



دانشگاه صنعتی امیرکبیر
(پلی‌تکنیک تهران)
دانشکده مهندسی برق

پایان‌نامه کارشناسی ارشد
گرایش کنترل

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

پایان‌نامه

نگارش
سجاد قدیری

استاد راهنما
دکتر محمداعظم خسروی

شهریور ۱۴۰۲

بِسْمِ اللّٰهِ الرَّحْمٰنِ الرَّحِيْمِ

صفحه فرم ارزیابی و تصویب پایان نامه- فرم تأیید اعضاء کمیته دفاع

در این صفحه فرم دفاع یا تایید و تصویب پایان نامه موسوم به فرم کمیته دفاع- موجود در پرونده آموزشی- را قرار دهید.

نکات مهم:

- نگارش پایان نامه/رساله باید به **زبان فارسی** و بر اساس آخرین نسخه دستورالعمل و راهنمای تدوین پایان نامه های دانشگاه صنعتی امیرکبیر باشد.(دستورالعمل و راهنمای حاضر)
- رنگ جلد پایان نامه/رساله چاپی کارشناسی، کارشناسی ارشد و دکترا باید به ترتیب مشکی، طوسی و سفید رنگ باشد.
- چاپ و صحافی پایان نامه/رساله بصورت **پشت و رو(دورو)** بلامانع است و انجام آن توصیه می شود.



دانشگاه صنعتی امیرکبیر
(پلی‌تکنیک تهران)

به نام خدا

تعهدنامه اصالت اثر

تاریخ: شهریور ۱۴۰۲

اینجانب سجاد قدیری متعهد می‌شوم که مطالب مندرج در این پایان‌نامه حاصل کار پژوهشی اینجانب تحت نظرارت و راهنمایی استاد 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، ترجمه و اقتباس از این پایان‌نامه بدون موافقت کتبی دانشگاه صنعتی امیرکبیر ممنوع است. نقل مطالب با ذکر مأخذ بلامانع است.

سجاد قدیری

امضا

نویسنده مایان نامه، در صورت تایل میتواند برای پیمان نامه خود را به شخص یا اشخاص و یا ارگان خاصی تقدیم نماید.

پاسکزاری

نویسنده پایان نامه می تواند مراتب امتحان خود را نسبت به استاد راهنمای و استاد مشاور و یا دیگر افرادی که طی انجام پایان نامه به نحوی او را یاری و یا با او همکاری نموده اند ابراز دارد.

سجاد قدری

شهریور ۱۴۰۲

چکیده

در این پایان نامه، به بررسی و طراحی یک ربات عنکبوتی چهار پا با استفاده از دوربین به عنوان سامانه‌ای برای تشخیص موانع با رنگ‌های مختلف و برنامه‌ریزی مسیر در محیط‌های پویا و متغیر می‌پردازیم. این سامانه از یک الگوریتم مبتنی بر داده استفاده می‌کند تا با تجزیه و تحلیل تصاویر از دوربین، به تشخیص موانع در مسیر خود بپردازد و برنامه‌ریزی مناسبی را برای جلوگیری از تصادفات و تصادمات انجام دهد.

واژه‌های کلیدی:

کلیدواژه اول، ...، کلیدواژه پنجم (نوشتن سه تا پنج واژه کلیدی ضروری است)

فهرست مطالب

عنوان

صفحه

۱	۱	۱ مقدمه
۲	۱-۱	۱-۱ مقدمه
۲	۱-۱-۱	۱-۱-۱ ربات و انواع آن
۲	۱-۱-۲-۱	۱-۱-۲-۱ ربات‌های سری
۳	۱-۱-۲-۱-۱	۱-۱-۲-۱-۱ ربات‌های موازی
۴	۱-۱-۲-۱-۲-۱	۱-۱-۲-۱-۲-۱ ربات‌های متحرک
۵	۱-۱-۲-۱-۲-۱-۱	۱-۱-۲-۱-۲-۱-۱ ربات عنکبوتی چهارپا
۵	۱-۱-۲-۱-۲-۱-۱-۱	۱-۱-۲-۱-۲-۱-۱-۱ ربات‌های عنکبوتی
۶	۱-۱-۲-۱-۲-۱-۱-۱	۱-۱-۲-۱-۲-۱-۱-۱ پایداری استاتیکی
۷	۱-۱-۲-۱-۲-۱-۱-۱-۱	۱-۱-۲-۱-۲-۱-۱-۱-۱ ربات‌های عنکبوتی چهارپا
۷	۱-۱-۲-۱-۲-۱-۱-۱-۱-۱	۱-۱-۲-۱-۲-۱-۱-۱-۱-۱ تاثیر هوش مصنوعی بر رباتیک
۸	۱-۱-۲-۱-۲-۱-۱-۱-۱-۱	۱-۱-۲-۱-۲-۱-۱-۱-۱-۱ مروری بر الگوریتم‌های مسیریابی
۱۰	۱-۱-۲-۱-۲-۱-۱-۱-۱-۱-۱	۱-۱-۲-۱-۲-۱-۱-۱-۱-۱-۱ اهداف و نوآوری
۱۱	۱-۱-۲-۱-۲-۱-۱-۱-۱-۱-۱-۱	۱-۱-۲-۱-۲-۱-۱-۱-۱-۱-۱-۱ ساختار پایان نامه
۱۱	۱-۱-۲-۱-۲-۱-۱-۱-۱-۱-۱-۱-۱	۱-۱-۲-۱-۲-۱-۱-۱-۱-۱-۱-۱-۱ جمع بندی
۱۲	۲	۲ مسیریابی و الگوریتم‌های پیاده سازی شده
۱۳	۲-۱	۲-۱ مقدمه
۱۳	۲-۱-۱	۲-۱-۱ مقدمه‌ای بر پردازش تصویر
۱۳	۲-۱-۱-۱	۲-۱-۱-۱ پردازش تصویر
۱۳	۲-۱-۱-۱-۱	۲-۱-۱-۱-۱ مقدمه
۱۴	۲-۱-۱-۱-۱-۱	۲-۱-۱-۱-۱-۱ تصویر چیست؟
۱۴	۲-۱-۱-۱-۱-۱-۱	۲-۱-۱-۱-۱-۱-۱ پردازش تصویر چیست؟
۱۵	۲-۱-۱-۱-۱-۱-۱-۱	۲-۱-۱-۱-۱-۱-۱-۱ الگوریتم جستجوی سریع درخت تصادفی
۱۵	۲-۱-۱-۱-۱-۱-۱-۱-۱	۲-۱-۱-۱-۱-۱-۱-۱-۱ مقدمه
۱۶	۲-۱-۱-۱-۱-۱-۱-۱-۱-۱	۲-۱-۱-۱-۱-۱-۱-۱-۱-۱ شیوه‌سازی دوبعدی با پایتون

۱۶	۱-۲-۴-۲ نتایج شبیه‌سازی
۱۷	۲-۲-۴-۲ شبیه‌کد و تشریح جزئیات برنامه
۱۹	۳-۲-۴-۲ ادغام الگوریتم با قابلیت‌های ربات به منظور پیاده‌سازی
۲۰	۵-۲ Localization
۲۰	۱-۵-۲ مقدمه
۲۱	۲-۵-۲ استفاده از تگ
۲۱	۱-۲-۵-۲ مفاهیم اصلی
۲۱	۲-۲-۵-۲ شبیه‌کد و تشریح جزئیات برنامه
۲۴	۶-۲ جمع‌بندی
۲۵	۳ نرم افزار و ارتباطات
۲۶	۱-۳ ارتباطات و سرور
۲۶	۱-۱-۳ اتصال بی‌سیم دوربین به سرور
۲۶	۱-۱-۱-۳ مفاهیم اصلی
۲۷	۲-۱-۱-۳ شبیه‌کد و تشریح جزئیات برنامه
۳۰	۲-۱-۳ سرور و هندرلرها
۳۰	۱-۲-۱-۳ نحوه کارکرد سرور و نقش هندرلرها
۳۰	۲-۲-۱-۳ شبیه‌کد و تشریح جزئیات برنامه
۳۳	۳-۲-۱-۳ کوئری استرینگ
۳۵	۴-۲-۱-۳ تابع هندرلر
۳۷	۳-۱-۳ ارتباط USART
۳۸	۲-۳ اتصال دو برد به یکدیگر
۳۹	۴ سخت افزار الکترونیکی ربات
۴۰	۱-۴ مقدمه
۴۰	۲-۴ پردازنده ARM سری f101c8t6
۴۱	۳-۴ ماژول ESP32-CAM
۴۲	۴-۴ دوربین OV2640
۴۳	۵-۴ سروروموتور

۴۴	PWM روش ۱-۵-۴
۴۴	۱-۱-۵-۴ مدولاسیون چیست؟
۴۴	۲-۱-۵-۴ مدولاسیون پهنهای پالس
۴۵	۴-۶ تجهیزات مکانیکی
۴۵	۱-۶-۴ سایر قطعات متصل کننده
۴۵	۷-۴ جمع بندی
۴۶	۵ جمع بندی و نتیجه‌گیری و پیشنهادات
۴۷	۱-۵ پیشنهادات
۴۸	کتاب‌نامه
۵۰	پیوست
۶۱	۱- کد Localization
۶۴	واژه‌نامه‌ی فارسی به انگلیسی
۶۶	واژه‌نامه‌ی انگلیسی به فارسی

فهرست تصاویر

صفحه

شکل

۳	۱-۱ ربات سری
۴	۲-۱ ربات موازی
۴	۳-۱ ربات کشاورزی
۶	۴-۱ ربات عنکبوتی
۶	۵-۱ پایداری استاتیکی
۷	۶-۱ نسخه‌اولیه طراحی شده
۱۱	۷-۱ نسخه‌اسمبل شده
۱۷	۱-۲ شبیه سازی الگوریتم RRT
۱۹	۲-۲ مراحل تولید گره جدید و تشکیل مسیر
۲۰	۳-۲ آزمایشگاه اپریل میشیگان
۲۶	۱-۳ نقطه دسترسی و مُد ایستگاهی
۳۴	۲-۳ ساختار کوئری استرینگ
۳۸	۳-۳ نحوه اتصال دو برد به یکدیگر
۴۱	۲-۴ ماژول ESP32CAM
۴۲	۳-۴ دوربین OV2640
۴۴	۴-۴ سروموتور SG90

فهرست جداول

صفحه

جدول

فهرست نمادها

نماد	مفهوم
\mathbb{R}^n	فضای اقلیدسی با بعد n
\mathbb{S}^n	کره یکه n بعدی
M^m	خمینه- m -بعدی
$\mathfrak{X}(M)$	جبر میدان‌های برداری هموار روی M
$\mathfrak{X}^1(M)$	مجموعه میدان‌های برداری هموار یکه روی (M, g)
$\Omega^p(M)$	مجموعه p -فرمی‌های روی خمینه M
Q	اپراتور ریچی
\mathcal{R}	تانسور انحنای ریمان
ric	تانسور ریچی
L	مشتق لی
Φ	۲-فرم اساسی خمینه تماسی
∇	التصاق لوی-چویتای
Δ	لاپلاسین ناهموار
∇^*	عملگر خودالحاق صوری القا شده از التصاق لوی-چویتای
g_s	متر ساساکی
∇	التصاق لوی-چویتای وابسته به متر ساساکی
Δ	عملگر لاپلاس-بلترامی روی p -فرم‌ها

مقدمه

فصل اول

۱-۱ مقدمه

با پیشرفت روزافزون دانش بشری در دهه اخیر و تلفیق هرچه بیشتر شاخه های علمی با یکدیگر و همچنین همگام شدن و بهره مندی از فناوری های موجود شاهد ساخته شدن محصولات جدیدی هستیم که ربات ها یکی از معروف و محبوب ترین این محصولات می باشند.

در این رساله، به اختصار مراحل طراحی و پیاده سازی ربات چهارپای عنکبوتی را که ساخته شده است توضیح داده و سپس به تفصیل درباره انتخاب و کالیبره سازی دوربین، نحوه ارتباطات بین قطعات الکترونیکی و مکانیکی، الگوریتم تشخیص موانع و نحوه اجرای و ارزیابی آزمایش ها خواهیم پرداخت. همچنین، نتایج و تحلیل های حاصل از آزمایش ها به منظور ارزیابی کارایی و کاربردی بودن این روش در محیط های واقعی را ارائه خواهیم داد.

در انتهای نیز به بررسی چالش های پیاده سازی پرداخته و پیشنهاداتی برای کارهای آینده خواهیم پرداخت.

۲-۱ ربات و انواع آن

۲-۱-۱ ربات های سری

بطور معمول ربات ها با توجه به نوع کاربردشان، به شکل سری، موازی و یا ترکیبی از هر دو ساخته می شوند. همان طور که در شکل ۱-۱ نشان داده شده است، ربات های سری از یک زنجیره سینماتیکی تشکیل شده اند که در آن مفاصل در یک ارتباط سری باهم قرار می گیرند. این ربات ها به دلیل ویژگی داشتن فضای کاری بزرگتر در صنعت بسیار پر کاربرد هستند. در این ربات ها عومولا عملگر^۱ ها با قرار گرفتن به صورت سری در هر مفصل قرار یک درجه آزادی ایجاد می کنند. علاوه مزیت های ذکر شده برای این دسته از ربات ها باید توجه داشت که ربات های سری لزوماً برای تمامی کاربردهای صنعتی و حتی غیر صنعتی، بهترین انتخاب نبوده و معايیتی نیز به همراه دارند. از جمله این معايیت آنکه خطاهای موقعیتی در آنها جمع شونده هستند. از دیدگاه انرژی میزان مصرف انرژی در این ربات ها بالا است؛ زیرا هر کدام از مفاصل تحریک شده نه تنها بار را حمل می کنند بلکه باید بازو ها ماقبل خود را نیز جابه جا نماید. در نتیجه برای جابه جایی بارهای سنگین بازو های قوی تری مورد نیاز است. در ربات های سری تمامی مفاصل باید کار کنند، اگر یک مفصل از کار بیافتد یا آزاد گذاشته شود ساختار ربات به هم ریخته

^۱Actuator

و در این حالت ربات قادر به حفظ موقعیت خود و انجام وظیفه^۱ محوله نخواهد بود.



شکل ۱-۱: ربات سری [۱]

۲-۲-۱ ربات‌های موازی

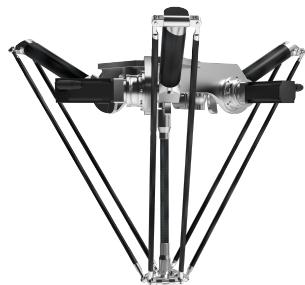
به مرور زمان احساس نیاز به ساختارهایی که محدودیت‌های ربات‌های سری را نداشته باشند به وجود آمد. همانگونه که ما انسان‌ها برای حرکتهای دقیق و یا برداشتن اجسام سنگین از هر دو دست خود استفاده می‌کنیم، می‌توان چنین تصور کرد که استفاده از زنجیره‌های سینماتیکی بسته که چند بازو باهم و به صورت موازی در حرکت دخیل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

همانطور که در شکل ۲-۱ مشاهده می‌شود ربات‌های موازی از زنجیره‌های بسته سینماتیکی ساخته می‌شوند که شامل یک تکیه‌گاه^۲ و یک صفحه متحرک^۳ که به وسیله تعدادی عملگر یا رابط به یکدیگر متصل شده‌اند، می‌باشد. مهمترین ضعف ربات‌های موازی محدودیت در فضای کاری آنهاست.

¹Task

²Base

³Moving Platform



شکل ۱-۲: ربات موازی [۲]

۳-۲-۱ 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 گسترده‌ی 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

^۱Mobile Robots

متمايز استفاده می‌کنند. ربات‌های چرخدار^۱، به عنوان مثال، از چرخ‌های محرک دارای موتورها برای حرکت و جابجایی 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 پاهادار^۲ تلاش می‌کنند تا حرکت حیوانات را تقلید کنند و از پاها برای حرکت استفاده می‌کنند. کنترل ربات‌های پاهادار 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. [۱۲]

۳-۱ ربات عنکبوتی چهارپا

۱-۳-۱ ربات‌های عنکبوتی

راه رفتن با چهار پا برای اکثر حیوانات رایج است و دلیل خوبی برای تکرار آن در ربات‌ها وجود دارد. ربات‌های عنکبوتی^۳، از جمله انواع پیشرفت‌های ربات‌ها، طراحی شده‌اند تا با تقلید از ساختار حرکت و عملکرد عنکبوت‌ها، قابلیت‌های منحصر به فردی در زمینه حرکت و کنترل را ارائه دهند. این ربات‌ها با دارا بودن چندین پا و قابلیت تحرك انعطاف‌پذیر، می‌توانند در محیط‌های مختلف و متغیر عملکرد خوبی داشته باشند. در میان ربات‌های پادار، ربات‌های چهارپا به علت پایداری مناسب تر 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 راه رفتن یک ربات چهارپا را می‌توان به روش‌های مختلف طراحی کرد.

¹Wheeled Robots

²Legged Robots

³Spider Robots

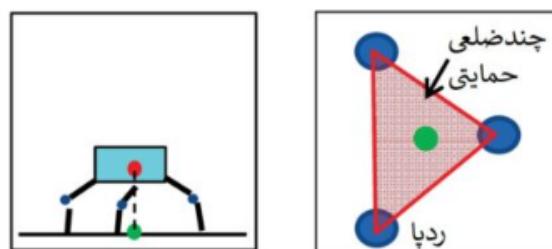


شکل ۱-۴: ربات عنکبوتی [۴]

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

۱-۱-۳-۱ پایداری استاتیکی

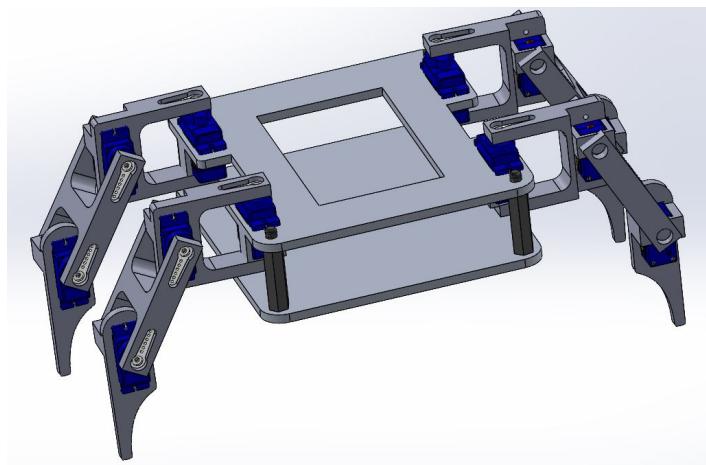
یک ربات با پایداری استاتیک تعادل خوبی دارد و در هنگام ایستادن به زمین نمی‌خورد. این بدین معنی است که مرکز ثقل ربات درون پایه تماس با زمین قرار دارد. فرض کنید یک ربات سه پا داریم که پاها به صورت مثلث تنظیم شده‌اند. این ربات تا زمانی که مرکز ثقل داخل مثلث قرار دارد، به هیچ نوع جابجایی برای ثابت ایستادن 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.



شکل ۱-۵: چندضلعی حمایتی [۵]

۲-۳ ربات‌های عنکبوتی چهارپا

ربات‌های عنکبوتی چهارپا^۱، با شباهت به حرکت عنکبوت‌های طبیعی، به کمک چهار پا محرک تحرک کرده و قادر به تغییر شکل در محیط‌های مختلف هستند. این انعطاف‌پذیری در حرکت، به ربات‌ها امکان حرکت در مسیرهای مختلف، شکل گیری 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 مورد استفاده 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 شده‌است، در شکل ۴-۱ قابل مشاهده



شکل ۴-۱: 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 به‌طور 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.

^۱quadruped Spider Robots

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 صنعت، پژوهشی، 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.

۵-۱ مروری بر الگوریتم‌های مسیریابی

یکی از چالش‌های مهم در علم رباتیک مربوط به مسیریابی است که هدف آن برنامه‌ریزی مسیر با حرکت ربات از موقعیت شروع به موقعیت هدف و در عین حال اجتناب از برخورد با موانع ایستا^۱ و پویا^۲ در محیط است. مسیریابی یک مسئله چالش‌برانگیز در تصمیم‌گیری و کنترل است و به دو صورت انجام می‌شود: اول، طراحی مسیر سراسری که اطلاعات محیط به طور کامل برای ربات در دسترس بوده و ربات قادر به رسیدن موقعیت هدف است. دوم، طراحی مسیر محلی تنها با استفاده از داده‌های حس شده توسط ربات به این معنی که اطلاعات محیط ناشناسی یا تا حدی ناشناس باشد. بسیاری از رویکردهای مسیریابی ارائه شده است که می‌توان آنها را به دو دسته رویکردهای مرسوم^۳ و ابتکاری^۴ تقسیم کرد. روش‌های رایج مانند نقشه راه^۵، میدان پتانسیل مصنوعی^۶ و تجزیه سلولی^۷ نمونه‌هایی از رویکردهای مرسوم هستند. از روش‌های ابتکاری می‌توان به نقشه راه احتمالی^۸، بهینه‌سازی کلونی مورچه‌ها^۹ و بهینه‌سازی ازدحام ذرات^{۱۰} اشاره کرد. با این حال، این الگوریتم‌ها در محیط‌های ایستا و پویا دارای مشکلاتی هستند.

¹Static

²Dynamic

³Conventional

⁴Heuristic

⁵Road Map

⁶Artificial Potential Field

⁷Cell Decomposition

⁸Probabilistic Road map

⁹Ant Colony Optimization

¹⁰Particle Swarm Optimization

یکی از ساده‌ترین الگوریتم‌های ابتکاری، الگوریتم دایکسترا^۱ است که مبتنی بر جستجوی گراف بوده و می‌تواند با گستره‌سازی محیط، حداقل مسیر را بین دو گره مختلف در یک گراف پیدا کند. الگوریتم دیگر *A* است که مشابه الگوریتم دایکسترا است؛ اما از دوتابع هزینه برای حرکت از موقعیت شروع به هدف استفاده می‌کند. این الگوریتم‌ها فقط برای محیط‌هایی با موانع ایستا اعمال شده و کارایی و بهینه بودن مسیر را تضمین می‌کنند، اما مسیر برنامه‌ریزی شده بهشت به گراف بستگی دارد. علاوه بر این، در نظر گرفتن محدودیت‌های دینامیکی ربات‌ها در طول فرایند برنامه‌ریزی دشوار است. در الگوریتم بهبودیافته *A* در [۱۷] از روش پیشنهادی در یک محیط متغیر استفاده کرده و نتیجه نشان می‌دهد که مسیر برنامه‌ریزی شده هموارتر از روش‌های سنتی است. با این حال، این روش محدودیت‌های دینامیکی موانع را نادیده می‌گیرد.

در روش میدان پتانسیل مصنوعی فضای کاری ربات را به صورت فضای پتانسیل در نظر می‌گیریم به صورتی که حداقل مطلق میدان پتانسیل در هدف قرار گرفته و موانع، بیشینه مطلق را اختیار می‌کنند. در این صورت مسئله طرح حرکت تبدیل به تغییرات گرادیان میدان پتانسیل در محیط کاری می‌شود. بسیاری از محققان روش میدان پتانسیل مصنوعی را برای کاربرد جداگانه ربات‌ها به کار برده‌اند و کاربرد و پیاده‌سازی این روش برای مجموعه‌ای از ربات‌های مشارکتی موضوعی چالش‌برانگیز است.

الگوریتم‌های درون‌یابی منحنی مانند: منحنی‌های اسپلاین^۲، بزیر^۳ و چندجمله‌ای^۴ و به عنوان طراحی مسیر آنلاین استفاده می‌شوند. این الگوریتم‌ها شبیه روش‌های مبتنی بر جستجوی گراف هستند و هزینه محاسباتی پایینی دارند؛ زیرا رفتار منحنی توسط چند پارامتر کنترلی تعریف می‌شود. با این حال، بهینه بودن مسیر به دست آمده تضمین نمی‌شود و محدودیت‌های دینامیکی ربات در طول فرایند طراحی مسیر در نظر گرفته نمی‌شوند و علاوه بر این به یک فرایند هموارسازی^۵ برای مسیر به دست آمده نیاز دارند. برای تولید مسیر در مقاله [۱۹]، نویسنده‌گان از پارامتری‌سازی اسپلاین استفاده کردند که محدودیت‌های سینماتیکی و موانع متحرک را نشان می‌دهد. علاوه بر این، سرعت ربات با استفاده از این پارامتر کنترل می‌شود. همچنین برای یافتن راه حل بهینه، یک الگوریتم توسط منحنی بزیر معرفی شده است که نتیجه شبیه‌سازی نشان می‌دهد روش پیشنهادی بهتر از روش سنتی عمل می‌کند.

در مقاله [۱۸]، نویسنده‌گان از ترکیبی از تکنیک‌های درخت جست‌وجوی تصادفی سریع^۶ و اسپلاین برای

¹Dijkstra's

²Spline

³Bezier

⁴Polynomial

⁵Smoothing

⁶Rapidly Random Tree

ایجاد یک مسیر هموار استفاده کردند. الگوریتم دو طرفه درخت جستجوی تصادفی سریع و اسپلاین پیشنهادی بر اساس منحنی مکعب بوده و محدودیت‌های جهت‌دار را برای هر دو موقعیت شروع و هدف ارضا می‌کند. این الگوریتم مشابه دیگر الگوریتم‌های برنامه‌ریزی مسیر نیست و نتیجه به دست آمده در حالت زیر بهینه قرار دارد.

برخی از الگوریتم‌های ابتکاری مانند شبیه‌سازی حرارتی^۱ [۱۵] برای طراحی مسیر در محیط‌هایی با موانع استاتیک و دینامیکی استفاده می‌شوند. این الگوریتم، مسیر به دست آمده برای ربات را بهبود می‌بخشد. مسیر به دست آمده یک راه حل تقریباً بهینه است و برای پیاده‌سازی آنلاین امکان‌پذیر است، اما ابعاد ربات را نادیده گرفته و تنها از موانع با اشکال دایره‌ای اجتناب می‌کند.

روش نقشه راه احتمالی و درخت جستجوی تصادفی سریع به عنوان روش‌های مبتنی بر نمونه‌گیری در نظر گرفته شده‌اند. و هر دو الگوریتم برای ربات‌های هولونومیک و غیره هولونومیک کاربرد دارد. این روش‌ها و انواع آن‌ها به طور گسترده برای تحقیقات استفاده می‌شوند. اما استفاده از آن در کاربردهای عملی دشوار است؛ زیرا پیچیدگی محاسباتی بالایی دارند. در روش‌های مبتنی بر بهینه‌سازی مسیر، ایده اصلی تدوین طراحی مسیر به عنوان یک مسئله بهینه‌سازی است که عملکرد و محدودیت‌های موردنظر ربات در نظر گرفته شود. این رویکرد قادر است مسیری مناسب بین موقعیت‌های شروع و هدف پیدا کند. در پژوهش [۱۴]، یک روش جدید برای پیش‌بینی و اجتناب از برخورد موانع استاتیک و دینامیک در یک محیط ناشناخته ارائه شده است. برای پیش‌بینی سرعت موانع، از فرایند تصمیم‌گیری با استفاده از اطلاعات سیستم سنسوری ربات استفاده کرده‌اند؛ بنابراین ربات قادر است مسیر مناسب را پیدا کند، و بدون برخورد به هدف برسد. نتیجه الگوریتمی کارآمد برای محیط‌های پیچیده و پویا است.

در [۱۶]، یک کنترل پیش‌بین مبتنی بر مدل^۲ غیرخطی برای یک ربات خودران زیر آب ارائه شده است که مسئله طراحی مسیر را با یک چارچوب بهینه‌سازی افق کاهشی همراه با اسپلاین حل می‌کند.

۶-۱ اهداف و نوآوری

با نگاهی به پژوهش‌های انجام شده در زمینه ربات‌های عنکبوتی چهار پا مشاهده می‌شود که در اکثر پژوهش‌های صورت گرفته به دیدگاه تئوری بیشتر اهمیت داده شده است. از این رو در پژوهش پیش رو تلاش شده است تا جنبه‌های عملی ربات‌های پادار مورد توجه قرار گرفته و همچنین تمرکز بر روی بهبود

¹Simulated Annealing

²Model Predictive Control

عملکرد و افزایش قابلیت‌های آنها گردد. در نتیجه این هدف، 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 پایان نامه

پس از اتمام پایان نامه روند هر بخش را مختصرا توضیح خواهم داد.

۸-۱ 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 اسمبل^۳ شدن آن به شروع جدی‌تر بخش نرمافزار و برطرف کردن چالش‌های عملی آن پرداخته شد. نمونه اسمبل شده اولیه ربات در شکل ۷-۱ قابل مشاهده می‌باشد.



شکل ۷-۱: نسخه‌اسمبل شده

^۱Real Time

^۲Online

^۳Assemble

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 تاکید 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 مقدمه‌ای بر پردازش تصویر

۳-۲ پردازش تصویر

۱-۳-۲ 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 ماشین های خودران، دوربین های کنترل جرایم رانندگی، سیستم های تشخیص چهره و... از پردازش تصویر بهره می برند.

۲-۳-۲ 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 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.

۳-۳-۲ پردازش تصویر چیست؟

پردازش تصویر یک زیرشاخه از پردازش سیگنال است که به تحلیل، تغییر، و بهبود تصاویر دیجیتالی می پردازد. در این فرآیند، 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 تحسین کیفیت تصاویر است. این 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 آنها 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.

۴-۲ الگوریتم جستجوی سریع درخت تصادفی

۱-۴-۲ مقدمه

همانطور که در بخش ۱-۵ اشاره شد، الگوریتم های متنوعی در رباتیک کاربرد دارد.

در حوزه‌ی رباتیک، پیشرفت‌های قابل توجهی در الگوریتم‌های برنامه‌ریزی مسیر رخ داده است که به ربات‌های خودکار امکان می‌دهد تا به طور کارآمد در محیط‌های پیچیده حرکت کنند. از میان این الگوریتم‌ها، الگوریتم جستجوی سریع درخت تصادفی^۱ یا به اختصار RRT به عنوان یک رویکرد نوآورانه شناخته می‌شود که به حل چالش یافتن مسیرهای قابل اجرا در فضاهای تنظیم با بعد بالا می‌پردازد. این الگوریتم به عنوان یک روش مبتنی بر هیوریستیک^۲، با استفاده از تصادف و کاوش به سرعت یک ساختار مانند درخت را ایجاد می‌کند که این امر آن را برای کاربردهای بهینه‌سازی زمان و محیط‌های پویا مناسب می‌کند. اصول اساسی این الگوریتم در قابلیت اکتشاف مؤثر فضاهای تنظیمی و تنوع آن، آن را به یک اصول اصلی در تحقیقات و کاربردهای رباتیک مدرن تبدیل کرده است.

الگوریتم RRT که در حوزه‌ی برنامه‌ریزی حرکت معرفی شده است، یک راه حل نوین برای چالش پیچیده یافتن مسیر برای عوامل خودکار در فضاهای با بعد بالا ارائه می‌دهد. در اصل، این روش به دنبال برقراری تعادل بین کاوش و بهره‌برداری می‌گردد. این الگوریتم با یک تنظیم اولیه آغاز می‌شود و به صورت تکراری با نمونه‌برداری تصادفی از نقاط در فضای تنظیمی، آن‌ها را به نزدیک‌ترین نقطه در درخت موجود متصل می‌کند. این رویکرد تضمین می‌کند که الگوریتم مناطقی از فضا را که هنوز کاوش نشده‌اند، کاوش کند، در عین حال به طور تدریجی از مناطقی که قبلاً کاوش شده‌اند بهره‌برداری می‌کند. یکی از ویژگی‌های برجسته‌ی RRT توانایی تطبیق به محیط‌ها و محدودیت‌های مختلف است. با تشکیل تک تک نقاط درخت به طور تدریجی، رشد الگوریتم به سمت مناطقی با کاوش کمتر سوق می‌شود، که این ویژگی آن را به ویژه برای موقعیت‌هایی که فضای تنظیم مشخص نشده یا با موانع پراکنده است، مناسب می‌کند. علاوه بر این، RRT همگام‌سازی سریع را اجرا می‌کند که به آن اجازه می‌دهد که مسیرهای قابل اجرا را به سرعت ایجاد کند، حتی در محیط‌های پیچیده و پویایی.

قابلیت‌های این الگوریتم در کاربردهای زمان‌واقع واقعی نیز یک جنبه جذاب دیگر است. به دلیل ساختار تکراری و تطبیق‌پذیری، این الگوریتم برای مواردی که تصمیم‌گیری باید سریع و واکنش‌پذیر باشد، مناسب است. علاوه بر این، محققان اقدام به توسعه RRT برای مواجهه با چالش‌های خاص، مانند

¹Rapidly Exploring Random Tree

²Heuristic

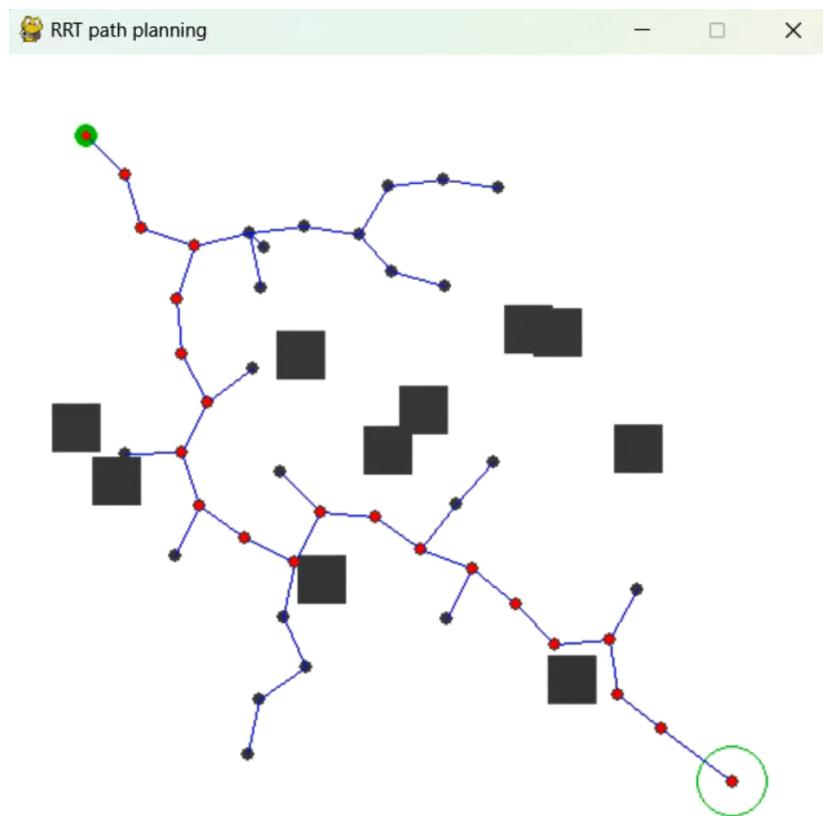
موانع پویا و سیستم های چند عاملی و... کرده اند که بیشتر نشان می دهد چقدر این الگوریتم چند منظوره و عملی است.

به عبارتی، الگوریتم RRT یک الگوریتم نوآورانه برای برنامه ریزی مسیر های خود کار است. با ترکیب کاوش و بهره برداری ، RRT به طور موثر در فضاهای تنظیمی با بعد بالا حرکت کرده و به شکل مناسبی به شرایط محیطی مختلف تطبیق می یابد. چند منظورگی، قابلیت های زمان واقع و تطبیق پذیری با چالش های خاص، RRT را به یک ابزار ارزنده در تحقیقات و کاربردهای رباتیک مدرن تبدیل کرده است و پیشرفت هایی در خود کاری، کارایی و ایمنی ایجاد می کند.

۲-۴-۲ شبیه سازی دو بعدی با پایتون

۱-۲-۴-۲ نتایج شبیه سازی

در ابتدا نتایج حاصل از شبیه سازی توسط پایتون قرار داده شده است. همانطور که در شکل ۱-۲ قابل مشاهده است، یک محیط با ابعاد مشخص و تعداد موانع 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 ۲-۱: شبیه سازی الگوریتم RRT

۲-۲-۴-۲ شبیه کد و تشریح جزئیات برنامه

در زیر شبیه کد این بخش مختصر اقراره داده شده و پس از آن به تشریح کامل تر 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

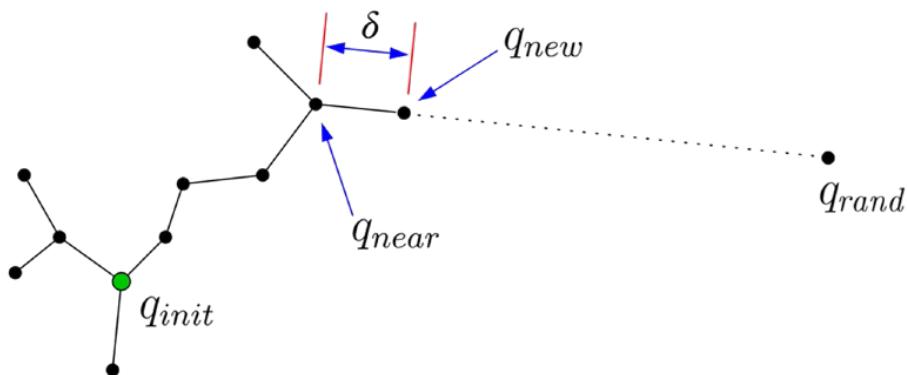
```

همانطور که در شبکه کد 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 می‌باشد. 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.



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

برای درک بهتر مراحل بالا می‌توان شکل ۲-۲ را در نظر گرفت. همچنین قابل ذکر است که برای هر یک از مراحل بالا یک یا دو متده^۱ در مازول نوشته شده در نظر گرفته شده است و این متدها با یکدیگر در ارتباط بوده و یا به صورت تو در تو^۲ در یکدیگر استفاده شده‌اند.

۳-۲-۴-۲ ادغام الگوریتم با قابلیت‌های ربات به منظور پیاده‌سازی

همانطور که مشخص است استفاده از الگوریتم پیشنهادی به تنها ی 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.

¹Method

²Nested

Localization ۵-۲

۱-۵-۲ مقدمه

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 معترض جهان که از میان آنها می‌توان به آزمایشگاه رباتیک اپریل^۱ در دانشگاه میشیگان اشاره کرد، روش‌هایی برای مکان‌یابی با استفاده از تگ‌هایی مشابه به تگ‌های معروف که با نام **کیوآر کد**^۲ شناخته شده‌اند، معرفی کرده اند.



شکل ۳-۲: آزمایشگاه اپریل میشیگان [؟]

^۱APRIL robotics lab

^۲QR code

۲-۵-۲ استفاده از تگ

۱-۲-۵-۲ 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 Aruco که OpenCV نام دارد، استفاده خواهد شد.

۲-۲-۵-۲ شبکه کد و تشریح جزئیات برنامه

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

```
##### Pseudocode #####
Start
1. import necessary libraries
2. Get camera access with IP
3. Define mtx matrix 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
```

مکان یابی در رباتیک به وسیله مختصات مکانی (مثلًا 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 دقت بیشتر در محاسبات، ما یک ماتریس را تعریف می‌کنیم که حاوی فاصله‌های مرکزی دوربین و فوکوس آن است و این ماتریس را به کد مکان یابی ارائه می‌دهیم. این اطلاعات به ربات امکان می‌دهند تا موقعیت و جهت خود را با دقت بیشتری 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

camera = cv2.VideoCapture(0)

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 occured in while loop")

camera.release()

cv2.destroyAllWindows()
```

۶-۲ جمع بندی

فصل سوم

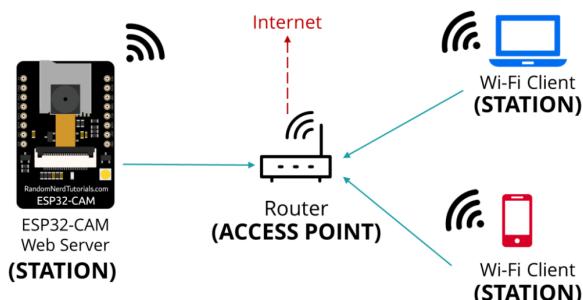
نرم افزار و ارتباطات

۱-۳ ارتباطات و سرور

۱-۱-۳ اتصال بیسیم دوربین به سرور

۱-۱-۱-۳ 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 کلیدوازه نقطه دسترسی^۲ و مُد^۳ ایستگاهی^۴ (یا به اختصار STA) وجود دارد. طبق تعریف، نقطه دسترسی [۶] 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^۵ داشته باشد. اگرچه در یک روتر بی‌سیم، می‌تواند جزء جدایی ناپذیر خود روتّر 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^۶ 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 [۷]

^۱Network

^۲Access Point

^۳Mode

^۴Station

^۵Router

^۶Internet Protocol address

۲-۱-۳ شبکه کد و تشریح جزئیات برنامه

در ادامه، به طور دقیق‌تر، فرآیند اتصال مژول به مودم با استفاده از برنامه نوشته شده در محیط آردوینو را تشریح خواهیم کرد. سپس نحوه استفاده از کتابخانه‌های ارتباط بی‌سیم برای اتصال به شبکه و اخذ 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 تعریف 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 آردوینو پرداخته می‌شود. نخست مشخصه Buad rate را که بیانگر تعداد تغییرات در سطح سیگنال در یک ثانیه است، برابر ۱۱۵۲۰۰^۱ قرار داده شده و سپس پارامترهای تصویر اعم از اندازه تصویر^۱، کیفیت تصویر (Jpeg quality) و... تنظیم گردید.

¹Frame size

²Jpeg quality

```

1 #include "esp_camera.h"
2
3 #include <WiFi.h>
4
5 #define CAMERA_MODEL_AI_THINKER // Has PSRAM
6
7 const char* ssid = "IRLab";
8 // The password is hide for personal reason :)
9 const char* password = "#####";
10 void startCameraServer();
11 void setup() {
12     Serial.begin(115200);
13     Serial.setDebugOutput(true);
14     camera_config_t config;
15     if(psramFound()){
16         config.frame_size = FRAMESIZE_QXGA; // UXGA
17         config.jpeg_quality = 10;
18         config.fb_count = 2;
19     } else {
20         config.frame_size = FRAMESIZE_QVGA; // SVGA
21         config.jpeg_quality = 12;
22         config.fb_count = 1;
23     }
24 }
```

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 مأذول 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 }
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 آردوینو چاپ می‌گردد. نتیجه این اتصال در شکل س قابل مشاهده است.

¹LocalIP

²Serial Monitor

۲-۱-۳ سرور و هندرلرها

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 به‌وسیله 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) 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^۴‌ها 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 شبکه‌کد و تشریح جزئیات برنامه

در زیر شبکه‌کد^۵ این بخش مختصراً 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
```

¹Server

²Request

³Respond

⁴Uniform Resource Locator

⁵Handler

⁶Pseudo code

```

3. Define same configs for camera
4. if(there is any request):
    Check it with all handlers
    if(any handler were called (except "current_color_uri")):
        respond properly
End

```

در قطعه برنامه زیر چند هندلر برای مُدهای مختلف احتمالی تعریف شده است. برخی از آنها برای ماژول ESP32-CAM از پیش تعریف شده‌اند. مهمترین هندلری که با هدف ارسال و مانیتورینگ اطلاعات بین سرور و ماژول و به طبع آن برد STM32 پیاده‌سازی شد، current_color_uri نام دارد. برای این هندلر مقدار uri که بیانگر مسیر^۱ در آدرس است، /currentColor تعریف شد. تابع هندلر آن در ادامه بطور کامل‌تر تعریف شده است. هدف از تعریف این هندلر 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 برشط 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    };

```

^۱Path

```

11 httpd_uri_t stream_uri = {
12     .uri      = "/stream", .method   = HTTP_GET,
13     .handler  = stream_handler, .user_ctx = NULL
14 };
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 هندلرهای
چک می‌شوند تا در صورت نیاز پاسخ مناسب را به آن درخواست برگردانند.

```

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

```

```

14     httpd_register_uri_handler(stream_httpd, &stream_uri);
15 }
16 }
```

همانطور که ذکر شد [۲-۲-۱-۳](#) می‌بایست هندر مورد نظر طوری تعریف شود تا اطلاعات مدنظر به بهترین شکل مدیریت شود. به همین جهت برای ارسال اطلاعات از کوئری استرینگ [۱](#) استفاده شده است.

۳-۲-۱-۳ کوئری استرینگ

گاهی اوقات لازم است که وبسایت از وضعیت [۲](#) ما مطلع باشد و بتواند در همان لحظه، مسیری را طی کند. مثلا 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 [۳](#) با استفاده از کوکی [۴](#) 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.

¹Query String

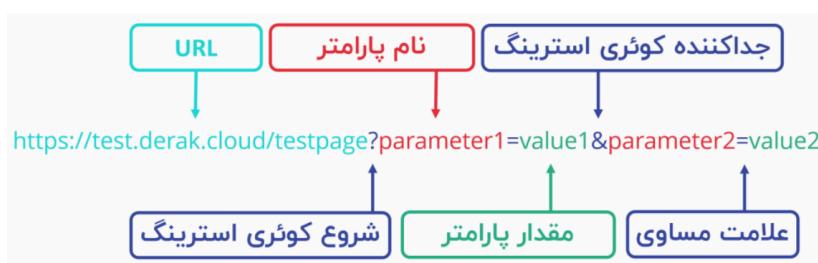
²State

³Client

⁴Cookie

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:



شکل ۲-۳: ساختار کوئری استرینگ [۸]

همانطور که در شکل ۲-۳ مشخص است آدرس های شامل کوئری استرینگ دارای بخش های 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.
- 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.

¹Key

²Value

استفاده از این روش مزایا و معایب متفاوتی دارد که در ذیل مختصرا به آنها اشاره شده است. [۱]

مزایا:

- استفاده ساده

- سریع‌ترین روش انتقال اطلاعات بین صفحات

- عدم تحمیل عملیات اضافه به سرویس‌دهنده و در نتیجه هزینه‌ی کم

معایب:

- اطلاعات، محدود به رشته‌های 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 مشاهده است. 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.

۴-۲-۱-۳ تابع هندلر

در زیر شبکه‌کد این بخش مختصرا قراره داده شده و پس از آن به تشریح کامل‌تر 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 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) {
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) ; }

```

^۱Buffer

```

14    }
15
16    free(buf) ;
17
18    const char resp[] = "Done" ;
19
20    httpd_resp_send(req, resp, HTTPD_RESP_USE_STRLEN) ;
21
22    return ESP_OK ;
23
24}

```

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 می‌باید و در روش انتقال سریال در هر لحظه فقط یک بیت ارسال می‌شود. در مقایسه بین این دو روش می‌توان 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 می‌دهد بیشتر است. همچنین برای فواصل طولانی بکار بردن روش انتقال موازی به علت افزایش سیم‌های ارتباطی، دارای هزینه بالا می‌باشد؛ بنابراین در فواصل طولانی انتقال به روش سریال مناسب تر است.

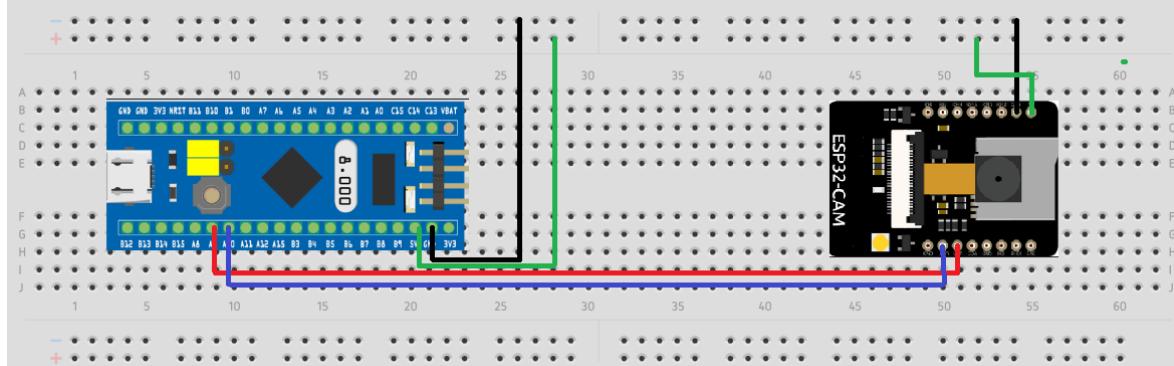
فرستنده و گیرنده سریال همزمان و غیر همزمان جهانی یا USART یک ارتباط از نوع انتقال سریال می‌باشد که امکان ارتباط دوطرفه کامل را می‌دهد و برای ارتباط بین دو دستگاه مفید می‌باشد. در حالتی که هیچ ارسال و دریافتی انجام نمی‌شود یا اصطلاحاً خط انتقال در حالت بیکار (Idle) قرار دارد، سطح ولتاژ مربوط به یک منطقی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

نرخ انتقال‌های که به صورت قراردادی بین فرستنده و گیرنده در یک ارتباط سریال، مشخص می‌شود مقادیر خاصی می‌توانند داشته باشند که مقادیرهای ۱۹۲۰۰، ۹۶۰۰، ۳۸۴۰۰، ۵۷۶۰۰، ۱۱۵۲۰۰ (بیت بر ثانیه) از معمول‌ترین این مقادیرند. 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 USART STM32f101c8t6 و ESP32 CAM به یکدیگر از طریق رابطシリال است که امکان انتقال داده‌ها بین دو دستگاه را فراهم می‌کند. این رابط اغلب برای ارتباط میان میکروکنترلرهای ماژولی 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 U0R از ماژول ESP32-CAM که برای دریافت اطلاعات تعییه شده‌است، به پایه ۹ PA9 از برد STM32 متصل می‌شود. همچنین پایه U0T از ماژول ESP32-CAM که برای ارسال اطلاعات تعییه شده‌است، به پایه ۱۰ PA10 از برد STM32 که برای دریافت اطلاعات تعییه شده‌است متصل می‌باشد. همچنان که در شکل ۳-۳ مشخص است پایه ۵ VDD از ماژول ESP32-CAM به پایه ۵ VDD از برد STM32 متصل می‌شود. 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

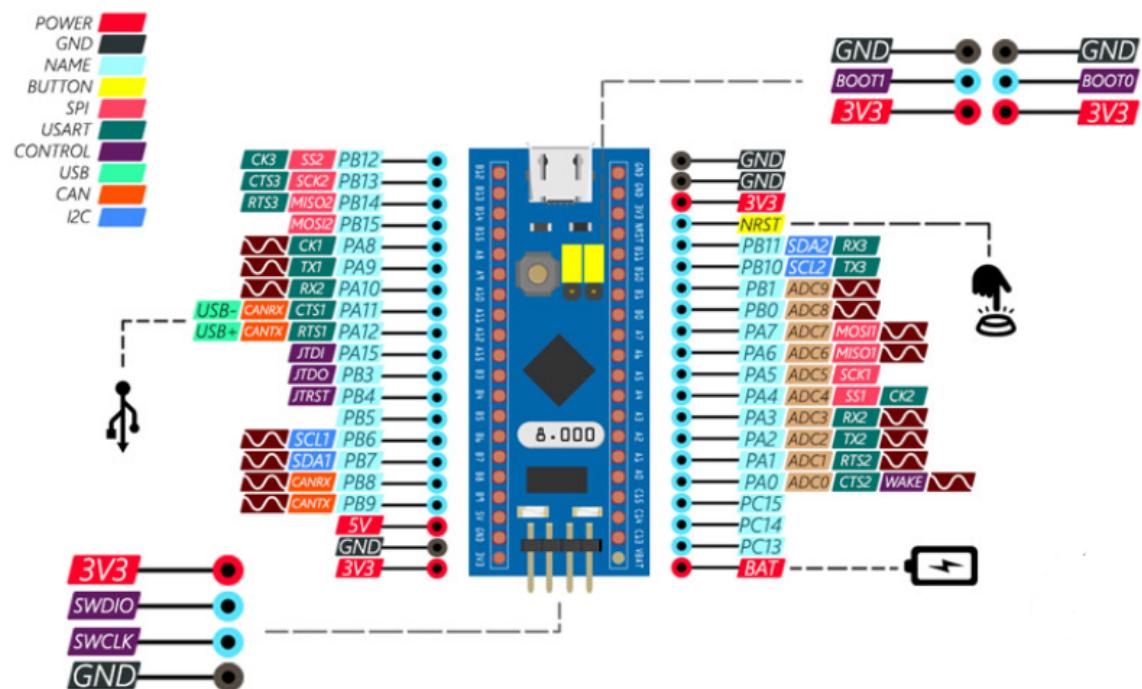
فصل چهارم

سخت افزار الکترونیکی ربات

۱-۴ مقدمه

۲-۴ پردازنده ARM سری f101c8t6

شاید واژه ARM^۱ برایتان آشنا باشد. ARM 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 بعلت تنوع بالا و ارائه کتابخانه‌های برنامه‌نویسی کاربردی، بسیار مشهور شده‌اند.



[۹] : شکل ۱-۴

¹ Advanced RISC Machine² Embedded Systems³ performance lines

۳-۴ مژول ESP32-CAM

ماژول ESP32-CAM یک ماژول 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 WiFi 802.11b به همراه پشتیبانی می‌کند.
- اتصالات: ۹ پایه GPIO و PWM، I2C، SPI و UART.
- فرکانس ساعت: تا ۱۶۰ مگاهرتز
- قدرت محاسباتی میکروکنترلر: حداقل ۶۰۰ DMIPS
- حافظه: ۵۲۰ کیلوبایت SRAM + SD، چهار مگابایت حافظه کارت PSRAM + SD
- دوربین: از دوربین‌های OV2640 پشتیبانی می‌کند که دارای ۲ مگاپیکسل روی سنسور با اندازه آرایه 1622×1200 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

۲-استفاده از یک برد دیگر مانند آردوینو^۲



[۱۰] شکل ۴-۲: ماژول ESP32CAM

^۱Programming

^۲برای برنامه‌ریزی از طریق ارتباطاتی همچون UART

نکته بسیار مهم اینکه برای برنامه ریزی شدن این مژول 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 برنامه‌هایی 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، از جمله پروژه‌هایی که به آن اشاره کردید، یعنی ربات عنکبوتی چهار پا.



شکل ۴-۳: دوربین OV2640 [۱۱]

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 متعدد: این دوربین از فرمت‌های مختلف تصویری مانند JPEG ، YUV ، RGB 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)، 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، Raw Capture و JPEG 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 مژول‌های دیگر را ممکن می‌سازد.
- کاربرد وسیع: دوربین OV2640 به عنوان یک مژول 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 کامپیوترا در نظر گرفت.

۵-۴ سرومотор

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 تنهایی یا به صورت گروهی در ربات‌های چهارپا استفاده می‌شوند تا حرکت و موقعیت دقیق در سیستم رباتیک تضمین شود.



[۱۲] شکل ۴-۴: سروموتور SG90

۱-۵-۴ روش PWM

۱-۱-۵-۴ 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.

۲-۱-۵-۴ 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 سایر قطعات متصل کننده

پین هدر ها و ...

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 معروفی شده در کمترین زمان ممکن انجامید.

فصل پنجم

جمع‌بندی و نتیجه‌گیری و پیشنهادات

در پایان گزارش‌های علمی و فنی لازم است که جمع‌بندی یا نتیجه‌گیری نهایی ارائه شود. در این موارد می‌توان آخرین فصل پایان نامه که پیش از مراجع قرار می‌گیرد را به این امر اختصاص داد.

۱-۵ پیشنهادات

در این بخش پیشنهاداتی که محقق جهت ادامه تحقیقات دارد ارایه می‌گردد. دقت شود که پیشنهادات باید از تحقیق انجام شده و نتایج آن حاصل شده باشد و از ذکر جملات کلی باید پرهیز کرد.

كتاب نامه

- [1] <https://devicebase.net/en/techman-tm12>.
- [2] <https://favpng.com/pngview/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://en.wikipedia.org/wiki/Wireless_access_point.
- [7] <https://randomnerdtutorials.com/esp32-cam-access-point-ap-web-server/>.
- [8]
- [9] <https://thecafrobot.com/learn/getting-started-w-stm32f103c8t6/>.
- [10] <https://www.srkelectronics.in/product/ov2640-camera-module-esp32-cam/>.
- [11] <https://www.aliexpress.com/item/33042013379.html>.
- [12] <https://www.amazon.in/Robodo-Electronics-Tower-Micro-Servo/dp/B00MTFFAE0?th=1>.
- [13] Craig, J. J. *Introduction to robotics: Mechanics and control*. Pearson Prentice Hall, 2005.

-
- [14] Kamil, Farah, Hong, Tang Sai, Khaksar, Weria, Moghrabiah, Mohammed Yasser, Zulkifli, Norzima, and Ahmad, Siti Azfanizam. New robot navigation algorithm for arbitrary unknown dynamic environments based on future prediction and priority behavior. *Expert Systems with Applications*, 86:274–291, 2017.
 - [15] Miao, Hui and Tian, Yu-Chu. Dynamic robot path planning using an enhanced simulated annealing approach. *Applied Mathematics and Computation*, 222:420–437, 2013.
 - [16] Shen, Chao, Shi, Yang, and Buckham, Brad. Model predictive control for an auv with dynamic path planning. In *2015 54th Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE)*, pages 475–480. IEEE, 2015.
 - [17] Sipahioglu, Aydin, Yazici, Ahmet, Parlaktuna, Osman, and Gurel, Ugur. Real-time tour construction for a mobile robot in a dynamic environment. *Robotics and Autonomous Systems*, 56(4):289–295, 2008.
 - [18] Sudhakara, Priyanka, Ganapathy, Velappa, and Sundaran, Karthika. Optimal trajectory planning based on bidirectional spline-rrt for wheeled mobile robot. In *2017 Third International Conference on Sensing, Signal Processing and Security (ICSSS)*, pages 65–68. IEEE, 2017.
 - [19] Yang, Shuaishuai, Wang, Zhuping, and Zhang, Hao. Kinematic model based real-time path planning method with guide line for autonomous vehicle. In *2017 36th Chinese Control Conference (CCC)*, pages 990–994. IEEE, 2017.

پیوست

موضوعات مرتبط با متن گزارش پایان نامه که در یکی از گروههای زیر قرار می‌گیرد، در بخش پیوستها آورده شوند:

۱. اثبات‌های ریاضی یا عملیات ریاضی طولانی.
۲. داده و اطلاعات نمونه (های) مورد مطالعه (Case Study) چنانچه طولانی باشد.
۳. نتایج کارهای دیگران چنانچه نیاز به تفصیل باشد.
۴. مجموعه تعاریف متغیرها و پارامترها، چنانچه طولانی بوده و در متن به انجام نرسیده باشد.

RRT کد پایتون

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 (uppercörnerx, uppéricnery)

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 مکان‌یابی 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

camera = cv2.VideoCapture(0)

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 occured in while loop")

camera.release()

cv2.destroyAllWindows()
```

واژه‌نامه‌ی فارسی به انگلیسی

حاصل ضرب دکارتی	آ
خ	اسکالر
Automorphism	ب
د	بالابر
Degree	پ
ر	پایا
microprocessor	ت
ز	تناظر
Submodule	ث
س	ثبت‌ساز
Character	ج
ص	جایگشت
Faithful	چ
ض	چند جمله‌ای

Connected	همبند	Inner product	ضرب داخلی	ط
Edge	یال	Loop	طوقه	ظ
		Valency	ظرفیت	ع
		Nonadjacency	عدم مجاورت	ف
		Vector space	فضای برداری	ک
		Complete reducibility	کاملاً تحویل‌پذیر	گ
		Graph	گراف	م
		Permutation matrix	ماتریس جایگشتی	ن
		Disconnected	ناهمبند	و
		Invertible	وارون‌پذیر	ه

واژه‌نامه‌ی انگلیسی به فارسی

A	Homomorphism	همریختی
Automorphism	خودریختی	I
B	Invariant	پایا
Bijection	دوسویی	L
C	Lift	بالابر
Cycle group	گروه دوری	M
D	Module	مدول
Degree	درجه	روش
E	Method	
Edge	یال	N
F	Nested	تو در تو، متوالی
Function	تابع	O
G	One to One	یک به یک
Group	گروه	P
H	Permutation group	گروه جایگشتی
	Q	

Graph	گراف	Trivial character	سرشت بدیهی
R		U	
Reducible	تحویل پذیر	Unique	منحصر بفرد
S		V	
Sequence	دنباله		
T		Vector space	فضای برداری



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

Department of ...

M. Sc. Thesis

Title of Thesis

By

Name Surname

Supervisor

Dr.

Advisor

Dr.

Month & Year