم شده که به دلیل طولانی بودن در بخش زیر ذکر نشده‌اند. سپس با بررسی یک شرط، در صورت ارسال موفقیت‌آمیز درخواست از طرف سیستم، تمامی هندرلرهای کمی‌شوند تا در صورت نیاز پاسخ مناسب را به آن درخواست برگردانند.

```

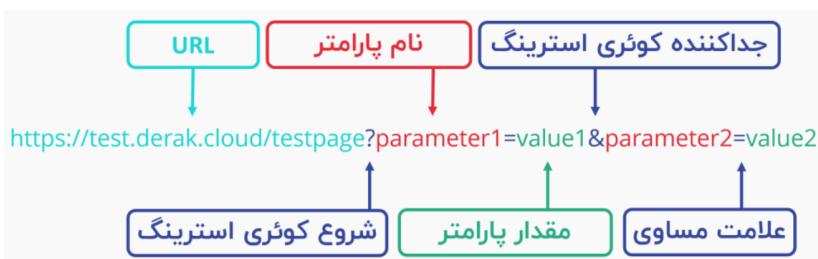
1 // some config set up here
2
3 Serial.printf("Starting webserver on port: %d\n",
4   config.server_port);
5
6 if (httpd_start(&camera_httpd, &config) == ESP_OK) {
7
8     httpd_register_uri_handler(camera_httpd, &index_uri);
9     httpd_register_uri_handler(camera_httpd, &stream_uri);
10    httpd_register_uri_handler(camera_httpd, &capture_uri);
11    httpd_register_uri_handler(camera_httpd, &current_color_uri);
12
13    config.server_port += 1; config.ctrl_port += 1;
14
15    Serial.printf("Starting stream server on port: %d\n",
16      config.server_port);
17
18    if (httpd_start(&stream_httpd, &config) == ESP_OK) {
19
20        httpd_register_uri_handler(stream_httpd, &stream_uri);
21
22    }
23
24 }
```

همانطور که در ۲-۱-۳ ذکر شد، می‌بایست هندر مورد نظر طوری تعریف شود تا اطلاعات مدنظر به بهترین شکل مدیریت شود. به همین جهت برای ارسال اطلاعات از Query String استفاده شده است.

۳-۲-۱-۳ کوئری استرینگ

گاهی اوقات لازم است که وبسایت از وضعیت^۱ ما مطلع باشد و بتواند در همان لحظه، مسیری را طی کند. مثلاً زمانی که قصد داریم یک فرم چند صفحه‌ای را پر کنیم، سرور باید مطلع باشد که ما در صفحه پیش چه کاری انجام دادیم و یا به عنوان مثالی دیگر می‌توان به مراحل یک خرید اینترنتی (اضافه کردن کالاها به سبد خرید، پرداخت آنلاین و...) اشاره کرد.

پروتکل HTTP امکان نگهداری وضعیت را ندارد. پس مکانیزم‌های دیگری به وجود آمدند تا بتوانند این کار را انجام دهند. از جمله‌ی این مکانیزم‌ها می‌توان به نگهداری وضعیت سمت سرور با استفاده از Session و یا نگهداری وضعیت سمت کاربر^۲ با استفاده از کوکی^۳ اشاره کرد. مکانیزم دیگری برای نگهداری وضعیت و انتقال اطلاعات بین صفحات وجود دارد. در این مکانیزم همراه با درخواست‌ها، وضعیت قبلی را نیز از طریق URL جدیدی که فراخوانی می‌شود، به سرور داده می‌شود. به این روش کوئری استرینگ گفته می‌شود. کوئری استرینگ هر مقداریست که بعد از علامت سوال (?) در انتهای URL قرار می‌گیرد که می‌تواند یک یا تعداد بیشتری پارامتر باشد. ساختار کوئری استرینگ را در شکل زیر مشاهده می‌کنید:



[۱۰] شکل ۲-۳: ساختار کوئری استرینگ

همانطور که در شکل ۲-۳ مشخص است آدرس‌های شامل کوئری استرینگ دارای بخش‌های مختلفی می‌باشند.

- URL : این بخش شامل دامنه مورد نظر است. همچنین از اجزای دیگر آن می‌توان به پروتکل، زیردامنه و مسیر اشاره کرد که در نهایت یک آدرس را تشکیل می‌دهد.
- ؟ : ابتدای کوئری استرینگ با این علامت شروع می‌شود و پس از آدرس قرار می‌گیرد.

^۱State

^۲Client

^۳Cookie

- نام پارامتر: در کوئری استرینگ پارامترهای مختلف را می‌بینیم که هر پارامتر یک نام^۱ و یک مقدار^۲ دارد. پس از علامت سوال، نام اولین پارامتر دیده می‌شود.
- =: برای تعیین مقدار یک پارامتر از این علامت استفاده می‌شود و پس از نام پارامتر قرار می‌گیرد.
- &: برای جداسازی پارامترهای مختلف از این علامت استفاده می‌شود. این علامت بین مقدار پارامتر قبلی و نام پارامتر بعدی قرار می‌گیرد.

استفاده از این روش مزایا و معایب متفاوتی دارد که در ذیل مختصرا به آن‌ها اشاره شده‌است. [۱۰] مزایا:

- استفاده ساده
 - سریع‌ترین روش انتقال اطلاعات بین صفحات
 - عدم تحمیل عملیات اضافه به سرویس‌دهنده و در نتیجه هزینه‌ی کم
- معایب:
- اطلاعات، محدود به رشتهداری ساده است (فقط کاراکترهای مجاز)
 - اطلاعات همواره به عنوان یک رشته بازیابی می‌شوند و در صورت نیاز باید آن‌ها را به نوع داده مورد نظر تبدیل کرد.
 - اطلاعات توسط همه قابل مشاهده‌است. برای مواردی که لازم است اطلاعاتی به‌طور مخفی از یک صفحه به صفحه دیگر ارسال و یا بر روی آن حساسیت خاصی از نظر امنیتی وجود دارد، قابل استفاده نیست.
 - کاربران می‌توانند محتویات کوئری استرینگ را تغییر داده و در بعضی موارد باعث ایجاد مشکل شوند.
 - تعداد زیادی از مرورگرها برای طول یک URL محدودیت دارند. بنابراین، نمی‌توان حجم بالایی از اطلاعات را در کوئری استرینگ ذخیره کرد.

¹Key

²Value

٤-٢-١-٣ تابع هندلر

در زیر شیوه کد این بخش مختصر اقرار داده شده و پس از آن به تشریح کامل تر آن پرداخته خواهد شد.

```
#####
# Pseudocode #####
#####

Start
1. Create a buffer
2. if(there are any "get" request with query string):
    assign buffer values to 0 to clear last data
    if(the "key" of query is "color"):
        assign the value to buf
3. Write data with serial to STM32
4. clear buffer
End
```

در برنامه زیر ابتدا یک بافر^۱ با مقدار زیاد انتخاب شده و سپس اگر یک درخواست از نوع GET که دارای کوئری استرینگ باشد برای سرور ارسال شده باشد، آنگاه نام پارامتر آن پارامتر آن بررسی می‌شود و اگر برابر color بوده باشد، مقدار آن پارامتر به عنوان داده معتبر درون بافر ریخته می‌شود. سپس دیتای داخل بافر را از طریق ارتباط UART برای STM32 ارسال و سپس داده داخل بافر برای درخواست بعدی خالی می‌شود.

```
1 static esp_err_t current_color_handler(httpd_req_t *req){
2     char* buf;
3     int buf_len = 300;
4     if (buf_len > 1) {
5         buf = (char*)malloc(buf_len);
6         if (httpd_req_get_url_query_str(req,
7             buf, buf_len) == ESP_OK) {
```

^۱Buffer

```

8      char param[300], dec_param[300]={0};

9      /* Get value of expected key from query string */

10     if (httpd_query_key_value(buf,
11                               "color", param,
12                               sizeof(param)) == ESP_OK)

13     { Serial.write(param); }

14   }

15   free(buf); }

16 const char resp[] = "Done";

17 httpd_resp_send(req, resp, HTTPD_RESP_USE_STRLEN);

18 return ESP_OK;

19 }

```

۳-۱-۳ ارتباط USART

انتقال اطلاعات را می توان به دو روش کلی موازی و سریال انجام داد. در روش انتقال موازی چند بیت اطلاعات به وسیله چند خط انتقال می یابد و در روش انتقال سریال در هر لحظه فقط یک بیت ارسال می شود. در مقایسه بین این دو روش می توان گفت که در روش انتقال موازی به علت انتقال چند بیت اطلاعات در یک لحظه، سرعت آن نسبت به سریال که در یک لحظه فقط یک بیت را انتقال می دهد بیشتر است. همچنین برای فواصل طولانی بکار بردن روش انتقال موازی به علت ازدیاد سیم های ارتباطی، دارای هزینه بالا می باشد؛ بنابراین در فواصل طولانی انتقال به روش سریال مناسب تر است.

فرستنده و گیرنده سریال همزمان و غیر همزمان جهانی یا USART یک ارتباط از نوع انتقال سریال می باشد که امکان ارتباط دوطرفه کامل را می دهد و برای ارتباط بین دو دستگاه مفید می باشد. در حالتی که هیچ ارسال و دریافتی انجام نمی شود یا اصطلاحاً خط انتقال در حالت بیکار (Idle) قرار دارد، سطح ولتاژ مربوط به یک منطقی بر روی خط ارسال قرار می گیرد. تغییر وضعیت از یک به صفر منطقی به معنی شروع ارسال است و گیرنده آماده دریافت اطلاعات می شود. این صفر شدن به مدت یک بیت باید طول بکشد و به آن "بیت شروع" گفته می شود. بعد از آن یک بیت داده به ترتیب از بیت کم ارزش (LSB) به بیت پر ارزش (MSB) ارسال می شود. در نهایت یک بیت برای آزمایش شدن درستی داده ارسال شده، روی خط ارسال قرار می گیرد که بیت "بیت توازن" نام دارد.

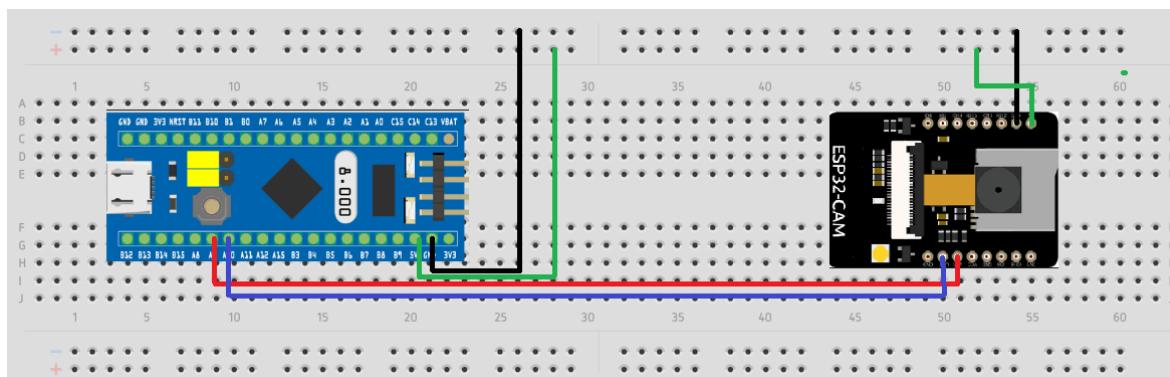
نرخ انتقال های که به صورت قراردادی بین فرستنده و گیرنده در یک ارتباط سریال، مشخص می شود

مقدادیر خاصی می‌توانند داشته باشند که مقدارهای ۱۱۵۲۰۰، ۱۹۲۰۰، ۴۸۰۰، ۹۶۰۰، ۳۸۴۰۰، ۵۷۶۰۰ (بیت بر ثانیه) از معمول ترین این مقدادیرند. در انجام آزمایش‌های عملی بر روی ربات از نرخ ۱۱۵۲۰۰ بیت بر ثانیه استفاده شده است.

۲-۳ اتصال دو برد به یکدیگر

در این بخش، روش اتصال دو برد ESP32 CAM و STM32f103c8t6 به یکدیگر از طریق رابط USART تشریح خواهد شد. رابط USART یک رابط سریالی است که امکان انتقال داده‌ها بین دو دستگاه را فراهم می‌کند. این رابط اغلب برای ارتباط میان میکروکنترلرها و ماژول‌ها استفاده می‌شود. استفاده از این روش برای ارتباط بین دو دستگاه از اهمیت ویژه‌ای در تبادل اطلاعات در پروژه‌های الکترونیکی و رباتیک برخوردار است و تجربه کار با رابطهای سریالی را به ارمغان می‌آورد.

ابتدا به تعیین پایه‌های RX و TX در هر یک از دو برد پرداخته و تنظیمات مورد نیاز در هر دستگاه برای فعال‌سازی رابط USART انجام می‌شود. بدین منظور همانطور که در شکل ۳-۳ مشخص است پایه UOT از ماژول ESP32-CAM که برای دریافت اطلاعات تعییه شده است، به پایه PA9 از برد STM32 متصل می‌شود. همچنین پایه UOT از ماژول ESP32-CAM که برای ارسال اطلاعات تعییه شده است، به پایه PA10 از برد STM32 متصل می‌شود. همچنان که برای ارسال اطلاعات تعییه شده است، به پایه PA10 از برد STM32 که برای دریافت اطلاعات تعییه شده است، متصل می‌باشد. برای تأمین انرژی مدار نیز پایه زمین هر دو مدار و منبع تغذیه یکسان شده و ولتاژ منبع تغذیه بر روی ۵ ولت تنظیم خواهد شد.



شکل ۳-۳: نحوه اتصال دو برد به یکدیگر

۳-۳ جمع‌بندی

در این فصل با توجه به اهمیت انتقال و مدیریت داده بین خود و ربات به طراحی یک سرور و نحوه ارتباط بخش‌های مختلف الکتریکی با یکدیگر پرداخته شد. بدین منظور از روشی با نام کوئری استرینگ استفاده شد. در این روش داده مورد نظر را در دو بخش (نام پارامتر، مقدار پارامتر) منتقل کرده و در سمت دیگر توسط همین دو معیار قابل شناسایی می‌باشند. همچنین برای پاسخ مناسب به درخواست‌های ارسال شونده به سمت سرور توسط توابعی که بطور معمول آن‌ها را هندلر می‌نامند، چند حالت مختلف درخواست و پاسخ طراحی گردید.

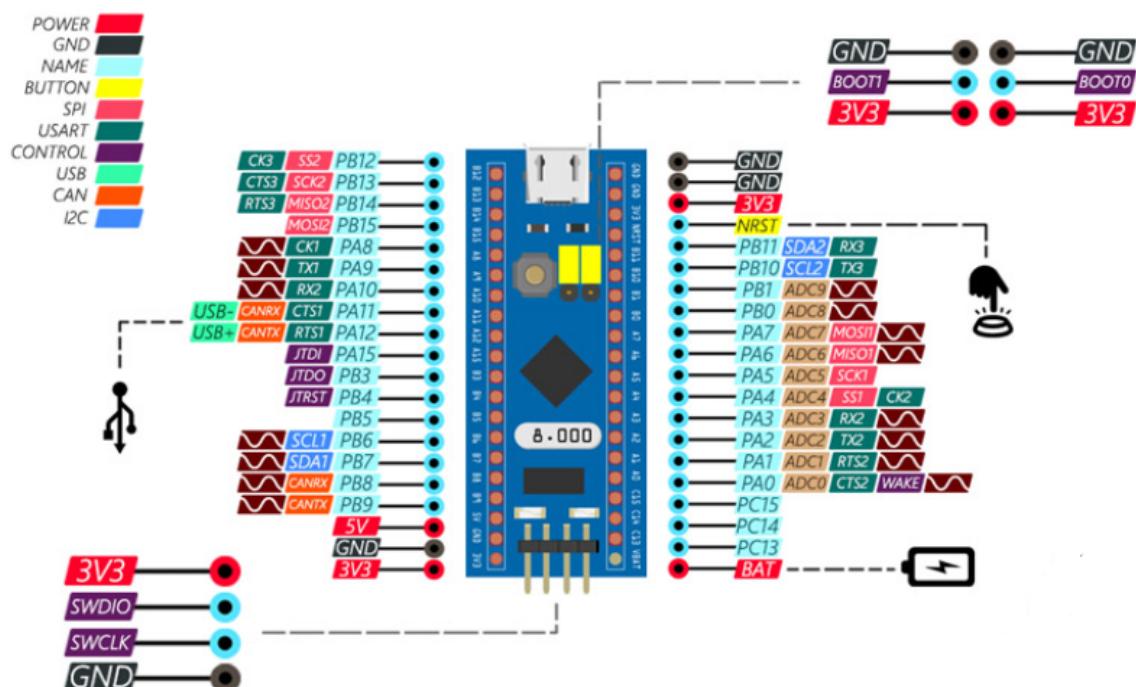
فصل چهارم

سخت افزار الکترونیکی ربات

۱-۴ مقدمه

۲-۴ پردازنده ARM سری F103C8T6

شاید واژه ARM^۱ برایتان آشنا باشد. نوعی معماری برای ساخت انواع پردازنده‌های ۳۲ و ۶۴ بیتی است. از دلایل محبوبیت و استفاده از پردازنده‌هایی با این معماری در انواع سیستم‌های نهفته^۲، می‌توان به قیمت مناسب، سرعت بالاتر، توان مصرفی پایین و... اشاره کرد. عدد بعد از این حروف نشانگر تعداد خطوط اجرایی^۳ پردازنده می‌باشد. یکی از شرکت‌هایی که این معماری را برای ساخت پردازنده‌های خود انتخاب می‌کند، شرکت ST می‌باشد. میکروکنترلرهای این شرکت تحت عنوان STM بلت تنوع بالا و ارائه کتابخانه‌های برنامه‌نویسی کاربردی، بسیار مشهور شده‌اند.



[۱۱] : شکل ۱-۴

¹ Advanced RISC Machine² Embedded Systems³ performance lines

از مشخصات فنی آن می‌توان به موارد زیر اشاره کرد:

- ولتاژ کاری: ۲.۷ تا ۳.۶ ولت

- فرکانس: ماکزیمم ۷۲ مگاهرتز

- تعداد پین‌های ورودی/خروجی: ۳۷

- تعداد پین‌های PWM: ۱۲

- تایمر: ۳+۱

- پروتکل‌های ارتباطی: I2C, SPI, USART, UART

۳-۴ مژول ESP32-CAM

ماژول ESP32-CAM یک مژول کامل است که برای بسیاری از کاربردهای صنعتی و اینترنت اشیاء استفاده می‌شود. همچنین دارای قابلیت‌های متعددی اعم از انواع ارتباطات، ذخیره‌سازی اطلاعات و... بوده و دارای مشخصات فنی زیر می‌باشد:

- ارتباطات: بلوتوث BLE ۲.۴ به همراه WiFi 802.11b. که از بارگذاری تصویر از طریق WiFi پشتیبانی می‌کند.

- اتصالات: SPI، I2C، UART و دارای ۹ پایه GPIO است.

- فرکانس ساعت: تا ۱۶۰ مگاهرتز

- قدرت محاسباتی میکروکنترلر: حداقل ۶۰۰ DMIPS.

- حافظه: ۵۲۰ کیلوبایت SRAM، چهار مگابایت حافظه کارت PSRAM + SD

- دوربین: از دوربین‌های OV2640 پشتیبانی می‌کند که دارای ۲ مگاپیکسل روی سنسور با اندازه آرایه 1622×1200 UXGA پیکسل هستند.

برای اتصال این ماژول به کامپیوتر و برنامه‌ریزی^۱ آن می‌توان از دو روش استفاده کرد:

۱-استفاده از یک مبدل USB به سریال مانند FTDI

۲-استفاده از یک برد دیگر مانند آردوینو^۲



[۱۲] شکل ۴-۴: ماژول ESP32CAM

نکته بسیار مهم اینکه برای برنامه ریزی شدن این ماژول باید حتماً حین برنامه ریزی دو پایه‌ی Io0 و GND به یکدیگر متصل باشند. همچنین پس از تکمیل برنامه ریزی باید اتصال این دو پایه را قطع کرد.

۴-۴ دوربین OV2640

دوربین OV2640 یک سنسور تصویر CMOS با رزولوشن بالا است که توسط شرکت OmniVision تولید می‌شود. این دوربین به ویژه برای برنامه‌های مناسب است که نیاز به تصاویر با کیفیت بالا و ویژگی‌های پیشرفته دارند. از آنجا که این دوربین بسیار محبوب و پرطرفدار است، در بسیاری از پروژه‌های الکترونیک و رباتیک استفاده می‌شود.

¹Programming

²برای برنامه ریزی از طریق ارتباطاتی همچون UART



شکل ۴-۳: دوربین OV2640

ویژگی‌های کلیدی دوربین OV2640 عبارت‌اند از:

- رزولوشن بالا: دوربین OV2640 قابلیت ثبت تصاویر با رزولوشن بالا را دارد. این دوربین قابلیت ثبت تصاویر با رزولوشن حداکثر ۲ مگاپیکسل (1600×1200) را دارد.
- فرمت تصویری متنوع: این دوربین از فرمت‌های مختلف تصویری مانند RGB ، YUV و Raw پشتیبانی می‌کند که این امر به کاربر امکان انتخاب فرمتی مناسب با توجه به نیازهای پروژه را می‌دهد.
- تمرکز اتوماتیک (Auto-Focus): دوربین OV2640 دارای ویژگی تمرکز اتوماتیک بر روی تصاویر است که بهترین تنظیمات فوکوس را برای صحنه‌ها و شرایط مختلف فراهم می‌کند.
- تنظیمات تصویری: این دوربین اجازه تنظیمات مختلف تصویری مانند شدت روشنایی (Exposure) و توازن رنگ (White Balance)، کنترast (Contrast) و... را به کاربر می‌دهد.
- حالت تصویربرداری پیوسته (Continuous Shooting): با امکان ثبت تصاویر به صورت پیوسته، دوربین OV2640 مناسب برای برنامه‌هایی است که نیاز به ثبت تصاویر متوالی دارند.
- حالت‌های مختلف عکسبرداری: این دوربین از حالت‌های مختلف عکسبرداری مانند Single Capture، JPEG Capture و Raw Capture پشتیبانی می‌کند.
- پشتیبانی از ارتباطات: دوربین OV2640 از ارتباطات مختلف مانند I2C ، SPI و UART پشتیبانی می‌کند که این امر ارتباط آسان با میکروکنترلرها و ماژول‌های دیگر را ممکن می‌سازد.
- کاربرد وسیع: دوربین OV2640 به عنوان یک ماژول کوچک و قابل حمل، در بسیاری از پروژه‌های رباتیک، دوربین‌های مدار بسته، دوربین‌های امنیتی، دستگاه‌های پزشکی و سایر برنامه‌های صنعتی و مصرفی مورد استفاده قرار می‌گیرد.

با توجه به ویژگی‌های برتر دوربین OV2640، می‌توان آن را به عنوان یک گزینه مناسب برای پروژه‌های شما در زمینه‌های رباتیک و دید کامپیووتری در نظر گرفت.

۴-۵ سروروموتور

سروروموتورها 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. سروروموتورها معمولاً دارای یک شفت قابل چرخش هستند که از طریق یک سیگنال کنترلی به جلو و عقب حرکت می‌کنند. سروروموتورها به دلیل دقت، سرعت و کاربردهای متعددشان، بخش مهمی از طراحی و ساخت ربات‌های چهارپا را تشکیل می‌دهند.

در ربات‌های چهارپا، سروروموتورها به عنوان موتورهای اصلی برای حرکت پاهای و اندام‌ها عمل می‌کنند. این سروروموتورها معمولاً دقیق و قابل کنترل با سرعت متغیر هستند، که این ویژگی‌ها امکان جابه‌جایی دقیق و پیچیده را فراهم می‌کنند. سروروموتورها 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 ۴-۴: سروروموتور SG90 [۱۴]

۴-۵-۱ روش PWM

۱-۱-۵ مدولاسیون چیست؟

احتمالاً عبارت‌های AM و FM را روی وسایل الکتریکی مانند رادیو و... مشاهده کرده اید. عبارت AM مخفف Amplitude Modulation به معنی مدولاسیون دامنه و عبارت FM مخفف مدولاسیون فاز است. مدولاسیون دامنه، راهی برای ارسال یک سیگنال روی یک حامل است. مدولاسیون اساساً یک راه برای رمزگذاری یک سیگنال بر روی سیگنال حامل است.

به عبارت دیگر مدولاسیون سوار کردن سیگنال مورد نظر بر روی یک سیگنال دیگر است. اینکار به منظور افزایش برد سیگنال و بهره‌وری انتقال انجام می‌شود. به طور کلی، فرایند گنجاندن سیگنال حاوی اطلاعات در سیگنالی دیگر را مدولاسیون می‌نامند.

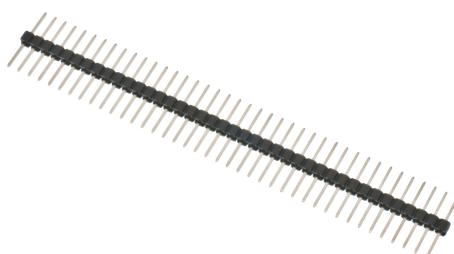
۲-۵-۴ مدولاسیون پهنهای پالس

مدولاسیون پهنهای پالس یا PWM، نوعی سیگنال است که می‌توان آن را توسط یک مدار مجتمع دیجیتال مانند میکروکنترلر تولید کرد. این سیگنال تولیدی یک قطار پالس بوده و یک شکل موج مربعی را تشکیل می‌دهند. به عبارتی در هر زمان مشخص، سیگنال تنها می‌تواند دو وضعیت بالا (High) که معادل ۱ دیجیتالی و یا پایین (Low) که معادل صفر دیجیتالی است را به خود اختصاص دهد. مدت زمانی که سیگنال در وضعیت High قرار دارد، «زمان روشن» (On Time) و مدت زمانی که سیگنال در وضعیت Low است، «زمان خاموش» (Off Time) نامیده می‌شود.

۶-۴ تجهیزات مکانیکی

۱-۶-۴ پین‌هدر

پین‌هدرها به دو دسته نر و ماده تقسیم می‌شوند که در شکل‌های ۵-۴ و ۶-۴ قابل مشاهده است. از پین‌هدرهای نر برای اتصالات متفرقه مخصوصاً در سیم‌کشی استفاده شده است.



شکل ۶-۴: پین‌هدر نر [۱۵]

از پین‌هدرهای ماده به صورت دو خط موازی و بر روی برد هزار سوراخ استفاده شده است تا برد STM32 با پایه‌های خود بر روی آن قرار گیرد.



شکل ۴-۶: پین هدر ماده [۱۶]

۷-۴ جمع بندی

همانطور که در بخش های این فصل بررسی شد، تمامی سخت افزار های استفاده شده در پروژه اعم از الکترونیکی و یا مکانیکی و یا سایر تجهیزات همگی با در نظر گرفتن کاربری تا حدامکان بصورت اقتصادی انتخاب شده و همچنین قابل دسترس در بازار ایران می باشند. از سوی دیگر داشتن شناخت کافی از جزئیات و اطلاعات فنی هریک از سخت افزارها منجر به انتخاب قطعات معرفی شده در کمترین زمان ممکن انجامید.

فصل پنجم

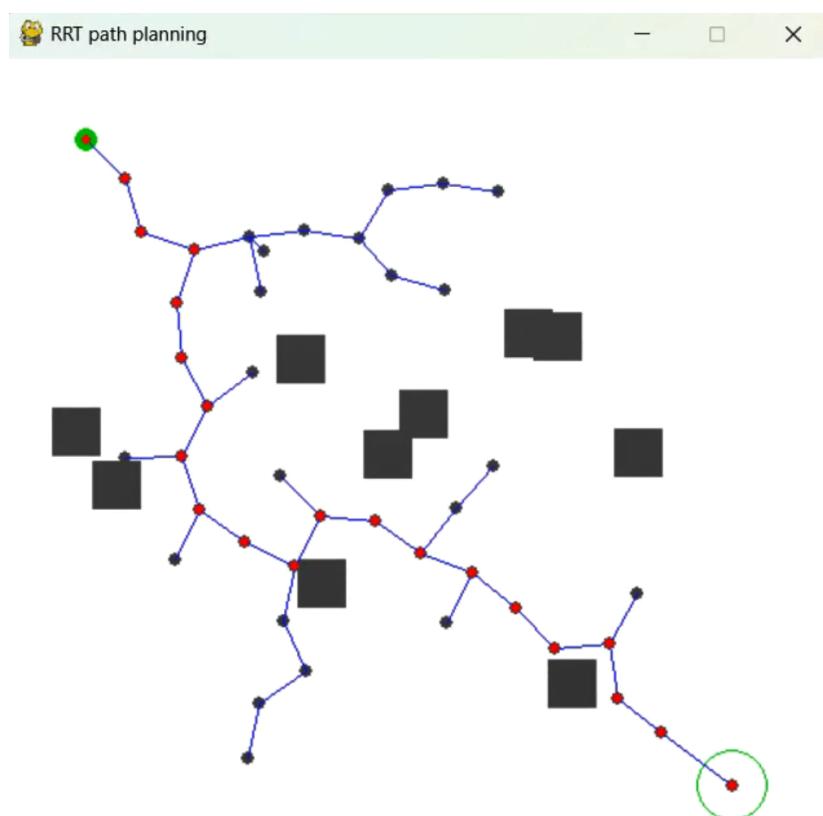
نتایج و پیشنهادات

۱-۵ نتایج

همان‌طور که در فصل‌های پیشین این پایان‌نامه بررسی شد، ربات‌های چهارپا از جمله ربات‌های محبوب در چند سال اخیر هستند که با توجه به تحقیقات روزافرون، کاربردها و بهینگی آنها در حال افزایش است. در این پژوهش نیز تلاش بر آن شد تا با به‌کارگیری انواع تجهیزات و تحلیل چندجانبه ربات طراحی شده، به بهبود عملکرد آن در محیطی شامل موانع با ابعاد و رنگ‌های متفاوت پرداخته شود. همچنین با بهره‌گیری از یک الگوی موجود به نام RRT، شبیه‌سازی آن و ادغام با ویژگی‌های ربات به روشنی مناسب برای برنامه‌ریزی مسیر این ربات دست یافتیم. در بخش‌های پیش رو به بررسی اجمالی برخی از نتایج و رأیه‌پیشنهاداتی بسنده می‌کنیم.

۱-۱-۵ نتایج شبیه‌سازی الگوریتم درخت جستجوی تصادفی

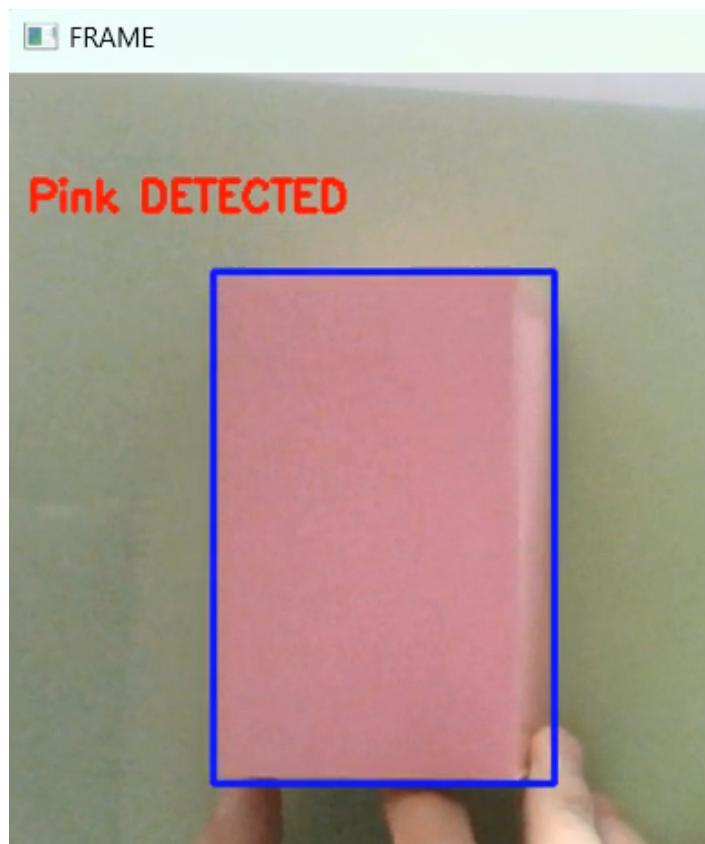
در بخش ۱-۳-۲ مراحل و نحوه عملکرد این شبیه‌سازی شرح داده شد. در ابتدا نتایج حاصل از شبیه‌سازی توسط پایتون قرار داده شده است. همان‌طور که در شکل ۱-۵ قابل مشاهده است، یک محیط با ابعاد مشخص و تعداد مowanع مشخص طراحی و سپس با تعیین نقطه شروع و پایان برای ربات، با استفاده از الگوریتم پیشنهادی ربات توانسته است مسیر خود را بدون برخورد با موانع بیابد.



شکل ۱-۵: شبیه‌سازی الگوریتم RRT

۲-۱-۵ نتیجه تشخیص مانع

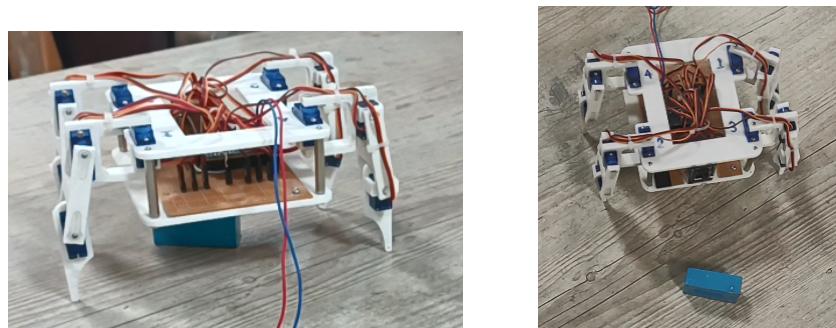
در فصل دوم به بررسی نحوه تشخیص موانع به کمک رنگ‌ها و ابعاد آن‌ها پرداخته شد. در زیر برخی از نتایج آن قابل مشاهده است:



شکل ۲-۵: مانع مکعب مستطیل با رنگ صورتی

۳-۱-۵ نتایج پیاده‌سازی عملی

همان‌طور که در فصل‌های قبل اشاره شد برای پیاده‌سازی عملی ملاحظاتی برای ادغام الگوریتم با ویژگی‌های ربات صورت گرفت و تصویر زیر یکی از نتایج عملی ربات است که با تشخیص مانع آبی‌رنگ و ارتفاع آن تصمیم به عبور از روی این مانع گرفته است.



شکل ۳-۵: تشخیص رنگ آبی

۴-۱-۵ نتایج مکان‌یابی

همان‌طور که نحوه مکان‌یابی ربات در فصل دوم بررسی گردید، با استفاده از تگ‌هایی که اطلاعات فاصله و زاویه دوربین نسبت به آن‌ها را بر می‌گردانند و با دانستن موقعیت هر کدام از تگ‌ها به موقعیت ربات در هر لحظه دست می‌یابیم. نتایج حاصل از این مکان‌یابی در شکل زیر آمده است:

```
None
[[4]]
[[[ 1.56948477  2.46780622 -0.31512746]]]
[[[-0.49942063 -0.27126649  1.09473013]]]
[[4]]
[[[ 1.56811937  2.43274652 -0.2749704 ]]]
[[[-0.50818517 -0.27615465  1.11367207]]]
[[4]]
[[[ 1.56061305  2.38518577 -0.25927664]]]
[[[-0.51372956 -0.27895358  1.12540822]]]
None
```

شکل ۴-۵: زوایا و فواصل از تگ

در این شکل دو ماتریس که هر کدام دارای سه المان هستند و یک عدد که بیانگر شماره تگ رویت شده است، قابل مشاهده است. ماتریس اول بیانگر زاویه تگ با دوربین ربات و ماتریس دوم بیانگر فاصله دوربین و تگ (بر حسب متر) در هر کدام از محورهای مختصات است.

۲-۵ پیشنهادات

در این قسمت با توجه به کارهای انجام شده در این پژوهش و نتایج حاصل شده، پیشنهاداتی برای کارهای آینده، به شرح زیر ارائه می‌گردد:

- می‌توان از دوربین باکیفیت بالاتر استفاده کرد تا در شرایط محیطی متفاوت که میزان نور و... در تشخیص رنگ‌ها تأثیرگذار هستند بدققت بیشتری بررسی شوند.
- استفاده از الگوریتم‌های دیگر مسیریابی و آزمایش عملی بر روی ربات برای دسترسی به روشی بهینه

كتاب نامه

- [1] <https://devicebase.net/en/techman-tm12>.
- [2] https://favpng.com/png_view/robot - delta - robot - robotics - machine - engineering - png/C0KYKNeA.
- [3] <https://www.nytimes.com/2020/02/13/science/farm-agriculture-robots.html>.
- [4] <https://wallpapercave.com/robot-spiders-wallpapers>.
- [5] <https://wiki.redronic.com/handbooks/robotics-handbook/what-is-legged-robots/>.
- [6] <https://www.youtube.com/@Algobotics>.
- [7] <https://april.eecs.umich.edu/software/apriltag>.
- [8] https://en.wikipedia.org/wiki/Wireless_access_point.
- [9] <https://randomnerdtutorials.com/esp32-cam-access-point-ap-web-server/>.
- [10]
- [11] <https://thecafrobot.com/learn/getting-started-w-stm32f103c8t6/>.
- [12] <https://www.srkelectronics.in/product/ov2640-camera-module-esp32-cam/>.
- [13] <https://www.aliexpress.com/i/33042013379.html>.

- [14] <https://www.amazon.in/Robodo-Electronics-Tower-Micro-Servo/dp/B00MTFFAE0?th=1>.
- [15] <https://www.electrokit.com/en/product/pin-header-2-54mm-1x40p-long-pins/>.
- [16] <https://ardushop.ro/en/home/114-40x-female-pin-header.html>.
- [17] Craig, J. J. *Introduction to robotics: Mechanics and control*. Pearson Prentice Hall, 2005.
- نمونه بر مبنای الگوریتم از استفاده با ربات های ریزی برنامه مه. امیری علی، [18] *arrt. در ۱۳۹۳.

پیوست

کد پایتون RRT

ماژول نوشته شده شامل کلاس‌های استفاده شده در فایل اجرایی :

```
import random
import math
import pygame

class RRTMap:

    def __init__(self, start, goal, MapDimensions, obsdim, obsnum):
        self.start = start
        self.goal = goal
        self.MapDimensions = MapDimensions
        self.Maph, self.Mapw = self.MapDimensions

        # window settings
        self.MapWindowName = 'RRT path planning'
        pygame.display.set_caption(self.MapWindowName)
        self.map = pygame.display.set_mode((self.Mapw, self.Maph))
        self.map.fill((255, 255, 255))
        self.nodeRad = 2
        self.nodeThickness = 0
        self.edgeThickness = 1
```

```

        self.obstacles = []
        self.obsdim = obsdim
        self.obsNumber = obsnum

        # Colors
        self.grey = (70, 70, 70)
        self.Blue = (0, 0, 255)
        self.Green = (0, 255, 0)
        self.Red = (255, 0, 0)
        self.white = (255, 255, 255)

    def drawMap(self, obstacles):
        pygame.draw.circle(self.map, self.Green, self.start, self.
                           nodeRad + 5, 0)
        pygame.draw.circle(self.map, self.Green, self.goal, self.
                           nodeRad + 20, 1)
        self.drawObs(obstacles)

    def drawPath(self, path):
        for node in path:
            pygame.draw.circle(self.map, self.Red, node, 3, 0)

    def drawObs(self, obstacles):
        obstaclesList = obstacles.copy()
        while (len(obstaclesList) > 0):
            obstacle = obstaclesList.pop(0)
            pygame.draw.rect(self.map, self.grey, obstacle)

class RRTGraph:
    def __init__(self, start, goal, MapDimensions, obsdim, obsnum):
        (x, y) = start

```

```

        self.start = start
        self.goal = goal
        self.goalFlag = False
        self.maph, self.mapw = MapDimensions
        self.x = []
        self.y = []
        self.parent = []
        # initialize the tree
        self.x.append(x)
        self.y.append(y)
        self.parent.append(0)
        # the obstacles
        self.obstacles = []
        self.obsDim = obsdim
        self.obsNum = obsnum
        # path
        self.goalstate = None
        self.path = []

    def makeRandomRect(self):
        uppercornerx = int(random.uniform(0, self.mapw - self.obsDim))
        uppercornery = int(random.uniform(0, self.maph - self.obsDim))

        return (uppercornerx, uppercornery)

    def makeobs(self):
        obs = []
        for i in range(0, self.obsNum):
            rectang = None
            startgoalcol = True
            while startgoalcol:
                upper = self.makeRandomRect()

```

```

        rectang = pygame.Rect(upper, (self.obsDim, self.obsDim
            ))
        if rectang.collidepoint(self.start) or rectang.
            collidepoint(self.goal):
            startgoalcol = True
        else:
            startgoalcol = False
        obs.append(rectang)
        self.obstacles = obs.copy()
    return obs

def add_node(self, n, x, y):
    self.x.insert(n, x)
    self.y.append(y)

def remove_node(self, n):
    self.x.pop(n)
    self.y.pop(n)

def add_edge(self, parent, child):
    self.parent.insert(child, parent)

def remove_edge(self, n):
    self.parent.pop(n)

def number_of_nodes(self):
    return len(self.x)

def distance(self, n1, n2):
    (x1, y1) = (self.x[n1], self.y[n1])
    (x2, y2) = (self.x[n2], self.y[n2])
    px = (float(x1) - float(x2)) ** 2

```

```

py = (float(y1) - float(y2)) ** 2
return (px + py) ** (0.5)

def sample_envir(self):
    x = int(random.uniform(0, self.mapw))
    y = int(random.uniform(0, self.maph))
    return x, y

def nearest(self, n):
    dmin = self.distance(0, n)
    nnear = 0
    for i in range(0, n):
        if self.distance(i, n) < dmin:
            dmin = self.distance(i, n)
            nnear = i
    return nnear

def isFree(self):
    n = self.number_of_nodes() - 1
    (x, y) = (self.x[n], self.y[n])
    obs = self.obstacles.copy()
    while len(obs) > 0:
        rectang = obs.pop(0)
        if rectang.collidepoint(x, y):
            self.remove_node(n)
            return False
    return True

def crossObstacle(self, x1, x2, y1, y2):
    obs = self.obstacles.copy()
    while (len(obs) > 0):
        rectang = obs.pop(0)

```

```

        for i in range(0, 101):

            u = i / 100

            x = x1 * u + x2 * (1 - u)

            y = y1 * u + y2 * (1 - u)

            if rectang.collidepoint(x, y):

                return True

        return False


    def connect(self, n1, n2):

        (x1, y1) = (self.x[n1], self.y[n1])

        (x2, y2) = (self.x[n2], self.y[n2])

        if self.crossObstacle(x1, x2, y1, y2):

            self.remove_node(n2)

            return False

        else:

            self.add_edge(n1, n2)

            return True


    def step(self, nnear, nrand, dmax=35):

        d = self.distance(nnear, nrand)

        if d > dmax:

            u = dmax / d

            (xnear, ynear) = (self.x[nnear], self.y[nnear])

            (xrand, yrand) = (self.x[nrand], self.y[nrand])

            (px, py) = (xrand - xnear, yrand - ynear)

            theta = math.atan2(py, px)

            (x, y) = (int(xnear + dmax * math.cos(theta)),

                      int(ynear + dmax * math.sin(theta)))

            self.remove_node(nrand)

            if abs(x - self.goal[0]) <= dmax and abs(y - self.goal[1]) <= dmax:

                self.add_node(nrand, self.goal[0], self.goal[1])

```

```

        self.goalstate = nrand
        self.goalFlag = True

    else:
        self.add_node(nrand, x, y)

def bias(self, ngoal):
    n = self.number_of_nodes()
    self.add_node(n, ngoal[0], ngoal[1])
    nnear = self.nearest(n)
    self.step(nnear, n)
    self.connect(nnear, n)
    return self.x, self.y, self.parent

def expand(self):
    n = self.number_of_nodes()
    x, y = self.sample_envir()
    self.add_node(n, x, y)
    if self.isFree():
        xnearest = self.nearest(n)
        self.step(xnearest, n)
        self.connect(xnearest, n)
    return self.x, self.y, self.parent

def path_to_goal(self):
    if self.goalFlag:
        self.path = []
        self.path.append(self.goalstate)
        newpos = self.parent[self.goalstate]
        while (newpos != 0):
            self.path.append(newpos)
            newpos = self.parent[newpos]
        self.path.append(0)

```

```

    return self.goalFlag

def getPathCoords(self):
    pathCoords = []
    for node in self.path:
        x, y = (self.x[node], self.y[node])
        pathCoords.append((x, y))
    return pathCoords

def cost(self, n):
    ninit = 0
    n = n
    parent = self.parent[n]
    c = 0
    while n is not ninit:
        c = c + self.distance(n, parent)
        n = parent
        if n is not ninit:
            parent = self.parent[n]
    return c

def getTrueObs(self, obs):
    TOBS = []
    for ob in obs:
        TOBS.append(ob.inflate(-50, -50))
    return TOBS

def waypoints2path(self):
    oldpath = self.getPathCoords()
    path = []
    for i in range(0, len(self.path) - 1):
        print(i)

```

```

        if i >= len(self.path):
            break

        x1, y1 = oldpath[i]
        x2, y2 = oldpath[i + 1]

        print('-----')
        print((x1, y1), (x2, y2))

        for i in range(0, 5):
            u = i / 5
            x = int(x2 * u + x1 * (1 - u))
            y = int(y2 * u + y1 * (1 - u))
            path.append((x, y))
            print((x, y))

    return path


def makeRandomRect(self):
    uppcornerx = int(random.uniform(0, self.mapw - self.obsDim))
    uppcornery = int(random.uniform(0, self.maph - self.obsDim))
    return (uppcornerx, uppcornery)

def makeobs(self):
    obs = []
    for i in range(0, self.obsNum):
        rectang = None
        startgoalcol = True
        while startgoalcol:
            upper = self.makeRandomRect()
            rectang = pygame.Rect(upper, (self.obsDim, self.obsDim))
            if rectang.collidepoint(self.start) or rectang.
                collidepoint(self.goal):

```

```

        startgoalcol = True

    else:

        startgoalcol = False

    obs.append(rectang)

    self.obstacles = obs.copy()

    return obs

```

فایل اصلی جهت اجرا :

```

import pygame

from RRTbasePy import RRTGraph

from RRTbasePy import RRTMap

import time


def main():

    dimensions =(512,512); start=(50,50); goal=(300,300)

    obsdim=30; obsnum=50; iteration=0; t1=0


    pygame.init()

    map=RRTMap(start,goal,dimensions,obsdim,obsnum)

    graph=RRTGraph(start,goal,dimensions,obsdim,obsnum)

    obstacles=graph.makeobs()

    map.drawMap(obstacles)

    t1=time.time()

    while (not graph.path_to_goal()):

        time.sleep(0.005)

        elapsed=time.time()-t1

        t1=time.time()

        #raise exception if timeout

        if elapsed > 10:

            print('timeout re-initiating the calculations')

```

```

    raise

if iteration % 10 == 0:
    X, Y, Parent = graph.bias(goal)
    pygame.draw.circle(map.map, map.grey, (X[-1], Y[-1]), map.
        nodeRad*2, 0)
    pygame.draw.line(map.map, map.Blue, (X[-1], Y[-1]), (X[
        Parent[-1]], Y[Parent[-1]]),
        map.edgeThickness)

else:
    X, Y, Parent = graph.expand()
    pygame.draw.circle(map.map, map.grey, (X[-1], Y[-1]), map.
        nodeRad*2, 0)
    pygame.draw.line(map.map, map.Blue, (X[-1], Y[-1]), (X[
        Parent[-1]], Y[Parent[-1]]),
        map.edgeThickness)

if iteration % 5 == 0:
    pygame.display.update()
    iteration += 1
map.drawPath(graph.getPathCoords())
pygame.display.update()
pygame.event.clear()
pygame.event.wait(0)

if __name__ == '__main__':
    result=False
    while not result:
        try:
            main()
            result=True

```

```

    except:

        result=False

```

۱- کد Localization

کد مکان یابی برای سنجش صحت پارامترهای دوربین و ارتباط با ربات

```

import numpy as np
import cv2
import PIL
import os
from cv2 import aruco
from mpl_toolkits.mplot3d import Axes3D
import matplotlib.pyplot as plt
import matplotlib as mpl
import pandas as pd

BASE_URL = "http://172.21.232.26"
camera = cv2.VideoCapture('http://172.21.232.26:81/stream')

mtx = np.array([[1.78952655e+03, 0.00000000e+00, 9.69572430e+02],
               [0.00000000e+00, 1.78952655e+03, 5.64872516e+02],
               [0.00000000e+00, 0.00000000e+00, 1.00000000e+00]])

dist = np.array([[5.33659854e+00], [-1.67904382e+02], [3.32943561e-03],
                 [-4.67385863e-03], [9.75622127e+02], [5.14691206e+00],
                 [-1.66105367e+02], [9.69643912e+02], [0.00000000e+00],
                 [0.00000000e+00], [0.00000000e+00], [0.00000000e+00],
                 [0.00000000e+00], [0.00000000e+00]])

dist.reshape(1,14)

while 1 :
    try:

```

```

_ , frame = camera.read()

gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

aruco_dict = aruco.Dictionary_get(aruco.DICT_6X6_250)

parameters = aruco.DetectorParameters_create()

corners, ids, rejectedImgPoints = aruco.detectMarkers(gray,
                                                       aruco_dict, parameters=parameters)

# SUB PIXEL DETECTION

criteria = (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER
            ,
            100, 0.0001)

for corner in corners:

    cv2.cornerSubPix(gray, corner, winSize=(3, 3),
                     zeroZone=(-1, -1), criteria=criteria)

frame_markers=aruco.drawDetectedMarkers(frame.copy(),corners,
                                         ids)

size_of_marker = 0.0285 # side lenght of the marker in meter

rvecs, tvecs, _ = aruco.estimatePoseSingleMarkers(corners,
                                                   size_of_marker,mtx, dist)

length_of_axis = 0.1

#print("hi ")

imaxis = aruco.drawDetectedMarkers(frame.copy(), corners, ids)
#print(imaxis)

for i in range(len(tvecs)):

    imaxis = aruco.drawAxis(
        imaxis, mtx, dist, rvecs[i], tvecs[i], length_of_axis)

cv2.imshow("hi",frame)

```

```
if (cv2.waitKey(1) & 0xFF) == ord('q'):  
    break  
  
cv2.imshow("hii", frame_markers)  
cv2.imshow("hiii", imaxis)  
  
data = pd.DataFrame(data=rvecs.reshape(len(rvecs), 3),  
                     columns=["tx", "ty", "tz"],  
                     index=ids.flatten())  
  
data.index.name = "marker"  
data.sort_index(inplace=True)  
print(data)  
  
except:  
    print("an error occurred in while loop")  
  
camera.release()  
cv2.destroyAllWindows()
```

Abstract

This page is accurate translation from Persian abstract into English.

Key Words:

Write a 3 to 5 KeyWords is essential. Example: AUT, M.Sc., Ph. D.,..