به نام خدا

عنوان پروژه: ربات خط یاب با کنترل فازی

دانشجو: علیرضا عرفانی

رشته: مهندسی برق و الکترونیک

مقطع: کارشناسی

استاد راهنما: آقای دکتر محمد رضا نصیری

بنام آنکه جان را فکرت آموخت

چکیده

از جمله مباحثی که در رباتیک بسیار مورد توجه قرار می‌گیرد، کنترل ربات مخصوصاً به منظور تعقیب مسیرهای از پیش طراحی شده است. به لحاظ پیچیدگی ساختار و دینامیک غیر خطی، و بدلیل وجود اصطکاک استاتیکی و گشتاورهای اغتشاشی و تغییرات شدید پارامترهای مدل ربات و همچنین امکان انجام کار در شرایط مختلف و مسیرهای متفاوت، کنترل ربات امری بس پیچیده و دشوار است. از این رو روشهای کنترلی متفاوتی ارائه گردیده که هر کدام دارای مزایا و معایبی مخصوص به خود هستند. یکی از روشهای کنترلی که طی دو دهۀ اخیر توسعه شگرفی را در کنترل سیستمهای پیچیده و غیرخطی داشته، کنترل فازی است. کنترل‌کننده‌های فازی دارای دو مزیت اساسی می باشند، یکی آنکه این کنترل‌کننده‌ها به مدل سیستم حساس نیستند و به چگونگی رابطه ورودی-خروجی سیستم تا حد زیادی غیر وابسته‌اند، و دیگر آنکه دارای ساختار بسیار ساده‌ای بوده و به سهولت قابل پیاده‌سازی اند. از آنجا که حرکات رباتهای هوشمند در پیست مسابقه وابستگی بسیار شدیدی به نوع برنامه و نیز شرایط پیست دارد، لذا با تدوین قوانین بسیار دقیق فازی می توان از انحراف آنها جلوگیری نمود، بطوری که گوئی توسط انسان هدایت می شوند. از این رو تجربیات شخص از طریق منطق فازی جهت عملکردی نه منطقی تر بلکه شبه انسانی تر به ربات اعمال شده، که این همان چیزی است که بشر برای تکامل هوش مصنوعی در پی دارد. همانطور که در بالا آمد، در منطق فازی عملکردی دقیق با منطق صفر و یک (دیجیتال) مد نظر نیست! بلکه در پی آن هستیم که صرفنظر از شکل ظاهری ربات، نتیجه کار تا آنجا که ممکن است، همانطور باشد که انسان می خواهد و یا انجام می دهد.

کلمات کلیدی

ربات- هوشمند- خط یاب- مسابقه- فازی- مکاترونیک- الکترونیک- هوش مصنوعی- مکانیک- تغذیه- کریستال- سنسور- میکروکنترلر- مقایسه کنندهٔ آنالوگ- درایور(راه انداز)- استپ موتور(موتورپله ای)- پروگرامر- کنترل- برنامه- چرخ.

فهرست مندرجات

فهرست تصاویر و جداول

مقدمه

قرن بیست و یکم، سن کودکی علم انسان است که در پی عصر انقلاب صنعتی و سیستمهای بزرگ مکانیکی، عصر بخار و عصر جمع آوری، پردازش و توزیع اطلاعات که به ترتیب در قرون هجدهم، نوزدهم و بیستم شکوفا شدند، آمده است. قرن بیست و یکم، عصر تکنولوژی اطلاعات و سیستمهای هوشمند است. مادر تمام این علوم، قویترین نیروی خلقت یعنی قوهٔ تخیل[۱] انسان می باشد. انسان برای دستیابی آسانتر به آرزوها و خواسته هایش و به عبارتی، خواسته یا ناخواسته به منظور پیشرفت و تکامل[۲] خویش، همواره در تخیلاتش، به دنبال استفاده از ماشینهایی جهت برآورده کردن نیازهای خود بوده است که نمونه های بارز آن را در بسیاری از نوشته ها و فیلمهای علمی و تخیلی می توان دید. در این بین نویسندگانی چون «هوگو گرنسبک»[۳] و «ایزاک آسیموف»[٤] بررسیهای زیادی را در زمینهٔ ماشینهای اتوماتیک ، هوش مصنوعی[٥] و رباتها انجام داده اند. به ویژه آثار «هوگو گرنسبک» که در بسیاری از داستانهای خود[٦] مفاهیم الکترونیک را بکار برده است.

هرچند کلمهٔ «ربات»[۷] اولین بار در سال ۱۹۲۱ توسط رمان نویسی اهل چکسلواکی بنام «کارل کاپک»[۸] در یکی از کتابهایش بکار رفت، ولی منشأ علم رباتیک را بایستی در زمان یونان باستان دانست، آن زمانی که اولین مجسمه های متحرک ساخته شدند.

«کارل کاپک» در کتا ب خود[۹] خدمتگزاران مکانیکی را به نمایش در آورد که قادر بودند کلیهٔ کارهای یک انسان را انجام دهند. در واقع «ربات» معادل کلمهٔ «کارگر» در زبان چک و به معنی «برده» می باشد. از آن زمان تا کنون ربات را به عنوان موجودی مکانیکی که توانایی انجام بعضی از کارها یا حداقل تقلید یکی از رفتارهای انسان را دارد، می شناسند.

نمونه هایی از رباتها را از ابتدا تا کنون به شرح زیر مرور می کنیم:

سال ۲۷۰ پیش از میلاد، مهندسی یونانی بنام «کرسیباس»[۱۰] بوسیلهٔ قطعات متحرک ، ارگ های بادی و ساعتهای آبی را ساخت. در قرن اول پیش از میلاد،«هرو دی الکسندریا»[۱۱] آزمایشاتی را با پرنده های مکانیکی طراحی و به مرحله اجرا در آورد. در سال ۷۷۰ میلادی،

---

۱ Fantasy
۲ Evolution
۳ Hugo Gernsback
٤ Isaac Asimov
٥ Artificial Intelligence (AI)
٦ http://www.hugogernsback.com
۷ Robot
۸ Karel Kapec
۹ R.U.R.- Rassum's Universal Robots
۱۰ Cresibus
۱۱ Hero De Alexandria

ساعتسازی سوئیسی بنام «پیر جاکت دروز»[۱] سه آدمک مکانیکی ساخت که قادر به نواختن موسیقی با استفاده از ارگ، کشیدن اشکال ساده و نگارش بودند. یکی از معروف ترین فیزیکدانان بنام «نیکلا تسلا»[۲] نیز در این زمینه اثری مهم از خود به جای گذاشت، یک زیردریایی مجهز به کنترل رادیویی.

امروزه ربات را سیستمی مکاترونیکی، مطیع[۳] و فاقد شخصیت[۴] که در دو نوع «هوشمند» و «غیر هوشمند» (فرمان پذیر از انسان) قابل ساخت است، تعریف می کنند.

در طول دو دهۀ اخیر از به هم پیوستن علوم مهندسی الکترونیک، برق، کنترل و کامپیوتر با مهندسی مکانیک جهت طراحی و ساخت سیستمهای پیشرفته و پیچیدۀ هوشمند[۵] و مدرن، زمینۀ جدیدی در مراکز آموزشی و پژوهشی کشورهای مختلف دنیا بخصوص در آمریکا، اروپا و ژاپن بوجود آمده است. واژۀ مکاترونیک[۶] جهت هرچه بهتر معرفی کردن این زمینۀ چند تخصصی انتخاب گردیده و بطور چشمگیری این واژه مورد قبول مراکز علمی و صنعتی قرار گرفته است. سالانه همایشهای علمی متعددی هم با این نام جهت ارائه مقالات علمی در سرتاسر دنیا تشکیل می گردد. از جمله کاربردهای آن نیز می توان به مصارف صنعتی، پزشکی، نظامی، خانگی و ... اشاره کرد.

یک سیستم مکاترونیکی در واقع متشکل از سیستمهای مختلفی است که عامل اصلی آن حرکت در یک یا چند قسمت از آن سیستم بوده و استفاده از سنسورهای دقیق هوشمند جهت اندازه گیری پارامترهای مختلف و استفاده از الگوریتمهای هوشمند کامپیوتری جهت اعمال فرامین کنترلی به قسمتهای عمل کننده نیز جزء احتیاجات اصلی هرسیستم مکاترونیکی هستند.

از آنجا که به هنگام مطالعۀ کاربردها و مدارات مربوط به مکاترونیک و رباتیک، در بسیاری از موارد این دو مبحث در کنار هم قرار می گیرند، بنابراین دارای نکات مشترک زیادی نیز می باشند. این امر به این دلیل است که به هنگام تحلیل و بررسی بسیاری از واحدهای درسی علم مکاترونیک، مشاهده می کنیم که این مباحث، عملکرد و ساختمان رباتها را تحت پوشش قرار می دهند. از طرفی دیگر واحدهای درسی رباتیک نیز با ساختمان دستگاهها و وسایلی سروکار دارند که ترکیبی از مکانیک و الکترونیک و البته در سطوح پیشرفتۀ آن هوش مصنوعی می باشند. اغلب طرحهای مربوط به رباتیک و مکاترونیک مدرن درجه ای از هوشمندی را شامل می شوند. و در آخر اینکه رباتیک تنها، شاخه ای از مکاترونیک می باشد.

امروزه استفاده از تکنولوژی ربات در زمینه‌های مختلف صنعت و اتوماسیون، افزایش چشمگیری یافته است. یکی از شاخه‌های این تکنولوژی، رباتهای متحرک می باشد که در صنایع هواپیماسازی و خودروسازی، ساخت وسایل الکترونیکی و لوازم خانگی و... کاربرد وسیعی پیدا کرده است. در واقع اولین نسل رباتهای واقعی نیز، رباتهای صنعتی[۷] می باشند که به عنوان ماشینهای کاربردی سازنده وسایل، که وظیفۀ انجام کارهای خطرناک، تکراری و خسته کننده را به عهده دارند، به دنیای مدرن ما وارد شدند. افزایش استفاده از رباتهای متحرک، به همراه نیاز به دقت عملکرد بالای آن‌ها موجب شده است تا مسئلۀ طراحی کنترل‌کننده های این سیستمها از اهمیت بالایی برخوردار شود. نسلهای برتر اینگونه رباتها را در انواع هوشمند آنها می توان یافت که بعضی از آنها عبارتند از رباتهای جنگجو، انسان نما،

---

[۱] Pierre Jacquet-Droz
[۲] Nicola Tesla
[۳] Obedient
[۴] Impersonal
[۵] Intelligent
[۶] Mechatronics
[۷] Industrial Robots

صخره نورد، مین یاب، امدادگر، خط یاب، نقاش، ماوز[۱] (لابیرنت[۲])، خدمتکار، فوتبالیست.

در این پایان‌نامه سعی بر آن است که طراحی و ساخت ربات خط یاب[۳] را که یکی از رباتهای کلاسیک در نسل جدید می باشد، با یکی از جدید ترین و بهترین روشهای کنترلی شناخته شده و به ساده ترین نحو آموزش دهیم. بدین منظور سطح متوسطی از دانش روز در ارتباط با مدارهای الکترونیکی، طراحی و پیاده سازی آنها، همچنین دربارهٔ میکروکنترلرها و برنامه نویسی آنها الزامی است. لذا آگاهی از نحوهٔ کار میکروکنترلرهای AVR، سنسورهای مادون قرمز(IR) و بایاسینگ آنها، مقایسه کننده های آنالوگ، استپ موتور و درایو(راه اندازی) آنها وهمچنین مهارت در برنامه نویسی به زبان C توصیه می شود.

روش کنترلی بکار رفته در این ربات، کنترل فازی[۴] می باشد که سعی بر آن است تا در این پروژه هرچند ساده ولی در حد نیاز، آن را آموزش دهیم. لذا در این باره نیازی به دانش قبلی نیست و جهت یادگیری آن تنها به فصل چهارم این پروژه بسنده می کنیم.

ابتدا قوانین و کلیاتی پیرامون مسابقات رباتهای خط یاب را مورد بررسی قرار داده و سپس اشاره‌ای به تاریخچهٔ پیدایش مجموعه‌های فازی و منطق فازی خواهیم داشت؛ بعد از مطرح نمودن مقدمات منطق فازی، جهت تشریح و آشنایی با بخشهای مختلف علم مکاترونیک و رباتیک طی فصلهایی متوالی به توضیح بخشهای مختلف تشکیل دهندهٔ این علم نوین می پردازیم. عمده بحث و تمرکز این پایان نامه حول معرفی و ساخت ربات خط یابی است که هدایت آن تماماً به صورت هوشمند و با کنترل فازی صورت گرفته است.

---

۱ Maze
۲ Labyrinth
۳ Line Follower
۴ Fuzzy Control

. فصل اول .

قوانین مسابقه

در کلیهٔ مسابقات رباتهای خط یاب، هدف نهایی یک چیز است؛ ولی از آنجایی که در مسابقات گوناگون، تغییرات جزئی در بعضی از بخشها مشاهده می شود، تفاوتهایی را نیز در قراردادها شاهد هستیم. در اینجا یکی از مسابقات اخیر را به عنوان الگو مورد بررسی قرار می دهیم.

۱-۱) مسابقات سال ۲۰۰۵

ربات خط یاب می بایست با حداکثر سرعتی که می تواند، خط را تعقیب کند. ربات باید توسط سازنده در نقطه شروع مسابقه قرار داده شود که حتی می تواند توسط داور، جهت قرارگیری آن نیز تعیین شود. همزمان با شروع حرکت ربات، داور زمان را محاسبه میکند.

اگر ربات از خط خارج شود و آن را گم کند، شخص همراه این اجازه را دارد که ربات خود را روی خط قرار داده و آن را از نو تنظیم کند[۱]. بازای هر بار تنظیم مجدد ربات، یک اخطار[۲] به آن ربات داده می شود. ربات بایستی دقیقاً از همان نقطه ای که خط را ترک کرده بود، مسیر را دنبال کند. در هر مرحله، ربات میتواند تا سه مرتبه از خط خارج شده و مجدّداً روی خط قرار داده شود.

---

[۱] To Reset
[۲] Penalty

به هر ربات حداکثر دو دقیقه در هر مرحله اختصاص داده می شود؛ اگر در این مدت، ربات نتواند مسیر تعیین شده را کامل بپیماید و همچنین اگر پیش از دو دقیقه ربات مرحلهٔ جاری را با موفقیت به اتمام رساند و یا برای بار چهارم از خط خارج شود، داور اتمام وقت آن مرحله را اعلام می کند. قابل توجه است که در طول زمانی که صرف تنظیم مجدّد ربات می شود، وقت نگه داشته نمی شود.

پس از نگه داشتن زمان، داور مقدار مسافت طی شده را ثبت می کند. با اتمام هر مرحله، مرحله بعد آغاز می شود.

طراحی مسیر نیز در روز برگزاری مسابقه انجام می پذیرد.

۱-۲) تعریف

هر چند ربات خط یاب طرحی ابتدایی و کلاسیک از رباتها است که با مکانیکی ساده نیز قابل ساخت می باشد ولی این ربات بسیار دیدنی، سرگرم کننده، آموزنده و توسعه پذیر نیز است. در این نوع از رباتها از سیستمهای کنترلی گوناگونی از جمله میکروپروسسورها و حتی سیستمهای کنترلی سادهٔ آنالوگ یا دیجیتال می توان استفاده کرد.

ربات خط یاب یکی از بهترین رباتهایی است که برای به نمایش در آوردن طرحهای جدید سنسورهای گوناگون و سیستمهای کنترلی متنوع و پیشرفته، در قالبی ساده از رباتهای هوشمند که می بایست خطوط سیاه را در زمینه سفید پیدا کرده و تعقیب کنند، بکار می رود. در مسابقات رباتهای خط یاب، بر حسب سرعت و میزان مسافتی که ربات طی میکند به آن امتیاز داده می شود.

۱-۳) مشخصه های طراحی

ربات خط یاب بدون هیچ راهنمایی و بدون کوچکترین فرمانی از جانب طراح و سازنده خود باید وظیفه خود را انجام دهد. ابعاد این ربات(طول و عرض آن) هر یک می بایست کمتر از ۹ اینچ (۲۳ سانتی متر) باشد و در کل امتیازات ویژه ای نیز به طراحی های اصولی و علمی تعلق می گیرد.

۱-٤) میدان مسابقه[۱]

مسابقه از چهار مرحله مجزا که هر یک در زمینی صاف و هموار به رنگ سفید و مربعی شکل به ابعاد تقریبی هر ضلع ٤٨ اینچ (۱,۲ متر) برگزار میشود، تشکیل شده است. پیست مسابقه نیز توسط چسب برق مشکی با عرضی حدود ۱۸ تا ۱۹ میلیمتر مسیردهی شده است.

هر مرحله از مسابقه از مرحله قبل پیچیده تر است، بطوری که ربات در طول مسابقه ممکن است با شرایط زیر مواجه شود:

- پیچهای خمیده ملایم[۲] (با شعاع[۳] بزرگتر از ۲۰ سانتیمتر)
- پیچهای تند[۱] (با شعاع کوچکتر از ۱۰ سانتیمتر)

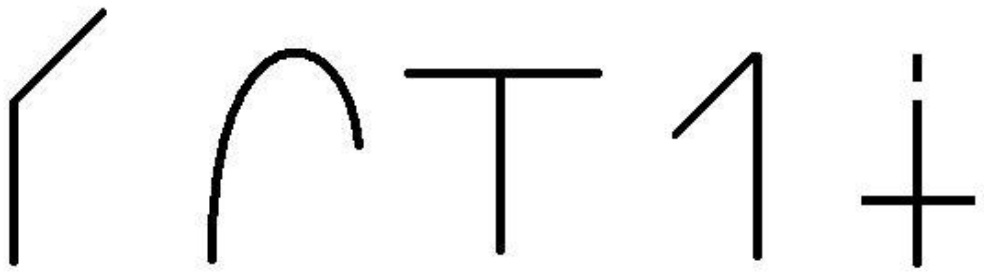
---

[۱] The Arena
[۲] Gently
[۳] Radius

- زاویه های منفرجه[۲] (بزرگتر از ۹۰ درجه)
- مسیرهای متقاطع
- زاویه های تند و نوک تیز (کوچکتر از ۹۰ درجه)
- شکافهایی[۳] به طول کمتر از ۵ سانتیمتر

همچنین ربات باید قادر باشد تا در شدت های متفاوت نور داخل اطاق، عملکرد صحیحی را به نمایش بگذارد.

در زیر چند نمونه از مسیرها را مشاهده می نمایید:



۵-۱) امتیاز دهی[۴]

هر مرحله دارای مسیری به طول تقریباً ۲ تا ۶ متر می باشد.
شرح کامل امتیازات به قرار زیر است:

توضیحات	امتیازات	مثال
مسافت طی شده[۵] امتیاز متناسب با درصدی از مسیر که طی شده است	بازای هر ۱۰% از کل مسیر ۱۰ امتیاز. ۱۰۰ امتیاز برای کل مسیر.	ربات ۳۰% از کل مسیر را طی کرده: ۳۰ امتیاز
زمان سپری شده[۶] امتیازمتناسب با سرعت ربات	۵۰۰ امتیازبخش بر، زمان طی شده،برحسب ثانیه: $\frac{500}{t}$	ربات در مدت ۲۵ ثانیه کل مسیر را طی کرده: ۲۰ امتیاز
جریمهٔ تنظیم مجدد[۷] هر بار تنظیم مجدد ربات، از آن، امتیاز کسر می کند	با هر بار تنظیم در حین مسابقه، ۱۰ امتیاز کسر می گردد.	شخص، ۲ مرتبه، در یک مرحله، ربات خود را از نو تنظیم کرده: (۲۰-) امتیاز
پاداش کامل طی کردن مسیر[۸] امتیاز خاصی به رباتی که کل	۵۰ امتیاز در هر مرحله	ربات دو مرحله را بدون اینکه در هر یک مجدداً تنظیم شود، طی کرده: ۱۰۰ امتیاز

---

[۱] Tightly
[۲] Obtuse
[۳] Gaps
[۴] Scoring Method
[۵] Distance Traveled
[۶] Elapsed Time
[۷] Reset Penalty
[۸] Completion Bonus

مسیر را بدون نیاز به تنظیم مجدد طی کند، اعطا می شود		
پاداش خلاقیت[۱] بر حسب تشخیص داور برای ابتکار در طراحی ربات، به آن امتیاز اطلاق می شود	تا ۵۰ امتیاز برای هر مرحله	۱.ساختار و مداربندی زیبا و برنامه ریزی نرم افزاری مناسب: ۵۰ امتیاز ۲.کیت ربات اصلاح نشده: ۰ امتیاز

. فصل دوم .

منطق فازی

خلق، بسط و گسترش اندیشهٔ فازی توسط «پروفسور لطفی زاده» استاد دانشگاه کالیفرنیا، برکلی.

در سال ۱۹۶۵ میلادی این دانشمند ایرانی اولین مقالهٔ خود در زمینهٔ «فازی» را با عنوان «مجموعه های فازی»[۲] منتشر کرد. شاید در تصور کسی نمی گنجید که این مقاله اولین جرقه از یک جهان بینی جدید در عرصهٔ ریاضیات و علوم و اولین قدم در معرفی بینشی نو و واقع گرایانه از جهان، در چهارچوب مفاهیمی کاملاً بدیع اما بسیار سازگار با طبیعت انسان باشد.

تفکر فازی[۳] از دیدگاهی فلسفی نشأت می گیرد که سابقه ای چندهزارساله و به قدمت تاریخ فلسفه دارد. همانگونه که فلسفهٔ ادیان الهی با طبیعت و سرشت انسان سازگار است، تفکر فازی با الهام از فلسفهٔ شرقی جهان را همانگونه که هست معرفی می کند. در فلسفهٔ ارسطویی که در مقابل فلسفهٔ شرق قرار دارد، همه چیز به دو دستهٔ سیاه و سفید یا بله و خیر تقسیم می شود. مفاهیم منطقی و نتایج حاصله از استدلالات منطقی نیز در فلسفهٔ ارسطویی هیچگونه حالت میانه ای ندارد. در این فلسفه نمی توان تا اندازه ای راستگو و ضمناً کمی هم دروغگو بود. نمی شود همزمان نسبتاً جوان و تا اندازه ای هم پیر بود. در فلسفهٔ ارسطویی مرزها کاملاً مشخص و تعریف شده هستند.

در تفکر فازی مرز مشخصی وجود ندارد و تعلق عناصر مختلف به مفاهیم و موضوعات گوناگون نسبی است. به این ترتیب می بینیم که این تفکر تا چه اندازه با طبیعت جهان و انسان سازگار است. تفکر فازی دیدگاهی تازه را معرفی می کند که تعمیم منطق ارسطویی است؛ اما بر اساس این دیدگاه، ریاضیات کلاسیک نیز که بر منطق ارسطویی استوار است زیر سؤال می رود. از این رو مخالفتهای بسیاری را نیز از بدو شکوفایی خود به همراه داشته است.

منطق دیجیتال متداول که فقط از یک تصمیم "بله" یا "خیر" تشکیل شده است، برای

---

[۱] Creativity Bonus
[۲] Fuzzy Sets
[۳] Fuzzy Thought

پروژه های شامل هوش مناسب نمی باشد. اگر یک حالت سوم متناظر با جواب "شاید" را پیاده سازی نماییم، کمی بیشتر به مفهوم هوش نزدیک شده ایم. این روش اساس منطق فازی[۱] است.

دانشمندان کامپیوتر، رشتهٔ "هوش مصنوعی" را ساخته اند زیرا دانش و معلومات در نظر آنها همان قوانین هستند که با منطق دیجیتال قابل نوشتن می باشند، ولی با گذشت مدت زمان زیاد و صرف هزینهٔ بسیار در این رشته هنوز نتوانسته اند محصولات هوشمند قابل توجهی را عرضه کنند.

مهندسان فازی، نرم افزارها و تراشه هایی را تهیه می کنند که می توانند به سیستمهای کامپیوتری قدرت استدلالی نزدیک به قدرت استدلال انسان بدهند. این توانایی باعث می شود ماشینها هوشمندتر شده و کار با آنها ساده تر گردد. محققان فازی نیز این پیشرفت در زمینهٔ هوشمندی را مدیون قوانین هستند ولی نه قوانین بکار گرفته شده توسط مهندسان هوش مصنوعی بلکه "قوانین فازی"؛ و علت آن نیز در نسبی بودن آن قوانین می باشد که در قوانین دیجیتالی چنین چیزی به چشم نمی خورد.

انواع دیگری از سیستمهای فازی با استفاده از تجربیات خود توانایی یادگیری و برنامه ریزی دارند. سیستمهای فازی خیلی سریع هوشمند هستند.

از جمله مصارفی که سیستمهای فازی سریع تا به امروز داشته اند، عبارتند از:

- کنترل هوشمند قطارهای زیرزمینی.
- هدایت و کنترل نرم و سریع هلی کوپترها به طور اتوماتیک.
- سیستمهای تهویهٔ مطبوع دقیق و سازگار.
- دوربینهای عکاسی و فیلم برداری با قابلیت تنظیم و قدرت زوم بهینه.
- ماشین های لباسشویی فازی که با توجه به ابعاد، وزن، جنس و بافت لباسها و میزان کثیفی آنها شستشویی مطلوب را ارائه می دهند.
- جاروبرقی های فازی که با اندازه گیری میزان غبار و با توجه به جنس سطح زیر، به بهترین نحو عمل نظافت را انجام می دهند.
- تلویزیونهای فازی که با توجه به شدت نور محیط و کیفیت تصویر و رنگ آن، دائماً و با سرعتی فوق العاده تصویری مناسب را برای بیننده قابل مشاهده می سازند.
- سیستمهای کنترل ترافیک هوشمند در خیابانها.

البته در حال حاضر ورودی کلیهٔ این سیستمهای خبره، سنسورهای پیشرفته ای از جمله سنسورهای مادون قرمز می باشند.

از منطق فازی می توان در برنامه های خاصی که برای هوش مصنوعی مناسبند، استفاده نمود. از این رو در ساخت این ربات خط یاب هوشمند نیز از منطق فازی برای کنترل مسیریابی و حرکت آن استفاده شده است.

۱-۲) مجموعه های فازی

در فضای U، مجموعهٔ فازی A، با یک تابع عضویت $\mu_A: U \rightarrow [۰،۱]$ مشخص می شود. یعنی برای هر عضو u در U یک مقدار $\mu_A(u)$ در فاصلهٔ [۰،۱] تعریف می شود، که درجهٔ عضویت u در A نامیده می شود.

یک مجموعهٔ فازی را برای مجموعهٔ مرجع پیوسته به شکل زیر نشان می دهیم:

$$A=\int_u \left(\mu_A(u)/u\right)$$

[۱] Fuzzy Logic

تکیه گاه[١] مجموعهٔ فازی، مجموعه u هایی است که در رابطهٔ $\mu_A(u)>0$ صدق کند.

عملگرهای اصلی بر روی مجموعه های فازی

از آنجا که هر مجموعهٔ فازی را با تابع عضویت آن بیان می کنیم، اعمال اصلی بر روی مجموعه های فازی نیز تبر حسب توابع عضویت قابل تعریف هستند که چند نمونه از این عملگرها در ادامه آورده شده است:

- اجتماع

$$\mu_{A\cup B}(u)=\max\{\mu_A(u),\mu_B(u)\}$$

- اشتراک

$$\mu_{A\cap B}(u)=\min\{\mu_A(u),\mu_B(u)\}$$

- مکمل

$$\mu_{\overline{A}}(u)=1-\mu_A(u)$$

٢-٢) متغیر های زبانی

متغیر های زبانی یا محاوره ای، متغیر هایی هستند که مقادیر آنها نه اعداد، بلکه کلمات یا جملات در زبان طبیعی می باشند. این متغیرها یکی از ابزارهای اساسی منطق فازی و استنتاجات تقریبی می باشند؛ هر متغیر زبانی X بوسیلهٔ یک پنج تایی مرتب به فرم (X ,T(x) ,U ,G , M) مشخص می شود که در آن X نام متغیر است که در مجموعهٔ U تغییر می کند و G یک قاعدهٔ نحوی (معمولاً به فرم دستور زبان)، برای تولید مجموع ترمهای T(x) مربوط به متغیر X است و M یک قاعدهٔ معنایی است که به هر ترم از T(x) معنای آن را مربوط می سازد. M(x) نیز زیرمجموعهٔ فازی U است.

٢-٣) استدلال و استنتاج تقریبی

گزاره های شرطی اهمیت اساسی در استنتاج واستخراج نتایج از مجموعه ای مفروضات، دارند. قوانین کنترل کننده های فازی، یک مجموعه گزاره های شرطی هستند که به یکی از دو صورت نوعی زیر بکار برده می شوند:

١) If X_1 is A_1 and X_2 is A_2 and …Then Y_1 is B_1 and Y_2 is B_2 and …

٢) If X_1 is A_1 and X_2 is A_2 and …Then $Y_1= f_1(X_1,X_2,\ldots)$ and $Y_2= f_2(X_1,X_2,\ldots)$ and…

که X_i ها متغیرهای زبانی و A_i ها مجموعه های فازی هستند که به عنوان مقادیر زبانی این متغیرها می باشند و f یک تابع ریاضی است که می تواند یک رابطهٔ خطی به شکل زیر باشد:

$$y=f(X_1,X_2,\ldots)=c_0+c_1x_1+c_2x_2+\ldots$$

[١] Support

که c_i ها مقادیر عددی غیر فازی و $x_۱$ و $x_۲$ و ... مقادیر غیر فازی به ترتیب مربوط به متغیرهای زبانی $X_۱$ و $X_۲$ و ... هستند.

با داشتن مقادیر مربوط به قسمت مقدم قوانین (ورودیها)، روشهای مختلفی برای استنتاج و بدست آوردن خروجیها وجود دارد که یک روش سادهٔ آن به صورت زیر می باشد:

فرض کنید n قانون داریم که قانون i ام به شکل زیر است:

R_i : If $X_۱$ is $A^i_۱$ and $X_۲$ is $A^i_۲$ Then y is B^i

دراین قانون $X_۱$ و $X_۲$ ورودیهای یک کنترلر و y خروجی آن هستند بطوریکه مقدار عددی $X_۱$ برابر $x^*_۱$ و مقدارعددی $X_۲$ برابر $x^*_۲$ (که $x^*_۱$ و $x^*_۲$ اعداد حقیقی هستند) می باشد. برای هر قانون یک درجهٔ همسازی به شکل زیر محاسبه می کنیم:

$$w_i = \min\left(\mu_{A_1^i}\left(x_1^*\right), \mu_{A_2^i}\left(x_2^*\right) \right)$$

لازم به توضیح است که اگر در قسمت مقدم قوانین بجای رابط and از رابط or استفاده شده بود، در رابطهٔ بالا از عملگر max بجای min استفاده می شد.

با داشتن w_i ها می توان خروجی استنتاج شده از قوانین را به شکل زیر بدست آورد:

$$y = \frac{\sum_{i=1}^{n} w_i \cdot b_i}{\sum_{i=1}^{n} w_i}$$

که در رابطهٔ فوق، b_i محل پیک تابع عضویت B_i است.

. فصل سوم .

الکترونیک ربات

منظور از الکترونیک ربات، کلیهٔ قطعات و مدارهای الکترونیکی بکار رفته در این ربات هوشمند می باشد. مدار استفاده شده، وظیفهٔ

- دیدن خط و ارسال این پیغام به مغز جهت پردازش نرم افزاری
- ارسال اطلاعات پردازش شده، ازطریق درایورها به موتورها برای چرخش و حرکت
- سرعت ارسال اطلاعات
- اعلام اینکه کدامیک از سنسورها خط را دیده اند
- تأمین انرژی لازم برای هر یک از وظایف فوق

را بر عهده دارد.

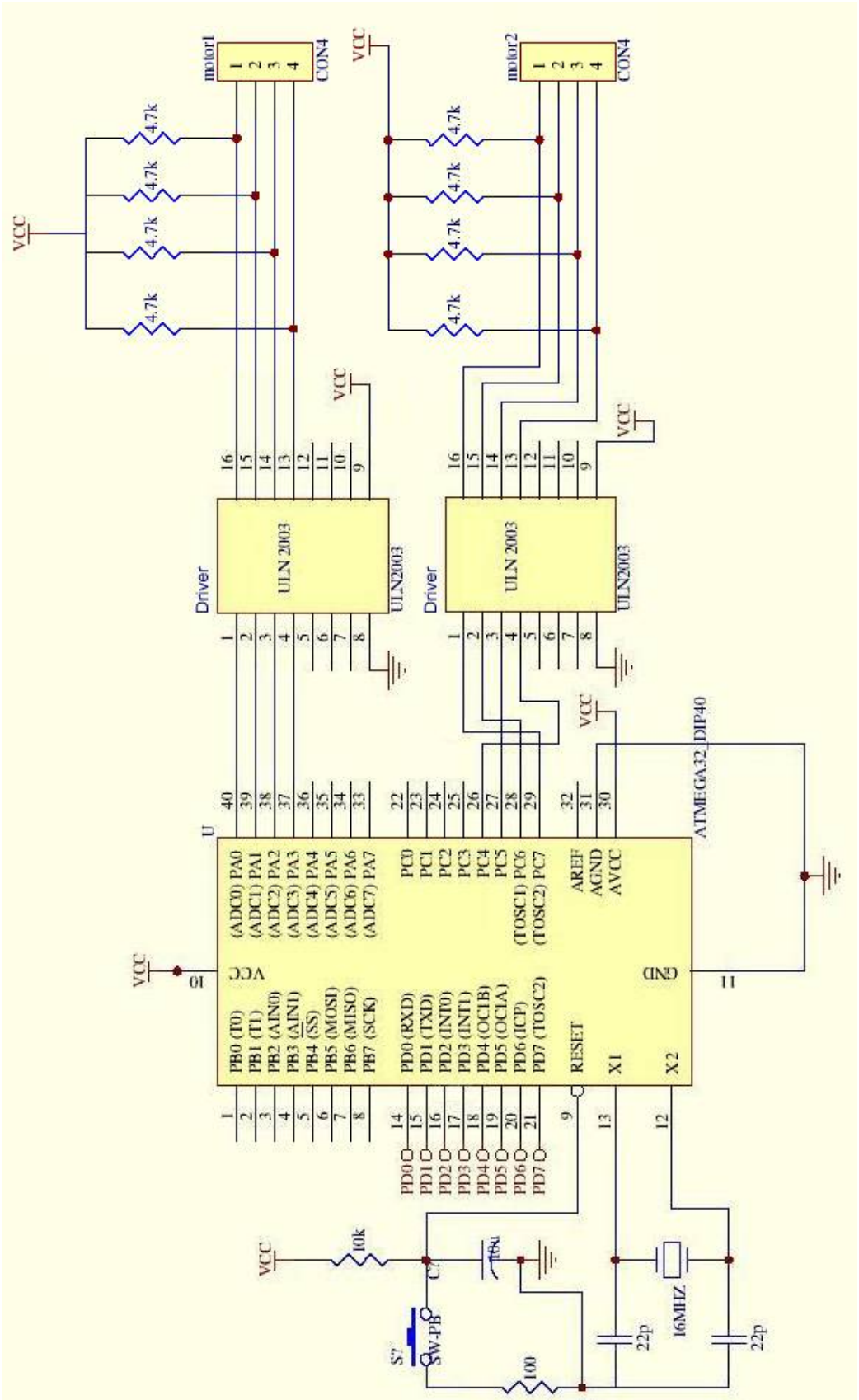
از آنجا که چشمها، مغز، استپ موتورها، اعصاب(خطوط انتقال دیتا) و همچنین تأمین انرژی لازم تماماً الکترونیکی می باشند، لذا مدار الکترونیکی این ربات را می توان به چند بخش که وظیفهٔ هر یک کاملاً مشخص است، تقسیم نمود. بخش بینایی، بخش عصبی و مغز، بخش حرکتی و بخش تغذیه.

در ادامه هر یک از این بخشها را بطور مجزا مورد بررسی قرار می دهیم. ابتدا با نقشهٔ مدار که در بخش بعد آمده است، آشنا می شویم.

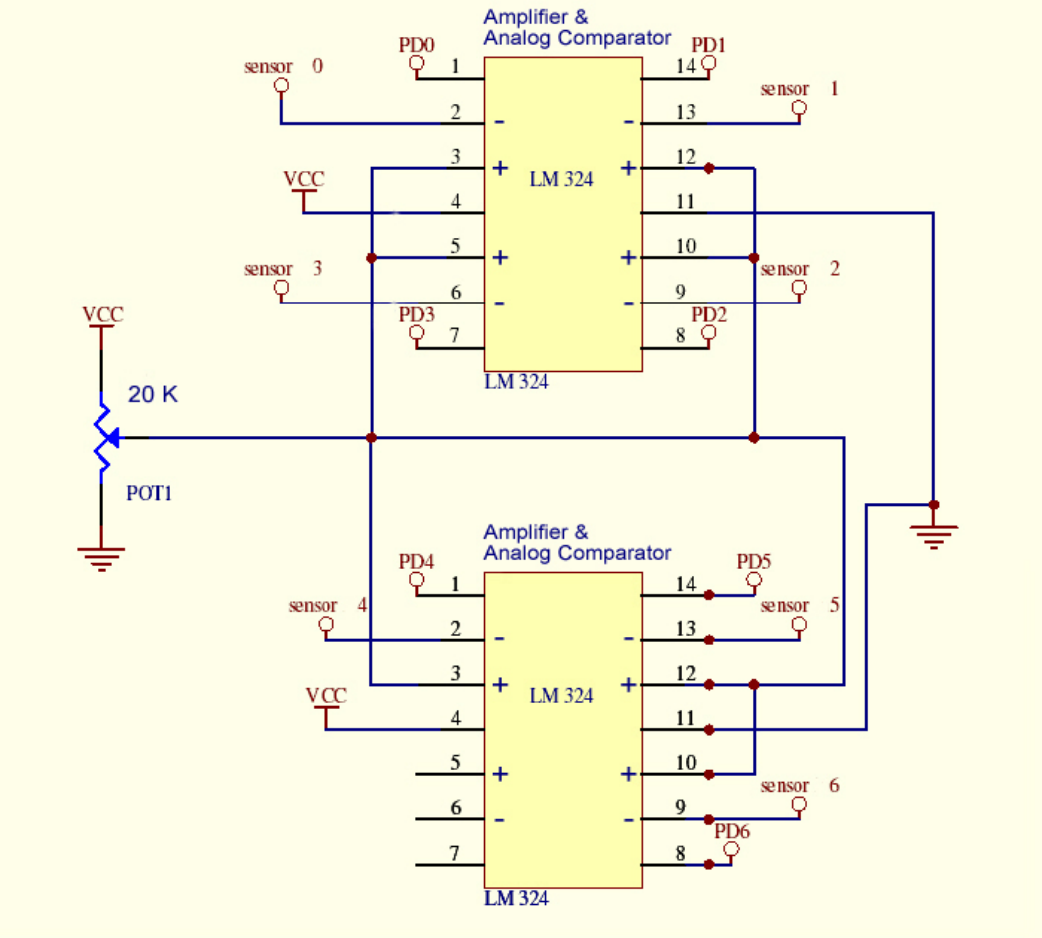
۱-۳) شماتیک مدار

طرح مدار را در چند قسمت مطابق شکلهای زیر ملاحظه می کنید:

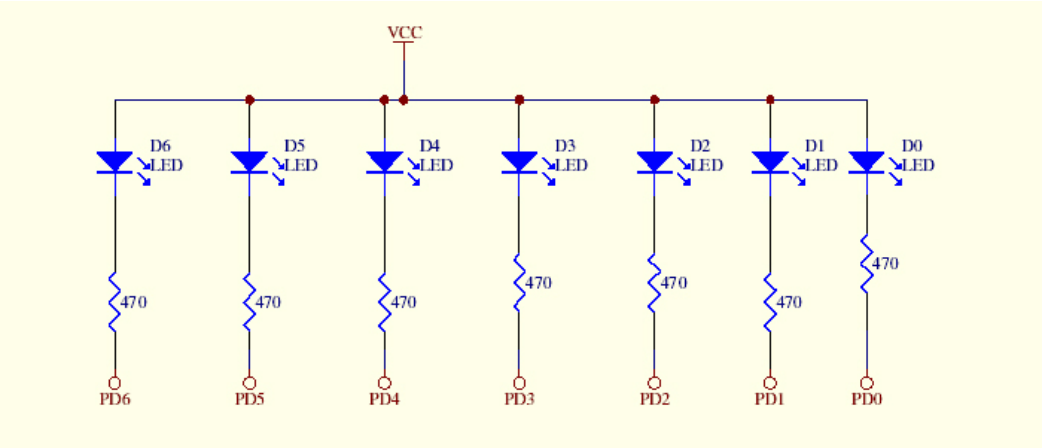
- بخش پردازشگر و مدارات مربوطه، درایور موتورها و دو استپ موتور. ورودی میکروکنترلر، پورت D می باشد:


motor1
motor2
CON4
4.7k
VCC
Driver
ULN 2003
ULN2003
ATMEGA32_DIP40
U
PB0 (T0)
PB1 (T1)
PB2 (AIN0)
PB3 (AIN1)
PB4 (SS)
PB5 (MOSI)
PB6 (MISO)
PB7 (SCK)
PD0 (RXD)
PD1 (TXD)
PD2 (INT0)
PD3 (INT1)
PD4 (OC1B)
PD5 (OC1A)
PD6 (ICP)
PD7 (TOSC2)
RESET
X1
X2
GND
AREF
AGND
AVCC
PC0
PC1
PC2
PC3
PC4
PC5
PC6
PC7
(ADC0) PA0
(ADC1) PA1
(ADC2) PA2
(ADC3) PA3
(ADC4) PA4
(ADC5) PA5
(ADC6) PA6
(ADC7) PA7
(TOSC1)
(TOSC2)
10k
100
22p
16MHZ
S7
SW-PB

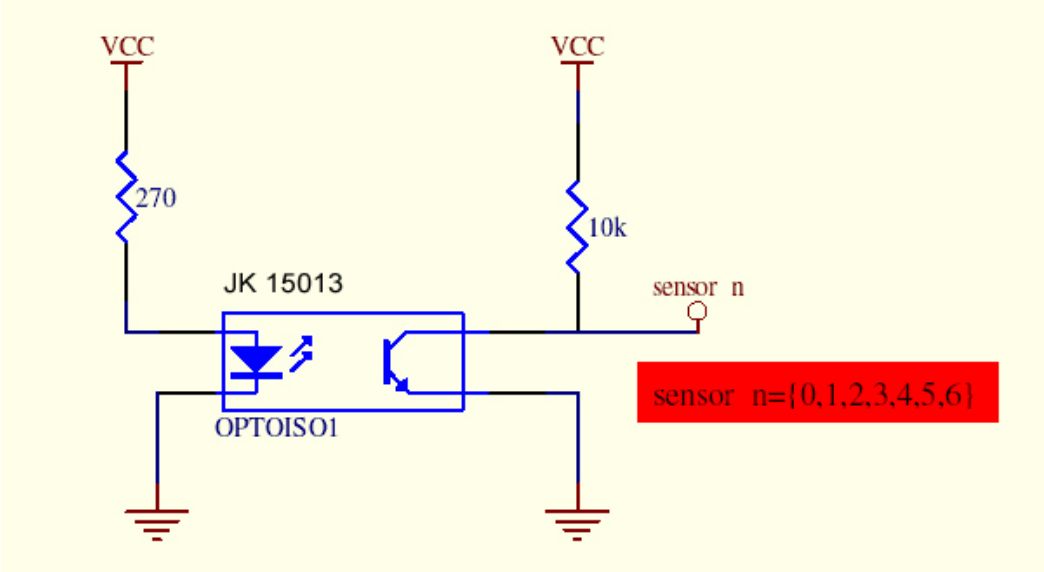
- بخش ورودی، سنسورهای مادون قرمز و مقایسه کننده های ولتاژ منفی. خروجی مقایسه کننده ها به ورودی میکروکنترلر(پورت D) می رود:



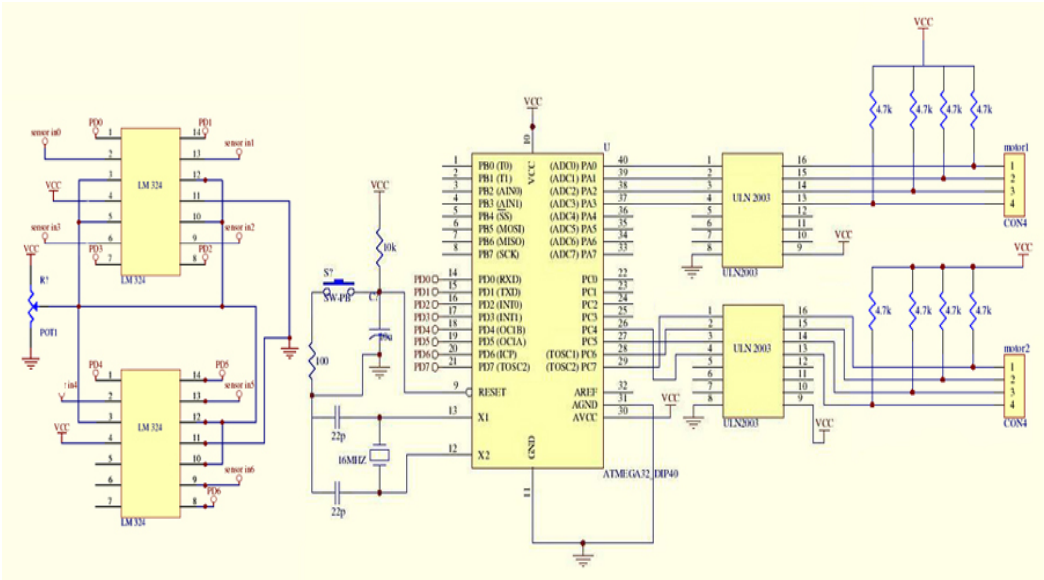
- مدار LEDهای نشان دهندهٔ وضعیت سنسورها. هر کدام به یکی از خروجیهای مقایسه کننده می روند:



- مدار بایاسینگ سنسورهای IR[۱]. خروجی هر سنسور به یکی از ورودیهای منفی مقایسه کنندهٔ آنالوگ[۲] می رود:



شماتیک کلی مدار به صورت زیر است:



[۱] مادون قرمز : InfraRed
[۲] Analog Comparator

۲-۳) تغذیه ربات

انرژی مورد نیاز در پروژه های رباتیک و مکاترونیک معمولاً توسط آداپتور یا باطری تأمین می شود.

در انتخاب باطری مناسب باید دقت و بررسی لازم صورت گیرد. اطلاعات مورد نیاز برای تعیین بهترین باطری ممکن در هر پروژه ای عبارتند از ولتاژ، جریان، توان، وزن، محل قرارگیری و ابعاد باطری. البته پیشرفته بودن باطری تأثیر بسیار مثبتی بر عملکرد ربات بویژه در مسابقات می گذارد. معمولاً استفاده از چند باطری کوچک در یک پکیج بسیار مناسب تر از یک باطری بزرگ بوده و همچنین در طراحی های پیشرفته تر نیز برای تغذیهٔ ربات از باطری های قابل شارژ استفاده می شود. جهت استفادهٔ هر چه بهتر از منابع تغذیه و کارایی بهتر و واضحتر آنها، توصیه می شود برای بخش پردازشگر و الکترونیک ربات و بخش حرکتی و موتورها، از منابعی مجزا استفاده شود.

باطریهای گوناگونی درربا‌تیک مورد استفاده قرار می گیرند. در زیر به چند نمونه از آنها اشاره داریم:

باطری سربی-اسیدی[۱]**:** این باطری فناوری ساخت ساده ای داشته و ارزان می باشد، مزیت این باطری، توانایی تولید جریانهای بالا یا پایین در محدودهٔ دمایی متفاوت و قابل شارژ بودن آن است. مهمترین عیب باطریهای سربی-اسیدی وزن زیاد و افت ولتاژ آنها هنگام دشارژ است. این باطریها در اندازه ها و شکلهای مختلف قابل دسترس هستند.

باطری Deep Cycle: این باطری همانطور که از نامش پیداست، به گونه ای ساخته شده که برای استفاده های طولانی مدت، مشکلی ندارد. سطح جریان این نوع باطری پایین تر از بقیه می باشد. این باطری نیز قابل شارژ بوده و البته به زمان شارژ طولانی نیاز دارد.

باطری روی-کربن: این باطری قابلیت شارژ دوباره را نداشته ولی به جهت ارزان بودن آن، کاربرد وسیعی پیدا کرده است. در این باطریها بر حسب میزان دشارژ، سطح ولتاژ کاهش می یابد.

باطری قلیایی با قابلیت شارژ دوباره: عموماً باطریهای قلیایی مانند باطری روی-کربن قابل شارژ نمی باشند، ولی نوعی باطری قلیایی طراحی شده است که قابلیت شارژ مجدد تا ۲۵ بار یا بیشتر را دارا می باشد.

باطری نیکل-کادمیوم: این باطری قابل شارژ مجدد می باشد. همچنین می توانند در یک ولتاژ نسبتاً ثابت، جریان بالایی را تولید کنند. به دلیل گران بودن فلز کادمیوم، این باطری گران می باشد.

باطری نیکل-هیدروکسید فلز[۲]**:** این باطری که از ترکیب آلیاژ فلز و هیدروژن و اکسید نیکل به همراه هیدروکسید پتاسیم تشکیل شده است، جایگزین بسیار خوبی برای باطری نیکل-کادمیوم به حساب می آید. چرا که علاوه بر ارزانتر بودن، عمر مفیدشان هم باندازهٔ % ٤٠ بیشتر است.

باطری لیتیوم و یون لیتیوم: لیتیوم واکنش دهنده ای ایده آل در فن آوری باطریها می باشد. انرژی تولید شده در یک باطری لیتیوم، ٥ برابر بیشتر از باطریهای سربی-اسیدی هم اندازهٔ آنها و۳ برابر بیشتر از سلولهای قلیایی است. سلولهای لیتیوم معمولاً دارای ولتاژ آغازین ۳ ولت می باشند، در نتیجه وزن این باتریها کمتر و هزینهٔ مصرفی آنها پایین تر و همچنین ولتاژ آنها بالاتر و پایدارتر از بقیه است. شایان ذکر است، لیتیوم در تماس با آب واکنش داده و هیدروژن آزاد شده باعث افزایش فشار باطری می شود که می تواند موجب

---

[۱] Lead-Acid Battery

[۲] Nickle-Metal Hydride Battery

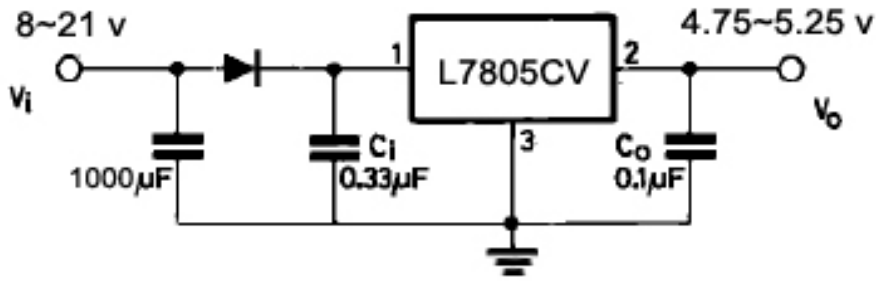
انفجار باطری شود.

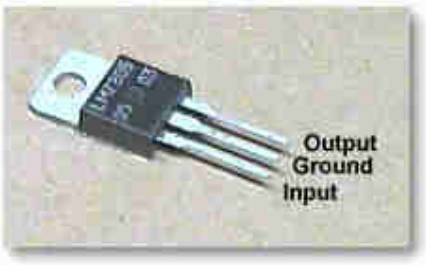
استفاده از برق شهر و آداپتور، گزینهٔ دیگری است که در بسیاری از تولیدات مکاترونیکی و همچنین در رباتها نیز به چشم می خورد. البته بسته به ولتاژ عملیاتی میکروکنترلر و مدار مربوطه باید ولتاژ تنظیم شده ای از برق شهر توسط آداپتور و در صورت لزوم رگولاتورهای ولتاژ، به مدار ربات اعمال شود.

در ارتباط با محل نصب باطریها در یک ربات باید توجه داشته باشیم که، از آنجا که باطریها سنگین ترین جزء تشکیل دهندهٔ ربات هستند، در نقطه ای قرار گیرند که به مرکز سقل ربات نزدیکتر باشند و به حفظ تعادل آن کمک کنند. باطریها باید در جایی تعبیه شوند که برای شارژ یا تعویض آنها مشکلی وجود نداشته باشد.

در این ربات هوشمند، به جهت عملکرد بهتر، از دو نوع تغذیه و به طور مجزا استفاده شده است. بخش الکترونیکی و کنترلی آن تماماً توسط باطریهای نیکل-هیدروکسید فلز، که قابل شارژ می باشند، تغذیه شده و دو موتور به همراه درایورهایشان، انرژی لازم را از آداپتور و رگولاتور ولتاژ بدست می آورند. آداپتور مورد استفاده نیز ۱۰۰۰mA می باشد که دلیل این انتخاب را در بخش ٤-٦ بررسی می کنیم.

مدار رگولاتور مطابق شکل زیر است:





۳-۳) بینایی ربات

سیستم بینایی این ربات از ۷ سنسور مادون قرمز "JK۱۵۰۱۳" و دو مقایسه کنندهٔ ولتاژ منفی تشکیل شده است.

مقایسه کننده ها وظیفهٔ تشخیص رنگ سیاه و سفید از یکدیگر را بر عهده دارند. مقایسه کننده در اصل یک تقویت کنندهٔ عملیاتی(OP-AMP) می باشد. هر تقویت کنندهٔ عملیاتی دارای دو ورودی به نامهای ورودی وارونگر(-) و ورودی غیروارونگر(+) بوده که می تواند به دو صورت مورد استفاده قرار گیرد.

در حالت اول که "حالت خطی" نام دارد، opamp به صورت یک تقویت کنندهٔ متداول AC یا DC با بهرهٔ ولتاژی که توسط حلقهٔ فیدبک تعیین می شود، عمل می کند. هنگامی که سیگنالها باید بدون تغییر شکل موج تقویت شوند، از این حالت عملکرد استفاده می شود. در این حالت اگر سیگنالی به ورودی وارونگر اعمال شود، همان سیگنال ولی با فاز مخالف در خروجی ظاهر می گردد. همچنین اگر این سیگنال به ورودی غیروارونگر داده شود، در خروجی نیز دقیقاً همان سیگنال البته با تأثیر بهرهٔ تقویت کننده تحویل گرفته می شود.

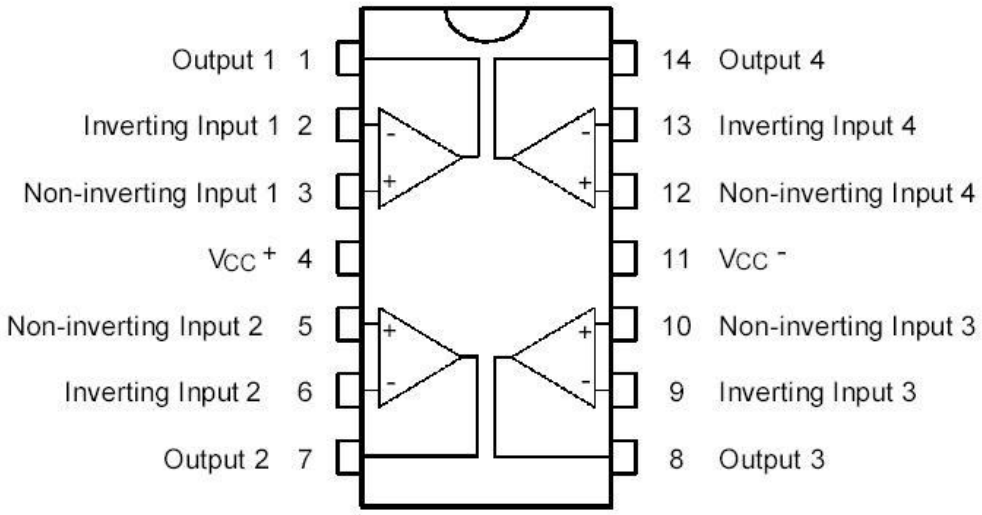
حالت دوم وقتی است که از opamp به عنوان "مقایسه کننده" استفاده می شود. مدار مقایسه کنندهٔ ولتاژ یک مدار تقویت کننده عملیاتی با بهرهٔ زیاد است که دو ولتاژ ورودی را با هم مقایسه می کند و در خروجی سیگنالی ایجاد می کند که نشان می دهد ورودیها با هم برابرند یا خیر. ورودی اول که معیار سنجش ورودی دیگر است، ولتاژ مرجع نامیده می شود. ولتاژ مرجع را خودمان و توسط یک مقسم ولتاژ ساخته شده از مقاومتها، تنظیم می کنیم. در این حالت نیز بسته به اعمال ولتاژ تست(ورودی دوم) به پایه های ورودی opamp، دو حالت در خروجی رخ می دهد. اگر ولتاژ تست به پایهٔ منفی(وارونگر) اعمال شود، در صورتی که از ولتاژ مرجع کمتر باشد و یا با ولتاژ مرجع برابر باشد، خروجی ولتاژی مثبت(High) و در صورتی که از ولتاژ مرجع بیشتر باشد، خروجی ولتاژی منفی یا برابر صفر خواهد شد؛ این نوع از مقایسه کننده ها را، مقایسه کنندهٔ ولتاژ منفی می نامند. همچنین اگر ولتاژ تست به ورودی مثبت opamp (غیروارونگر) اعمال شود، به شرطی که از ولتاژ مرجع کمتر و یا با آن برابر باشد، خروجی ولتاژی منفی (Low) و هنگامی که از ولتاژ مرجع بیشتر باشد، خروجی مقداری مثبت خواهد شد؛ این گونه از مقایسه کننده ها، مقایسه کننده های سادهٔ ولتاژ یا مقایسه کنندهٔ ولتاژ مثبت نامیده می شوند. در هر دوحالت هنگامی که مقدار ولتاژ ورودی از ولتاژ مرجع می گذرد، به دلیل بهرهٔ زیاد حالت "عملکرد ناگهانی" رخ می دهد. به این معنی که زمانی که ورودی به آرامی تغییر می کند، خروجی opamp به سرعت از سطح منطقی High به Low یا از سطح منطقی Low به High می رود. این حالت عملکرد، زمانی مناسب است که از مدار برای تریگر کردن بار توسط سیگنالهای فرستاده شده از سنسورها یا مدارات دیگر برای پاسخگویی سریع استفاده می شود.

در این ربات از دو IC[۱]، "LM۳۲٤" به عنوان مقایسه کنندهٔ ولتاژ منفی استفاده شده است که هر یک، از چهار تقویت کنندهٔ عملیاتی تشکیل شده اند. خروجی هر سنسور(پایهٔ کلکتور) به یکی از ورودیهای منفی این دو مقایسه کننده وارد میشود.

مدار داخلی این ICها در شکل زیر نشان داده شده است. جهت اطلاعات بیشتر دربارهٔ این ICها به دیتاشیت آن که در فایلی ضمیمهٔ پایان نامه آمده است، رجوع نمایید.

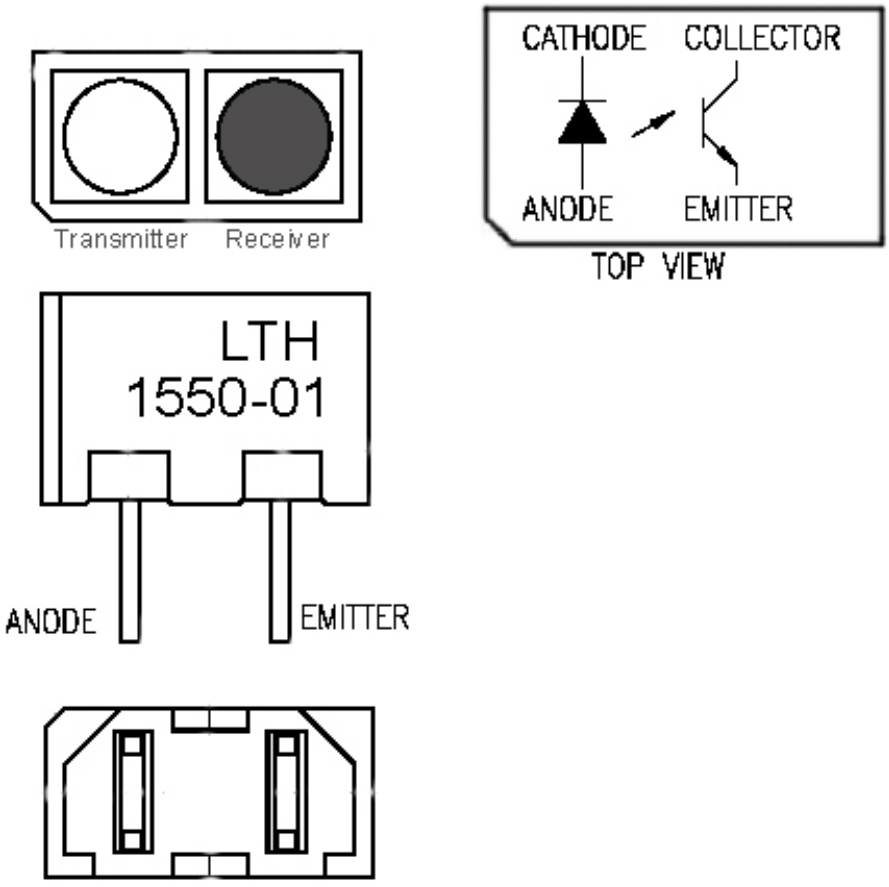
---

[۱] Integrated Circuit : مدار مجتمع



سنسورهای مادون قرمز بکار رفته در این ربات هوشمند، یکی از جدیدترین سنسورهای تشخیص رنگ سیاه و سفید است که با تابش موج مادون قرمز به سطح و دریافت بازتابش آن از همان سطح، می تواند رنگها را از هم تشخیص دهد؛ البته فقط دو رنگ سیاه و سفید برایش معنا دارد. در واقع این سنسور یک فتوترانزیستور IR می باشد. همانطور که روی بدنهٔ این سنسور نیز چاپ شده است، در دیتاشیتها و در اینترنت این سنسور با نام "LTH ۱۵۵۰-۰۱" دیده می شود.

در شکل زیر ساختار این سنسور و ترتیب پایه های آن را می بینید:



مدار بایاسینگ این سنسور در بخش (١-٤) آمده است. برای اطلاعات بیشتر به فایل ضمیمهٔ این پایان نامه مراجعه نمایید.

سنسورها در هر لحظه طیف رنگ زیر خود را به ولتاژ تبدیل نموده و این ولتاژ به ورودی منفی مقایسه کنندهٔ آنالوگ می رود. در آنجا این ولتاژ نمونه با ولتاژ مرجع مقایسه می شود. از آنجا که مقایسه کننده، منفی می باشد به هنگامی که سنسوری خط مشکی را حس کند(روی خط قرار گیرد)، ولتاژی برابر صفر در خروجی مقایسه کننده ظاهر می شود. در این هنگام LED- ی متناظر با همان سنسور به نشانهٔ دیده شدن خط توسط آن سنسور، روشن می گردد.

۳-٤) مغز ربات

منظور از مغز ربات قطعه ای سخت افزاری و تماماً الکترونیکی است که در حالت بسیار سادهٔ امروزی یک پردازشگر می باشد به این معنی که وظیفهٔ انتقال اطلاعات، حساب و منطق و تصمیم گیری را بر عهده دارد. این نوع را "ریزپردازنده" یا "میکروپروسسور"[۱] می نامند. هر میکروپروسسور وظیفهٔ دیگری نیز بر عهده دارد که آن کنترل بخشهای خارجی از قبیل حافظه[۲] و ورودی-خروجی[۳] می باشد. اتصالات این بخشها توسط گذرگاه[٤] آدرس، گذرگاه اطلاعات و گذرگاه کنترل برقرار می باشد که شباهت به سیستم عصبی طبیعی دارند. در واقع میکروپروسسور یک کنترل کنندهٔ قابل برنامه ریزی است تا هر وضعیتی را کنترل نماید.

این قطعات در سطوح پیشرفته تر دست کم دو بخش حافظه و ورودی-خروجی را نیز در خود دارند. این دسته جدید را "میکروکنترلر"[٥] می نامند.

یکی از این میکروکنترلرهای جدید، میکروکنترلر AVR است که به عنوان خانواده ای بزرگ از سه خانوادهٔ میکروکنترلرهای امروزی، سهم عمده ای از مصرف را به خود اختصاص داده است. AVRها، میکروکنترلرهای ۸ بیتی از نوع CMOS[٦] می باشند که توان مصرفی پایینی دارند و بر اساس ساختار پیشرفتهٔ RISC[۷] ساخته شده اند. این ساختار به میکروکنترلرهای AVR این امکان را می دهد که اکثر دستورات را تنها در یک پالس ساعت اجرا نمایند؛ به این ترتیب می توان بازای هر یک مگا هرتز، یک مگ[۸] دستور را در ثانیه[۹] اجرا کرده و برنامه را از لحاظ سرعت پردازش و مصرف توان، بهینه نمود. AVRها، ۳۲ رجیستر[۱۰] همه منظوره و مجموعه دستورات قدرتمندی را شامل می گردند. تمامی این ۳۲ رجیستر مستقیماً به ALU[۱۱] متصل شده اند. بنابراین دسترسی به دو رجیستر هم در یک سیکل ساعت ممکن است. این ساختار موجب می گردد تا سرعت آن نسبت به میکروکنترلرهای CISC[۱۲] تا ۱۰ برابر هم افزایش یابد.

خانوادهٔ میکروکنترلرهای AVR، تراشه هایی پیشرفته با امکانات جانبی کامل هستند. این میکروکنترلرها در سه نوع کلی وجود دارند:

- Tiny AVR (AT tiny)
- Classic AVR (AT ۹۰S)
- Mega AVR (AT mega)

تفاوت بین این سه نوع به امکانات موجود در آنها مربوط می شود. Tiny AVRها غالباً تراشه هایی با تعداد پایه و مجموعه دستورات کمتری نسبت به Mega AVRها می باشند و به عبارتی "از لحاظ پیچیدگی" حداقل امکانات را دارند. Mega AVRها دارای حداکثر

---

[۱] Microprocessor
[۲] Memory
[۳] Input/Output (I/O)
[٤] Bus
[٥] Microcontroller
[٦] Complementary Metal Oxide Semiconductor (CMOS)
[۷] Reduced Instruction Set Computers
کامپیوترهایی که در آنها ساختار به گونه ای است که با کاهش تعداد دستورات، سرعت سیستم افزایش یافته است.
[۸] Meg : ۱۰^٦
[۹] Mega Instruction Per Second (MIPS)
[۱۰] Register : ثبات
هر رجیستر از تعدادی سلول الکترونیکی(Flip Flop) که کوچکترین واحدهای حافظهٔ سخت افزاری می باشند، تشکیل شده است.
[۱۱] Arithmetic Logic Unit : واحد محاسبه و منطق
[۱۲] Complex Instruction Set Computers
کامپیوترهایی که در آنها ساختار به گونه ای است که تعداد دستورات زیاد و پیچیده است.

امکانات بوده و Classic AVRها که قبل از دو گروه دیگر تولید شده اند، جایی بین آن دو را دارا هستند.

امروزه بجای استفاده از سری AT ۹۰S، تراشه ای از گروه AT tiny یا AT mega را جایگزین می کنند.

اسامی میکروکنترلرها بیانگر "نوع" و "میزان حافظهٔ Flash[۱]" در آنها است. به طور مثال میکروکنترلرهای AT۹۰S۸۵۳۵، ATtiny۲۵ و ATmega ۱۶۱ به ترتیب از خانواده و نوع Classic، Tiny و Mega می باشند. برای محاسبهٔ مقدار حافظهٔ آنها نیز کافی است بزرگترین توان دو از عدد موجود در نام هر یک را، انتخاب کنیم؛ در این صورت سه میکرو کنترلر یاد شده، به ترتیب دارای حافظهٔ Flash- ی برابر با ۸، ۲ و ۱۶ هستند.

تقسیم بندی دیگری از جهت "میزان ولتاژ قابل قبول" و "محدودهٔ فرکانس کاری"، برای میکروکنترلرهای AVR وجود دارد. بدین جهت میکروکنترلرهای بدون پسوند، با پسوند L و با پسوند V را داریم.

اسم میکروکنترلر	محدودهٔ ولتاژ تغذیه	فرکانس قابل قبول کریستال
میکروکنترلر AVR بدون پسوند	٤_٥,٥ v	۰_۱٦ MHz
میکروکنترلر AVR با پسوند L	۲,۷_٥,٥ v	۰_۸ MHz
میکروکنترلر AVR با پسوند V	۱,۸_٥,٥ v	۰_٤ MHz

در حقیقت از جدول فوق پیداست که در انتخاب میکروکنترلر AVR مناسب برای پروژهٔ مورد نظر، توجه به دو ویژگی "توان مصرفی کم" و "فرکانس کاری بالاتر" که در تضاد با یکدیگر هستند، از اهمیت ویژه ای برخوردار است؛ و بسته به شرایط باید بهترین انتخاب ممکن را کرد.

میکروکنترلرها در بسته بندی های متفاوت و به شکلهای گوناگونی وجود دارند که بسته به کاربرد و میزان استفاده از آنها و مکان قرارگیریشان، از یکی از آنها استفاده می شود؛ در حالت کلی دو شکل بسته بندی برای میکروکنترلرها وجود دارد؛

انواعی که در آنها پایه های میکرو در چهارطرف قرار دارد. بعضی از آنها عبارتند از:

MLF, PLCC, QFN, TQFP, TSSOP, …

و نوع دوم که در آنها پایه های میکرو در دو طرف تعبیه شده اند، از قبیل:

MSOP, PDIP, SOIC, SSOP, …

در هر یک از این دو گروه، تفاوتهایی در ساختار پایه ها و ساخت آنها مشهود است.

در هر میکرو برای ترتیب دهی، انجام عملیات متوالی و سرعت دهی به اجرای عملیات نیاز به منبعی می باشد که در واقع "نبض" و قلب سیستم ما است. در میکروکنترلر AVR از ٥ منبع مختلف، می توان جهت تولید کلاک استفاده کرد. این منابع عبارتند از:

اسیلاتور RC کالیبره شدهٔ داخلی[۲]، اسیلاتور کریستالی[۳]، اسیلاتور کریستالی فرکانس پایین[٤]، اسیلاتور RC خارجی[٥] و کلاک خارجی[٦].

---

[۱] حافظه ROM داخلی میکرو، که برنامه بر روی آن قرار می گیرد.

[۲] Calibrated Internal RC Oscillator

منبع کلاک پیش فرض میکرو، اسیلاتور RC داخلی ۱MHz با بیشترین زمان شروع می باشد.

[۳] External Crystal/Ceramic Resonator

[٤] External Low-frequency Crystal

[٥] External RC Oscillator

[٦] External Clock

میکروکنترلرهای AVR، بخشها و امکانات جانبی زیادی از قبیل حافظهٔ EEPROM[۱] داخلی، حافظهٔ SRAM[۲] داخلی، تایمر-کانتر[۳]، PWM[٤]، اسیلاتور داخلی، مقایسه کنندهٔ آنالوگ داخلی، RTC[٥]، مبدل آنالوگ به دیجیتال[٦]، Watchdog[۷]، فیوز بیتها[۸]، انواع ارتباطات سریال، ارتباط JTAG[۹] و ... دارند که آنها را از دیگر میکروکنترلرها متمایز و پر استفاده تر نموده است.

یکی از ویژگیهای جالب میکروکنترلرهای AVR، در مقایسه با دیگر میکروکنترلرها، فیوزبیتها هستند. این فیوزها در زمان برنامه ریزی تراشه، قابل برنامه ریزی هستند و هر یک می تواند یکی از امکانات جانبی میکروکنترلر را فعال یا غیرفعال کند و یا نحوهٔ استفاده از آن را مشخص نماید. همچنین فیوز بیتها با پاک کردن میکرو، پاک نشده و حتی می توان آنها را قفل نمود.

از مزایای دیگر این میکروکنترلر طراحی ویژهٔ آن است که امکان کار با زبانهای سطح بالا، بدون افزایش بیش از حد کدهای برنامه را فراهم می کند. از آنجایی که زبان برنامه نویسی C به عنوان یک زبان سطح بالا و همچنین نزدیک به زبان اسمبلی(زبان قابل فهم برای ریزپردازنده ها) می تواند به بهترین وجه برای کار با این میکروکنترلر استفاده شود، لذا در پروژهٔ ربات خط یاب فعلی از این میکروکنترلر و زبان برنامه نویسی C برای برنامه ریزی آن استفاده شده است.

در این ربات هوشمند از میکروکنترلر ATMEGA۳۲L با اسیلاتور کریستالی خارجی ۱٦MHz استفاده شده است؛ در زیر به بررسی خصوصیات و ویژگیهای اختصاصی این میکرو ازخانوادهٔ AVR و یا به طور کلی میکروکنترلر سری ATmega۳۲ می پردازیم.

این میکرو دارای ۱۳۰ دستور قدرتمند است که اکثر آنها در یک سیکل ساعت اجرا می شوند. در ادامه نگاهی به وضعیت حافظهٔ داخلی و امکانات جانبی این میکروی قدرتمند داریم:

<u>حافظه</u>

- ۳۲ KB حافظهٔ Flash قابل برنامه ریزی داخلی.
- مجهز به قسمت Boot Loader.
- ۱۰۲٤ بایت حافظهٔ EEPROM داخلی.
- ۲ کیلو بایت حافظهٔ SRAM داخلی.

---

[۱] Electrically Erasable Programmable Read Only Memory : حافظهٔ قابل برنامه ریزی با امکان پاک کردن اطلاعات به صورت الکتریکی

[۲] Static Random Access Memory :
حافظهٔ RAM داخلی میکرو برای ذخیره سازی موقت داده ها

[۳] Timer-Counter : زمان سنج و شمارنده

[٤] Pulse Width Modulation : مدولاسیون پهنای باند

[٥] Real Time Clock : واحدی برای تولید یک ساعت و تقویم دقیق که حتی با قطع برق نیز تنظیم آن به هم نمی ریزد

[٦] Analog to Digital Converter (ADC) (A/D)

[۷] سگ نگهبان؛ بخشی از میکرو می باشد که در صورت ایجاد مشکل در اجرای برنامه، سبب Reset میکرو می گردد. به این معنی که تمامی رجیسترهای داخلی در یک حالت پیش فرض قرار می گیرند و برنامه از بردار Reset اجرا می شود.

[۸] Fuse Bit
قسمتی از حافظه Flash که بعضی از امکانات میکرو را تنظیم می کند و با پاک کردن Flash از بین نمی رود.

[۹] JTAG (IEEE۱۱٤۹,۱) یک روش ارتباط با تراشه های FPGA است که برای ارتباط با تراشه های AVR، برنامه ریزی آنها و انجام دیباگ نیز مورد استفاده قرار می گیرد.

امکانات جانبی

- ارتباط JTAG شامل اسکن کردن امکانات جانبی، حمایت از دیباگ کردن تراشه و برنامه ریزی حافظه های Flash و EEPROM، فیوزبیتها و بیتهای قفل.
- دو تایمر-کانتر ۸ بیتی با تقسیم کنندهٔ فرکانسی مجزا و دارای مد Compare.
- یک تایمر-کانتر ۱۶ بیتی با تقسیم کنندهٔ فرکانسی مجزا و دارای مدهای Compare و Capture.
- دارای RTC با با اسیلاتور مجزا.
- چهار کانال PWM.
  ۸ کانال ADC، ۱۰ بیتی.
  ۸ کانال تک پایه.
  ۷ کانال تفاضلی در بسته بندی TQFP.
  ۲ کانال تفاضلی با بهرهٔ قابل تنظیم در x۱، x۱۰ و x۲۰۰.
- ارتباط سریال دو سیمه (I^2C) (TWI)[۱].
- USART[۲] سریال قابل برنامه ریزی.
- ارتباط سریال USI به صورت Master/Slave.
- Watchdog قابل برنامه ریزی با اسیلاتور مجزا.
- مقایسه کنندهٔ آنالوگ داخلی.
- اسیلاتور RC داخلی کالیبره شده.
- دارای شش مد Sleep[۳]: Extended Standby, Standby, Powe-down, Power-save, ADC Noise Reduction, Idle.

ولتاژهای عملیاتی و فرکانسهای کاری میکروکنترلر سری ATmega۳۲ طبق جدول زیر می باشد.

میکروکنترلر	ولتاژهای عملیاتی	فرکانسهای کاری
ATmega۳۲	٤,٥ _ ٥,٥ v	۰ _ ١٦ MHz
ATmega۳۲L	٢,٧ _ ٥,٥ v	۰ _ ٨ MHz

این میکروکنترلر دارای دو بایت فیوز است. چگونگی قرارگیری فیوزبیتها در این دو بایت، عملکرد و وضعیت پیش فرض هر یک از فیوزبیتها در جدول زیر نشان داده شده است.

---

[۱] Two Wire Interface / Inter-In Circuit
پروتکل ارتباط سریال دو سیمه از شرکت Philips

[۲] Universal Synchronous/Asynchronous serial Receiver & Transmitter : ارتباط سریال سنکرون/آسنکرون جهانی

[۳] Sleep Mode : حالتی که در آن، قسمتهایی از میکرو برای کاهش توان مصرفی، غیر فعال می شوند

Fuse High Byte

Fuse High Byte	Bit No.	Description	Default Value
OCDEN	7	Enable OCD	1 (unprogrammed, OCD disabled)
JTAGEN	6	Enable JTAG	0 (programmed, JTAG enabled)
SPIEN	5	Enable SPI Serial Program and Data Downloading	0 (programmed, SPI prog. enabled)
CKOPT	4	Oscillator options	1 (unprogrammed)
EESAVE	3	EEPROM memory is preserved through the Chip Erase	1 (unprogrammed, EEPROM not preserved)
BOOTSZ1	2	Select Boot Size (see Table 100 for details)	0 (programmed)
BOOTSZ0	1	Select Boot Size (see Table 100 for details)	0 (programmed)
BOOTRST	0	Select reset vector	1 (unprogrammed)

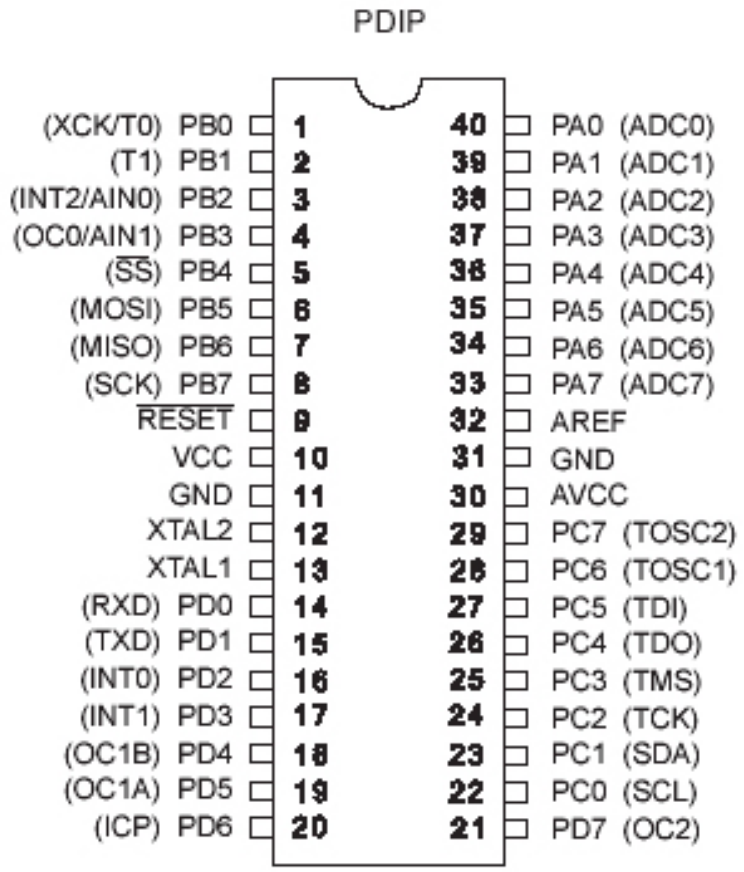
Fuse Low Byte

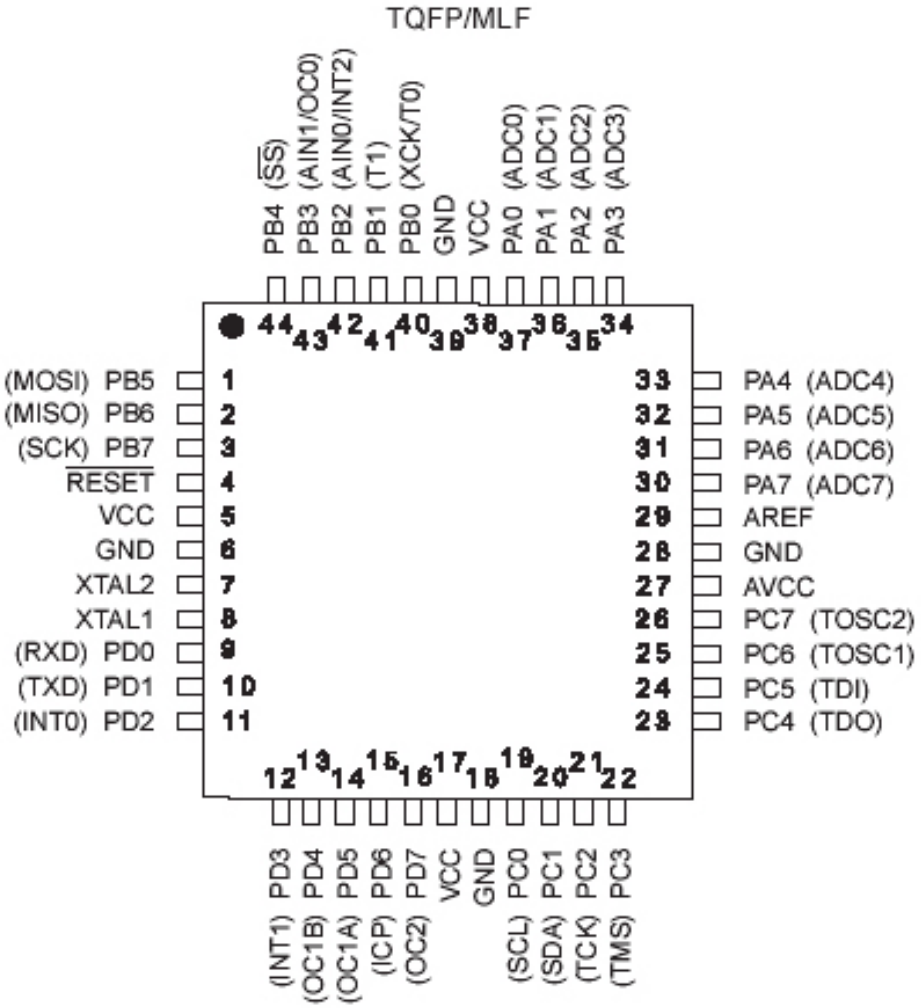
Fuse Low Byte	Bit No.	Description	Default Value
BODLEVEL	7	Brown-out Detector trigger level	1 (unprogrammed)
BODEN	6	Brown-out Detector enable	1 (unprogrammed, BOD disabled)
SUT1	5	Select start-up time	1 (unprogrammed)
SUT0	4	Select start-up time	0 (programmed)
CKSEL3	3	Select Clock source	0 (programmed)
CKSEL2	2	Select Clock source	0 (programmed)
CKSEL1	1	Select Clock source	0 (programmed)
CKSEL0	0	Select Clock source	1 (unprogrammed)

سه نوع بسته بندی برای میکروکنترلر سری ATmega۳۲ طراحی شده است:

- ٤٠ پایه در نوع PDIP
- ٤٤ پایه در نوع TQFP
- ٤٤ پایه در نوع MLF

ترکیب این بسته بندیها مطابق اشکال زیر می باشد.


PDIP
(XCK/T0) PB0 1
(T1) PB1 2
(INT2/AIN0) PB2 3
(OC0/AIN1) PB3 4
(SS) PB4 5
(MOSI) PB5 6
(MISO) PB6 7
(SCK) PB7 8
RESET 9
VCC 10
GND 11
XTAL2 12
XTAL1 13
(RXD) PD0 14
(TXD) PD1 15
(INT0) PD2 16
(INT1) PD3 17
(OC1B) PD4 18
(OC1A) PD5 19
(ICP) PD6 20
40 PA0 (ADC0)
39 PA1 (ADC1)
38 PA2 (ADC2)
37 PA3 (ADC3)
36 PA4 (ADC4)
35 PA5 (ADC5)
34 PA6 (ADC6)
33 PA7 (ADC7)
32 AREF
31 GND
30 AVCC
29 PC7 (TOSC2)
28 PC6 (TOSC1)
27 PC5 (TDI)
26 PC4 (TDO)
25 PC3 (TMS)
24 PC2 (TCK)
23 PC1 (SDA)
22 PC0 (SCL)
21 PD7 (OC2)



میکروکنترلر ATmega۳۲ دارای ۳۲ خط ورودی-خروجی قابل برنامه ریزی می باشد. این ۳۲ خط در چهار گروه ۸ تایی تقسیم شده اند، پورتهای D, C, B, A. که هر یک از این پایه ها را می توان جهت کاربرد دیگری بجز I/O دوطرفه نیز مورد استفاده قرار داد. این وظایف به اختصار در جداول زیر آورده شده است.

Port A Pins Alternate Functions

Port Pin	Alternate Function
PA7	ADC7 (ADC input channel 7)
PA6	ADC6 (ADC input channel 6)
PA5	ADC5 (ADC input channel 5)
PA4	ADC4 (ADC input channel 4)
PA3	ADC3 (ADC input channel 3)
PA2	ADC2 (ADC input channel 2)
PA1	ADC1 (ADC input channel 1)
PA0	ADC0 (ADC input channel 0)

Port B Pins Alternate Functions

Port Pin	Alternate Functions
PB7	SCK (SPI Bus Serial Clock)
PB6	MISO (SPI Bus Master Input/Slave Output)
PB5	MOSI (SPI Bus Master Output/Slave Input)
PB4	$\overline{SS}$ (SPI Slave Select Input)
PB3	AIN1 (Analog Comparator Negative Input) OC0 (Timer/Counter0 Output Compare Match Output)
PB2	AIN0 (Analog Comparator Positive Input) INT2 (External Interrupt 2 Input)
PB1	T1 (Timer/Counter1 External Counter Input)
PB0	T0 (Timer/Counter0 External Counter Input) XCK (USART External Clock Input/Output)

Port C Pins Alternate Functions

Port Pin	Alternate Function
PC7	TOSC2 (Timer Oscillator Pin 2)
PC6	TOSC1 (Timer Oscillator Pin 1)
PC5	TDI (JTAG Test Data In)
PC4	TDO (JTAG Test Data Out)
PC3	TMS (JTAG Test Mode Select)
PC2	TCK (JTAG Test Clock)
PC1	SDA (Two-wire Serial Bus Data Input/Output Line)
PC0	SCL (Two-wire Serial Bus Clock Line)

Port D Pins Alternate Functions

Port Pin	Alternate Function
PD7	OC2 (Timer/Counter2 Output Compare Match Output)
PD6	ICP (Timer/Counter1 Input Capture Pin)
PD5	OC1A (Timer/Counter1 Output Compare A Match Output)
PD4	OC1B (Timer/Counter1 Output Compare B Match Output)
PD3	INT1 (External Interrupt 1 Input)
PD2	INT0 (External Interrupt 0 Input)
PD1	TXD (USART Output Pin)
PD0	RXD (USART Input Pin)

در این ربات هوشمند از بسته بندی PDIP استفاده شده است. در زیر به توضیح هر یک از پایه های استفاده شده از این میکرو، می پردازیم.
علاوه بر ٤ پورت فوق، ٨ پایهٔ دیگر نیز در بسته بندی PDIP وجود دارد که در تمامی خانواده های دیگر میکروکنترلر AVR نیز همواره استفاده از آنها ضروری است. این پایه ها که در حقیقت برای راه اندازی میکروکنترلر بکار می روند، به شرح زیر می باشند.

VCC

این پایه ولتاژ تغذیهٔ میکرو است که باید در میکروکنترلر ATmega۳۲L به ولتاژی در محدودهٔ ۲,۷ V تا ۵,۵ V متصل شود.

GND

این پایه به زمین متصل می شود.

RESET

این پایه، یک ورودی است که اگر برای حداقل ۱,۵µs صفر (منطقی) شود، میکرو Reset می گردد.

AVCC

این پایه منبع تغذیهٔ مربوط به پورت A و مبدل آنالوگ به دیجیتال می باشد و حتی در صورتی که از ADC استفاده نشود هم، باید حتماً به VCC بطور مستقیم یا توسط یک فیلتر میان گذر متصل گردد.

AGND

این پایه به زمین متصل می شود.

AREF

این پایه ولتاژ مرجع آنالوگ برای مبدل آنالوگ به دیجیتال می باشد.
در صورتی که برای منبع کلاک میکرو از "اسیلاتور کریستالی" یا "اسیلاتور کریستالی فرکانس پایین" یا "اسیلاتور RC خارجی" و یا "کلاک خارجی" استفاده شود، پایه زیر نیز مورد استفاده قرار می گیرد.

XTAL۱

این پایه نیز پایه ای ورودی است که به تقویت کنندهٔ اسیلاتور متصل می شود و به مدار

تولید کلاک تراشه نیز وارد می گردد.
اگر از "اسیلاتور کریستالی" یا "اسیلاتور کریستالی فرکانس پایین" برای تأمین کلاک لازم میکرو استفاده شود، پایهٔ زیر نیز باید متصل شود.

XTAL۲

این پایه، پایهٔ خروجی است که از تقویت کنندهٔ اسیلاتور خارج می شود.
در این ربات به جهت استفاده از اسیلاتور کریستالی هر دو پایهٔ XTAL۱ و XTAL۲ بسته شده اند.

دیگر پایه های بکارگرفته شده در مدار این ربات عبارتند از:

PD۰~PD٦

این پایه ها به عنوان پایه های ورودی مورد استفاده قرار گرفته اند. مقایسه کننده ها پس از تشخیص رنگ دیده شده توسط سنسورها، اطلاعات هر ۷ سنسور را از طریق این ۷ پایه به میکرو انتقال می دهند.

PA۰~PA۳

این پایه ها به عنوان خروجی استفاده شده اند. از این ٤ پایه برای انتقال دیتا و فرمان از میکرو به درایور موتور سمت راست استفاده شده است.

PC٤~PC۷

از این ٤ پایه به عنوان خروجی دیتا از میکرو به درایور موتور سمت چپ استفاده شده است.

SCK[۱]

این پایه همان پایهٔ PB۷ است که در اینجا نه به عنوان I/O بلکه به عنوان پایهٔ واسط کلاک خروجی Master و کلاک ورودی Slave، برای برنامه ریزی میکرو بکار رفته است.

MISO[۲]

این پایه، همانند پایهٔ قبل از کاربرد دومش بهره گرفته شده است. این پایه واسط ورودی دادهٔ Master و خروجی دادهٔ Slave در برنامه ریزی میکرو به روش ISP[۳] می باشد. نام دیگر این پایه، PB٦ می باشد.

MOSI[٤]

این پایه که از آن به عنوان کاربرد دوم پایهٔ PB٥ استفاده می شود، در حقیقت ورودی دادهٔ Slave و خروجی دادهٔ Master در برنامه ریزی میکرو به روش ISP است.
برای برنامه ریزی میکرو، سه پایهٔ اخیر باضافهٔ پایه های GND, VCC, RESET به عنوان واسطهٔ برنامه ریزی از کامپیوتر مورد استفاده قرار می گیرند. برای این منظور و به جهت ایمنی و یکطرفه بودن جریان اطلاعات، می توان از بافری مانند "٧٤HC٢٤٤" و یا "٧٤ALS٢٤٤" در میان این ارتباط که توسط کابل ISP میسر می شود، استفاده نمود.

---

[۱] Serial Clock
Master (کامپیوتر) پالسهای کلاک لازم برای انتقال اطلاعات بین خود و Slave (میکرو) را روی این پایه تولید می کند.
[۲] Master In Slave Out(MISO)
داده ها به کمک این پایه از Master به Slave شیفت می یابند.
[۳] In System Programming : برنامه ریزی در داخل مدار
[٤] Master Out Slave In(MOSI)
داده ها به کمک این پایه از Slave به Master شیفت می یابند.

۳-۵) واسط برنامه ریزی

به طور کلی برای برنامه ریزی میکروکنترلرهای AVR می توان از چهار روش مختلف استفاده کرد.

موازی (Parallel)

در این روش که برای میکروهای قدیمی تر رایج است، میکرو باید در سوکت پروگرامر قرار داده شده وبرای شروع برنامه ریزی، لازم است تا ولتاژ ۱۲ v+ به پایهٔ Reset اعمال شود. این روش تنها روشی است که برای تمام انواع خانوادهٔ AVR قابل اجرا است.

JTAG

برای میکروهایی که بیزان حافظهٔ Flash در آنها از ۱٦ KB بیشتر باشد، امکان استفاده از ارتباط JTAG برای برنامه ریزی، دیباگ کردن تراشه و چک کردن امکانات جانبی وجود دارد.

خود برنامه ریزی (Self Programming)

در میکروهای پیشرفته، قابلیتی به نام Boot Loader به آنها اضافه شده، که میکرو به کمک این قسمت و توسط هرگونه ارتباط جانبی(از قبیل Parallel, SPI, TWI,...) می تواند به صورت خودکار خودش را برنامهریزی کند.

ISP (In System Programming)

این روش در برنامه ریزی میکروکنترلرهای خانوادهٔ AVR به دلیل سادگی مورد توجه قرار گرفته است. در این روش میکرو در مدار اصلی خود قرار گرفته و به صورت مستقیم برنامه ریزی می شود. این روش یکی از جدیدترین و بهترین روشهای برنامه ریزی میکروکنترلر می باشد.

میکروی بکار رفته در این پروژه دارای ۳۲ کیلو بایت حافظهٔ Flash با قابلیت برنامه ریزی درون مداری می باشد؛ از این رو، از روش ISP برای برنامه ریزی آن استفاده شده است. برای برنامه ریزی کافیست مدار Programmer را بین کامپیوتر و میکرو ببندیم. این مدار تنها از یک کابل ISP تشکیل شده است. که یک سر آن کانکتور DB۲۵ و در سر دیگر آن یک سوکت ٦ تایی مخابراتی وجود دارد(این مدار یا کابل را خود نیز می توانیم ببندیم). ابتدا کانکتور DB۲۵ را به کامپیوتر متصل نموده و پین مخابراتی را نیز به ٦ پایهٔ میکرو(معرفی و توضیح این ٦ پایه در بخش قبل آمده است) وصل می کنیم؛ سپس تغذیهٔ میکرو را بسته و فرمان(دکمهٔ) "Program" را در نرم افزار برنامه نویسی که برنامهٔ میکرو را با آن نوشته، کامپایل و اسمبل[1] نموده ایم، انتخاب می کنیم. برنامه ریزی ظرف مدت کوتاهی که معمولاً چند ثانیه بیشتر طول نمی کشد(بسته به حجم برنامه)، به اتمام رسیده و بلافاصله اجرای کدها توسط میکرو آغاز می گردد.

[1] Compile & Make

٦-٣) حرکت ربات

راهبری و نیروی حرکت این ربات توسط دو موتور پله ای[١] می باشد. موتور پله ای یا استپ موتور از یک هستهٔ متحرک مغناطیسی دائمی که به آن روتور یا شفت نیز می گویند و یک بخش ثابت به نام استاتور که به همراه سیم پیچهایش روتور را احاطه می کند، تشکیل شده است. از موتورهای پله ای می توان برای جابجایی، حرکت، تعیین موقعیت و بسیاری از کارهای دیگر که در آنها کنترل دقیق موقعیت یک محور، اهرم یا قسمت متحرک یک دستگاه مکاترونیکی مورد نیاز است، استفاده کرد. از جمله کاربردهای رایج می توان به درایور دیسکت، پرینتر، اسکنر و رباتیک اشاره نمود.

اساس عملکرد یک موتور پله ای تفاوت زیادی با یک موتور DC یا AD ندارد. تنها تفاوت موجود در نحوهٔ حرکت محور آن (روتور) است. در این موتورها با اعمال توان به سیمپیچها، به طور نوبتی و با ترتیبی خاص، روتور به صورت پله ای(مرحله به مرحله) حرکت کرده و می چرخد؛ به عبارت دیگر موتورهای پله ای برخلاف موتورهای DC که به وسیلهٔ تغییر جریان عبوری از آنها کنترل می شوند، از نظر عملکرد دیجیتال می باشند.

دو مزیت مهم موتورهای پله ای نسبت به دیگر موتورهای متداول، یکی دقت در چرخش و کنترل آنها و دیگری قدرت بازدارندهٔ آنها از چرخش در مواقعی خاص که نیاز به ترمزی قوی لازم است، می باشد.

در هنگام استفاده از یک موتور پله ای، باید هم ویژگیهای مکانیکی و هم مشخصه های الکتریکی موتور را شناخت. در ادامه به توضیح این مشخصه ها می پردازیم.

موتورهای پله ای مختلف مشخصات گشتاور، ولتاژ و جریان و تعداد پله در دور متفاوتی دارند. تعداد پله در دور، میزان دقت موتور را نشان می دهد. برای بدست آوردن زاویهٔ هر پله می توان، ٣٦٠ را بر تعداد پله در دور تقسیم نمود. و همچنین با داشتن زاویهٔ پله(که غالباً اینچنین است)، می توان با تقسیم ٣٦٠ بر آن، تعداد پله در دور را محاسبه نمود. زاویهٔ پله در موتورهای مختلف در محدودهٔ ٠,٨° تا ٩٠ درجه می باشد. تعداد پله در دور و زاویهٔ هر پله را در چند موتور مختلف در جدول زیر مشاهده می کنید.

پله در دور	زاویه پله[٢]
٥٠٠	٠,٧٢
٢٠٠	١,٨
١٨٠	٢,٠
١٤٤	٢,٥
٧٢	٥,٠
٤٨	٧,٥
٢٤	١٥
٤	٩٠

آهنگ پالسها، سرعت موتور را تعیین می کنند. اگر از یک موتور با زاویهٔ پله ١,٨° استفاده شود و تعداد ٢٠٠ پالس در هر ثانیه به موتور اعمال کنیم، این موتور با سرعت ١ دور در ثانیه یا ٦٠ دور بر دقیقه[٣] می چرخد. بدین ترتیب با دانستن زاویهٔ پله، محاسبهٔ rpm به

[١] Stepper Motor
[٢] Step Angle
[٣] Round Per Minute(rpm)

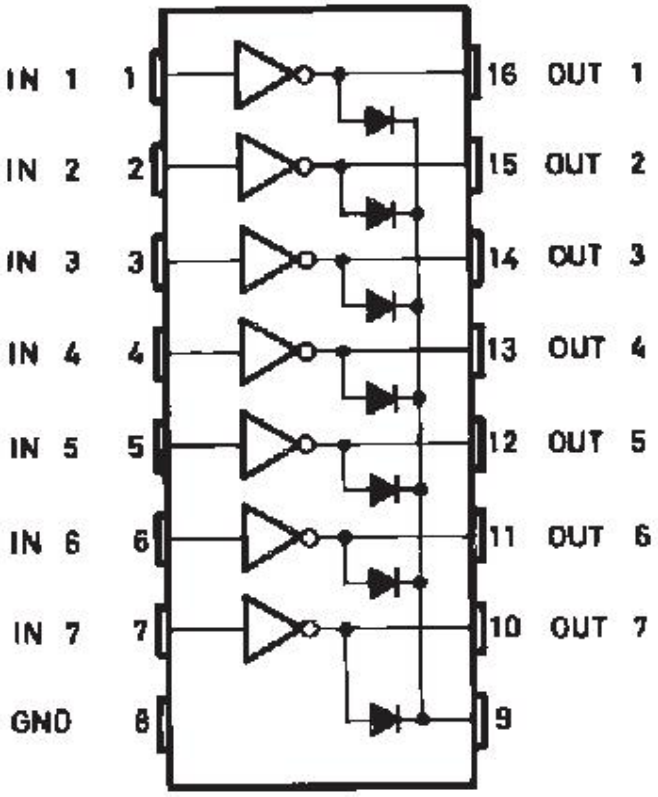
سادگی امکان پذیر خواهد شد. موتورهای پله ای برای کاربردهای با سرعت زیاد مورد استفاده قرار نمی گیرند. حداکثر سرعت پیشنهادی در محدودهٔ ۲ تا ۳ دور در ثانیه (۱۲۰ rpm یا ۱۸۰ rpm) می باشد. در این نوع موتور با افزایش سرعت، گشتاور کاهش می یابد.

گشتاور ایجاد شده بوسیلهٔ موتور پله ای زیاد نیست. یک موتور پله ای متداول در عمل قادر است فقط به اندازهٔ چند گرم بر هر سانتی متر گشتاور ایجاد کند. بنابراین در کاربردهایی که گشتاور زیاد مورد نیاز است، افزودن جعبه دنده به موتور توصیه می شود. از آنجا که گشتاور در سرعتهای زیاد افت پیدا می کند، بهتر است این نوع موتور برای حالتهای با سرعت کم مورد استفاده قرار گیرد.

یکی از ویژگیهای جالب و مفید این نوع موتورها، اثر ترمزی آنها می باشد. این حالت موقعی به وقوع می پیوندد که پس از اعمال پالس، جریان در سیم پیچ برقرار نگه داشته شود. که در نتیجهٔ آن محور موتور قفل شده و موتور قادر به چرخیدن نخواهد بود، گویی با ترمز الکترونیکی قوی متوقف شده است.

موتورهای پله ای معمولاً دارای ولتاژ تغذیهٔ نامی ۵v، ۶v یا ۱۲v می باشند. برخلاف موتورهای DC، اضافه ولتاژ بر روی سیم پیچهای یک موتور پله ای توصیه نمی گردد. اضافه ولتاژی بیش از ۳۰٪ ولتاژ نامی می تواند باعث سوختن سیم پیچها شود. میزان جریان نامی به نوع کاربرد (اندازه و گشتاور) بستگی دارد. انواع متداول جریانی در محدودهٔ ۵۰mA تا مقداری بیش از ۱A مصرف می کنند. هرچقدر مقدار ولتاژ و جریان بیشتر باشد، مقدار گشتاور نیز بیشتر خواهد بود. به هنگام طراحی یک منبع تغذیه برای دستگاهی که از موتور پله ای استفاده می کند توجه به این نکته که جریان نامی به ازای هر سیم پیچ ذکر شده باشد اهمیت دارد. بنابراین منبع تغذیه باید قادر به تأمین حداقل دو برابر جریان هر سیم پیچ در موتورهای دو فاز و در مورد موتورهای نوع چهار فاز حداقل هشت برابر جریان هر سیم پیچ باشد. موتورهای پله ای استفاده شده در این ربات ۵v, ۲۰۰mA می باشند.

از آنجا که جریان مورد نیاز سیم پیچهای موتور پله ای زیاد می باشد و میکروکنترلر قادر به تأمین آن نیست، لذا برای راه اندازی آنها از ترانزیستورهای با توان متوسط یا زیاد و یا از ICها استفاده می کنند. برای راه اندازی و کنترل موتورهای پله ای از ICهای بسیاری که بدین منظور طراحی شده اند می توان سود جست. این ICها یا از نوع واحدهای ساده می باشند که فقط شامل چهار ترانزیستور برای راه اندازی موتور می باشند و یا از انواع واحدهای پیچیده تر که شامل مولدهای پله، مبدلها و بسیاری عملکردهای دیگرند. یکی از این ICها که در این ربات نیز به عنوان واسطهٔ میکرو و موتور استفاده شده است "ULN۲۰۰۳" می باشد. IC-ی دیگری که به این منظور می توان استفاده کرد، ULN۲۰۰۲ می باشد که هر دو از یک خانواده هستند. موتورهای پله ای با جریان مورد نیاز حداکثر ۲۰۰mA تا ۳۰۰mA را می توان مستقیماً با ICهای سری ULN۲۰۰x راه اندازی نمود. مدار درونی این ICها از ترانزیستورهای قدرت دارای مقاومت بیس و همچنین دیود (محافظ در برابر Spikeهای ولتاژ ایجاد شده به هنگام روشن و خاموش شدن سیم پیچهای موتور پله ای) تشکیل شده است. تفاوت دو IC فوق در این است که مقاومتهایی با مقادیر متفاوت در ورودیهای آنها قرار داده شده است. در IC شمارهٔ ULN۲۰۰۳، مقدار مقاومت ۲۷۰۰Ω می باشد که مشخصهٔ IC را با مدارات منطقی TTL سازگار می سازد. در IC شمارهٔ ULN۲۰۰۲ مقاومتهای موجود در ورودی هر طبقه ۱۰٫۵KΩ می باشد که مشخصهٔ IC را با مدارات منطقی CMOS سازگار می کند. این ICها به Heatsink احتیاج نداشته و می توان آنها را با ولتاژ حداکثر ۱۲v تغذیه نمود. در شکل زیر مدار داخلی ULN۲۰۰۳ آمده است.

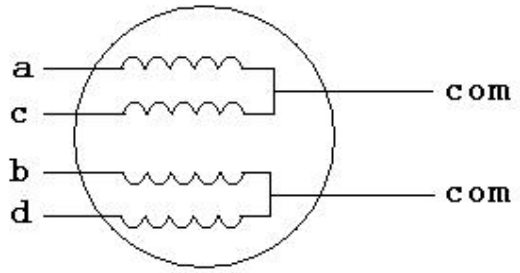


در خروجیهای این IC و به عبارتی در ورودی موتور پله ای برای کنترل جریان سیم پیچها، از مقاومتهای Pull-Up با مقدار ٤,٧KΩ استفاده شده است.

موتورهای پله ای دارای سه نوع اصلی هستند: آهن ربای دائمی، رلوکتانس متغیر و هایبرید. عملکرد هر موتور پله ای را حالت سازمان یافتهٔ سیم پیچهای داخلی آن موتور تعیین می کند. متداولترین نوع موتور پله ای، نوع چهار فاز آن است. البته انواع دو فاز و شش فاز نیز وجود دارند.

در این ربات خط یاب از دو موتورپله ای چهار فاز استفاده شده است. در داخل این موتور چهار سیم پیچ استاتور وجود دارند که هر زوج سیم پیچ دارای یک اتصال مشترک و در نتیجه یک سیم دیگر می باشند. مشخصهٔ موتورهای پله ای چهار فاز نیز ٦ سیمه بودن آنها است. برای تشخیص سرهای مشترک و یا زوج سیم پیچهای متصل به هم از یک اهم متر استفاده می کنیم. بدین طریق که با اتصال دو سر اهم متر به دو سیم از ٦ سیم موتور(مثلاً a,b)، اگر مقدار فرضی X (تقریباً بزرگ) ظاهر شد، این دو سیم قطعاً به هم متصل هستند که برای تعیین سر مشترک باید مقاومت بین یکی از این سیمها (مثلاً a) را با چهار سیم باقیمانده اندازه گیری کنیم؛ تنها یکی از این چهار سیم (مثلاً c)است که مقداری(مخالف و نه نزدیک به صفر) را نشان می دهد؛ اگر این مقدار برابر X باشد، آنگاه سیم a سر مشترک و دو سیم b,c یکی از زوج سیم پیچهای به هم متصل هستند؛ و اگر مقدار نمایش داده شده (X)۲ باشد، آنگاه سیم b سر مشترک و دو سیم a,c یکی از زوج سیم پیچهای به هم متصل هستند که در این صورت

اتصال بین دو سیم c,b نیز مقداری برابر X را نشان خواهد داد؛ و اگر مقدار اندازه گیری شده (X/۲)=½(X) باشد، در آن صورت سیم c سر مشترک و دو سیم a,b یکی از زوج سیم پیچهای به هم متصل می باشند. طبیعتاً سه سیم باقیمانده نیز به هم متصل اند که برای تشخیص و تمیز سر مشترک از دو سر دیگر به روش فوق عمل می کنیم. در شکل زیر نحوهٔ قرار گیری سیم پیچهای استاتور یک نمونه موتور پله ای چهار فاز نمایش داده شده است.



در حالت عملکرد معمولی سیمهای مشترک به قطب مثبت منبع تغذیه متصل می شوند و سیمهای دیگر برای مدت زمانی بسیار کوتاه به زمین وصل می شوند. هر بار که موتور تغذیه شود، محور موتور باندازهٔ کسری از یک دور می چرخد. به منظور این که موتور به صورت پیوسته و مناسب بچرخد، باید سیم پیچها با یک رشته خاص از پالسها یا موجها تغذیه شوند.

برای راه اندازی موتور پله ای، به طور کلی دو روش Full-Step و Half-Step وجود دارند که روش Full-Step خود به دو روش "تک فاز" و "دو فاز" تقسیم می گردد.

راه اندازی به روش تک فاز

در این روش در هر لحظه تنها یک جفت از سیم پیچهای استاتور با قطبهای ناهمنام مغناطیسی فعال می شوند که در این صورت قطبهای ناهمنام در استاتور و روتور همدیگر را جذب می کنند. جهت راه اندازی موتور به صورت تک فاز طبق جدول زیر موتور را تغذیه می نماییم. با اجرای هر یک از این حالتها، روتور یک پله جابجا می شود.

سیم پیچ d	سیم پیچ c	سیم پیچ b	سیم پیچ a		
۰	۰	۰	۱	در جهت ساعت ↓	خلاف جهت ساعت ↑
۰	۰	۱	۰		
۰	۱	۰	۰		
۱	۰	۰	۰		

راه اندازی به روش دو فاز

در این روش در هر لحظه هر دو جفت از سیم پیچهای استاتور با قطبهای ناهمنام فعال می شوند که در این صورت قطبهای روتور در جهت برآیند قطب مخالف استاتور قرار می گیرند. تفاوت این روش با روش قبل این است که در این حالت گشتاور ایجاد شده افزایش می یابد و برای کاربردهایی که به قدرت بیشتری نیاز دارند مفید است. در این حالت برای راه اندازی موتور از جدول زیر استفاده می شود. با اجرای هر یک از این حالتها شفت موتور یک پله جابجا می شود.

سیم پیچ d	سیم پیچ c	سیم پیچ b	سیم پیچ a		
۰	۰	۱	۱	در جهت ساعت ↓	خلاف جهت ساعت ↑
۰	۱	۱	۰		
۱	۱	۰	۰		
۱	۰	۰	۱		

راه اندازی به روش Half-Step

در این حالت در هر بار تغییر وضعیت سیم پیچهای استاتور، میزان گردش موتور نصف حالتهای قبل خواهد بود. تفاوت این روش با روش راه اندازی دو فاز این است که در اینجا دقت موتور به دو برابر، افزایش پیدا می کند ولی در عوض میزان گشتاور تولید شده در این حالت به میزان ۱۵ تا ۳۰ درصد کاهش می یابد. در این حالت از جدول زیر برای راه اندازی موتور استفاده می شود.

سیم پیچ d	سیم پیچ c	سیم پیچ b	سیم پیچ a		
۰	۰	۰	۱	در جهت ساعت ↓	خلاف جهت ساعت ↑
۰	۰	۱	۱		
۰	۰	۱	۰		
۰	۱	۱	۰		
۰	۱	۰	۰		
۱	۱	۰	۰		
۱	۰	۰	۰		
۱	۰	۰	۱		

در سه جدول فوق ترتیب سیم پیچها همانند شکل نمونه ای است که برای موتور پله ای آمده است.

در این ربات خط یاب از روش راه اندازی Full-Step به صورت دو فاز استفاده شده است. که توضیحات بیشتر را به بخش ۵-۵ موکول می کنیم.

۳-۷) قطعات بکار رفته در مدار ربات هوشمند

در این بخش می خواهیم قطعات بکار رفته در مدار الکترونیکی این ربات هوشمند را فهرست وار عنوان کنیم:

- ۷ سنسور مادون قرمز تشخیص رنگ سیاه و سفید JK۱۵۰۱۳.
- ۷ مقاومت ۱۰KΩ برای بایاسینگ سنسورها.
- ۷ مقاومت ۲۷۰Ω برای بایاسینگ سنسورها.
- ۲ مقایسه کنندهٔ ولتاژ LM۳۲٤.
- ۱ پتانسیومتر ۲۰KΩ برای تنظیم ولتاژ مرجع مقایسه کننده ها.
- ۷ LED برای نمایش وضعیت هر یک از سنسورها.
- ۷ مقاومت ٤۷۰Ω، Pull-Up برای LEDها.
- ۱ عدد میکروکنترلر ATmega۳۲L.
- ۲ درایور ULN۲۰۰۳.
- ۲ استپ موتور ۲۰۰mA / ۵v.
- ۸ مقاومت ٤,۷KΩ، Pull-Up برای موتورها.
- ۱ کریستال ۱٦MHz.
- ۲ خازن ۲۲pF برای اسیلاتور کریستالی.
- ۱ کلید Reset.
- ۱ خازن ۱۰µF برای مدار Reset.
- ۱ مقاومت ۱۰۰Ω برای مدار Reset.
- ۱ مقاومت ۱۰KΩ، Pull-Up برای مدار Reset.
- ۱ رگولاتور L۷۸۰۵CV برای مدار تغذیه.
- ۱ خازن ۰,۳۳ µF برای مدار تغذیه.
- ۱ خازن ۰,۱ µF برای مدار تغذیه.
- ۱ خازن ۱۰۰۰µF برای مدار تغذیه.
- ۱ دیود برای مدار تغذیه.
- ٤ باطری نیکل-هیدروکسید فلز(Ni-MH)، ۱,۲ v قابل شارژ SONY، ۲۳۰۰mAh سایز AA.
- ۲ عدد جا باطری دوتایی برای باطریهای فوق.
- ۱ عدد کابل ISP که یک سر آن کانکتور DB۲۵ و سر دیگر آن پین ٦ تایی مخابراتی(مادگی) است.
- ۱ پین ٦ تایی مخابراتی(نری)، برای اتصال کابل ISP به مدار.
- ۱ رابط(کابل) ۸ تایی که یک سر آن پین ۸ تایی مخابراتی(مادگی) است، برای ارسال اطلاعات سنسورها به دو مقایسه کننده.
- ۱ پین ۸ تایی مخابراتی(نری) برای اتصال کابل فوق به مدار اصلی.
- ۲ پین ٦ تایی مخابراتی(نری و مادگی) برای اتصال موتورها به درایورها.
- ۱ پین ۲ تایی مخابراتی برای اتصال تغذیه به مدار اصلی.
- سیم مسی به مقدار کافی برای اتصال قطعات و بستن مدار.

مدار اصلی و مدار سنسورها روی دو بورد سوراخ دار "خطی" بسته شده اند، و برای جدا کردن اتصال بین دو قسمت روی یک خط، از کاتر استفاده شده است.

. فصل چهارم .

کنترل

کنترل بسیاری از فرآیندهای پیچیده توسط اپراتورهای با تجربه، به راحتی قابل اجرا است؛ در صورتی که انجام عملیات کنترل مربوطه توسط کنترل کننده های اتوماتیک ممکن است بسیار مشکل یا غیرعملی باشد. یک اپراتور، استراتژی کنترل را اغلب بوسیلهٔ مجوعه ای از قواعد که بخاطر سپردن آنها توسط انسان بسادگی صورت می پذیرد ولی اجرای این قوانین توسط روشهای محاسباتی مرسوم مشکل است، هدایت می کند.

مشکل طراحی کنترل کننده های اتوماتیک، بویژه زمانی که فرآیندها از خواص غیرخطی و تغییرپذیر با زمان برخوردار باشند، زیادتر است. بر این اساس برای کنترل سیستمهای پیچیده، به کنترل فازی توجه خاصی شده است. در کنترل کننده های فازی رفتار وطرز تفکر اپراتور انسانی توسط مجموعه ای از دستورالعملها شبیه سازی می گردد. اساس کنترل فازی، منطق فازی می باشد؛ این منطق نیز بر اساس مجموعه های فازی که در فصل دوم به آن پرداخته شد، پایه گذاری گردیده است.

بخش کنترل در واقع «مغز» هر پروژه در یک سیستم رباتیکی می باشد. تمامی قسمتهای الکتریکی یک ربات، توسط مدارات الکترونیکی کنترل می شوند.

در ادامه انواع کنترل های اصلی موجود برای رباتها را بررسی می نماییم:

کنترل موقعیت[۱]: بازوهای دارای چنگک یا دیگر ساختارهایی که با گرفتن و جابجایی اشیاء سر و کار دارند، باید دارای مدارات کنترل بسیار دقیق به منظور قرار گرفتن در موقعیت صحیح باشند. حرکت یک سر دارای چشم توسط یک بلوک کنترل تک-محور،کنترل می شود.

کنترل سینماتیک[۲]: هر پروژه ای که دارای قسمتهای متحرک باشد به این نوع کنترل نیازمند است. سرعت هر کدام از قسمتهای متحرک باید توسط اینگونه مدارات به دقت تعیین و کنترل شوند. یکی از مهمترین مدارات کنترلی در این گروه، مداری است که سرعت موتور محرک یک ربات را کنترل می کند.

کنترل دینامیک[۳]: کنترل دینامیکی به کنترل حرکت یا کنترل بلادرنگ فرآیندها گفته می شود. بسیاری از قسمتهای یک ربات، نیروهایی را ایجاد می کنند که باید در حین عملکرد، کنترل شوند. هنگامی که دست ربات یک شئ را برمی دارد، استفاده از مدارات کنترلی برای تعیین مقدار نیروی لازم برای نگه داشتن شئ بدون شکستن آن ضروری است. یکی از موارد دشوار برای سازندگان این پروژه ها، ساخت دستی هوشمند است که بتواند یک تخم مرغ را از یک سبد برداشته و بدون اینکه آسیبی به آن برسد در سبد دیگری قرار دهد. چنین اهدافی کنترل دینامیک دقیقی نیاز دارند.

---

[۱] Position Control
[۲] Kinematic Control
[۳] Dynamic Control

کنترل تطبیقی[۱]: هنگامی که لازم است یکی از عملکردهای ربات در حین اجرای یک فرآیند به طور مداوم تغییر یابد، باید از کنترل تطبیقی استفاده شود. اعمال توان بیشتر به موتور به منظور ثابت نگه داشتن سرعت یک ربات، به هنگام حرکت ربات از یک سطح افقی به یک سطح شیب دار یا هنگام جابجایی یک شئ سنگین توسط ربات، نمونه ای از کاربرد این نوع کنترل می باشد.

کنترل خارجی[۲]: زمانی که از یک انسان به عنوان اپراتور برای صدور فرمان انجام تمامی وظایف ربات استفاده می شود، مدارات کنترل خارجی مورد نیاز می باشند. در این حالت انسان به عنوان «مغز» عمل کرده و با استفاده از انواع سنسورها نظیر سنسورهای تصویری به عنوان «حواس»، عملکرد ربات را کنترل می کند. برای انتقال فرامین کنترلی لازم به یک ربات، شخص اپراتور می تواند از انواع مختلفی از «مدارات واسطه»[۳] استفاده نماید. بهترین روش برای ارسال فرامین، بکارگیری مداراتی است که از امواج رادیویی یا مادون قرمز، سیم و حتی فرامین صوتی برای این منظور استفاده می کنند. امروزه در پروژه-های مدرن، مدارات تشخیص صوتی استفاده می شوند که قادر به دریافت مستقیم دستورات از اپراتور هستند. همچنین از یک کامپیوتر نیز می توان به عنوان مدار واسطه برای ارتباط واحد کنترل با ربات استفاده نمود.

در قدیم با نفوذ روشهای کنترلی کلاسیک در زندگی روزمره و صنعت، پیشرفتهای زیادی را در زمینهٔ رباتیک هم شاهد بوده ایم که به دلیل اینکه سبب افزایش دقت کار ربات می شوند، هنوز طرفداران خود را دارند؛ ولی از آنجا که این روشها، پیچیدگیهای محاسباتی و ریاضی زیادی داشته و در ظاهر رفتارهای خشک و تحکم آمیزی را منجر می شوند لذا طراحان را به استفاده از روشهایی غیرکلاسیک ترغیب نموده است.

امروزه با بکارگیری مدارات کنترلی غیرکلاسیک و پیشرفته، رباتهای هوشمندی طراحی می شوند که با استفاده از اطلاعات سنسورهای خود و یا اطلاعاتی که اپراتور انسانی به آنها می دهد، تصمیماتی بسیار مهم که بعضی از آنها برای انسان بسیار مشکل و پیچیده می باشند را اتخاذ کرده و جامع عمل می پوشانند.

۱-٤) روشهای غیر کلاسیک کنترل

بکارگیری روشهای غیرکلاسیک هوشمند در رباتیک را می توان از سال ۱۹۸۳ و با پیشنهاد «ساریداس»[٤] دانست. وی در مقالهٔ خود به لزوم بکارگیری روشهای هوشمند در کنترل ربات پرداخته است. اصولاً روشهای غیرکلاسیک کنترل بر پایهٔ تصمیم گیری و تفکر انسان کار می کنند. اینگونه روشها به مدل ریاضی سیستم کاری ندارند، بلکه با استفاده از ورودی و خروجی سیستم، عمل کنترل را انجام میدهند. از میان این روشها می توان به استفاده از مجموعه های فازی و شبکه های عصبی در طراحی کنترل کننده ها اشاره نمود.

---

[۱] Adaptive Control
[۲] External Control
[۳] Interfaces
[٤] Saridas

٤-٢) کنترل کننده های فازی[1]

در این بخش سعی بر آن داریم که روش کنترل فازی را در حد مورد نیاز و به بیانی ساده آموزش دهیم. از این رو مطالعهٔ این بخش خالی از لطف نبوده و به کلیهٔ دوستانی که برای اولین بار قصد یادگیری آن را دارند، توصیه می گردد.

یکی از روشهای هوشمندی که امروزه در کنترل سیستمها بکار می رود، کنترل فازی است. در سال ۱۹۸۵ با راهیابی روشهای فازی در کنترل ربات، گام جدیدی در بکارگیری روشهای هوشمند در کنترل ربات برداشته شد. معمولاً از کنترل کننده های فازی در مواقعی که دقت بالایی مد نظر نبوده و مدل مناسبی برای سیستم موجود نباشد، همچنین انسان در کنترل سیستم، بهتر و سریعتر از ماشین عمل کند، استفاده می شود. البته یکی از شروط اصلی موفقیت در جایگزین نمودن کنترل کننده های فازی بجای کنترلگرهای انسانی، بیان صحیح تجارب انسان در قالب قوانین شرطی می باشد.

به دلیل اینکه روشهای کنترل فازی متکی به مدل ریاضی سیستم نمی باشند، می توان در مواقعی که ساختار و یا پارامترهای سیستم تحت کنترل، بطور دقیق مشخص نبوده و یا خیلی پیچیده باشند، علوم تجربی انسان در کنترل این سیستمها را در قالب کنترل کننده های فازی پیاده سازی نمود.

به طور کلی دو منطق تعریف شده است؛ Crisp و Fuzzy .

در منطق Crisp کلیهٔ توضیحات در ارتباط با موضوع مورد بحث تماماً بصورت مقادیر عددی بیان می شوند، به طور مثال اگر فرض را در تمام نقاط روی زمین بر این بنا نهیم که ۱ متر قد، یعنی کوتاه قد و ۲ متر قد، یعنی بلند، پس بنا بر قرارداد فوق تنها کسانی که قدشان دقیقاً ۱ متر است، قد کوتاه و همچنین فقط کسانی که قدشان دقیقاً برابر ۲ متر باشد، قدبلند هستند؛ به این ترتیب کسانی که قدشان کمتر از ۱ متر(حتی ...۹۹,۹۹۹ سانتیمتر) یا بین ۱ متر و ۲ متر(حتی ۱...۱,۰۰۰ متر یا ...۱,۹۹۹ متر) و یا بیش از ۲ متر(حتی ۱...۲,۰۰۰ متر) باشد، نه قد کوتاه هستند و نه قد بلند به حساب می آیند. به بیانی دیگر هر کس که قد کوتاه باشد، دارای قدی برابر ۱ متر و هر کس که قد بلند باشد یعنی قدش ۲ متر است! پس به این دلیل و برای مشخص کردن قد همهٔ افراد روی زمین بایستی برای هر قد موجود روی زمین یک تعریفی داشته باشیم! که این غیر ممکن می باشد.

در مثالی دیگر به دمای آب می پردازیم؛ اگر ۰ درجه، سرد و ۱۰۰ درجه، گرم تعریف شده باشد پس هر آب سردی دمایش ۰ درجهٔ مطلق و هر آب گرمی دمایش ۱۰۰ درجه است! حال اگر آبی داشته باشیم که دمایش ۵۵ درجه باشد، دمایش چقدر است؟ جواب این است که هیچ! یعنی هیچ آبی در دنیا یافت نمی شود که دمایش ۵۵ درجه یا هر مقدار دیگری بجز ۰ یا ۱۰۰ درجه باشد!! و یا پاسخی دیگر برای این پرسش به این نحو است که هر آبی با دمای کمتر از ۰ درجه یا کمی بیشتر از آن، سرد و هر آبی با دمای بیشتر از ۱۰۰ درجه یا کمی کمتر از آن، گرم است ولی این «کمی» چقدر است و هر کسی چه تعریفی از آن دارد، مشخص نیست. حال، آبی با دمای ۵۰ درجه، یا به هیچ گروهی تعلق ندارد و یا هر کس به دلخواه خود آن را تعریف و بیان می کند.

همانطور که ملاحظه شد در منطق Crisp برای بیان صحیح هر چیز بایستی کاملاً عددی پیش رویم و با قراردادهای بیشماری سر و کار داریم که این خود، کار را بسیار سخت و بهتر بگوییم غیر عملی می سازد.

---

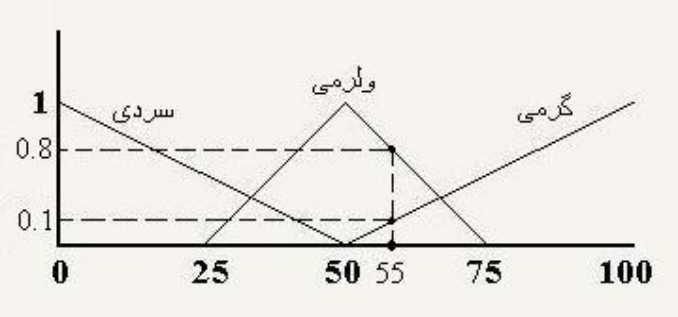
[1] Fuzzy Controllers

در منطق Fuzzy برخلاف منطق قبلی، تماماً با عناوین و اسامی سر و کار داریم؛ به طور مثال می گوییم دمای آب سرد است یا ولرم است یا گرم است یا خیلی سرد است یا خیلی گرم است و یا درصدی گرم و درصدی نیز ولرم است ویا .... به این ترتیب برای هر دمایی از آب می توانیم دمای آن را نه بطور صریح و واضح ولی با بیانی ساده و قابل فهم مشخص کنیم. در اینجا می بینیم که با مقادیر عددی مطلقی سر و کار نداریم و این دستمان را باز می گذارد و نحوهٔ بیان و فهم قضیه را آسان می نماید. به بیانی ساده تر در منطق Fuzzy با گرمی، سردی، ولرمی، کمی، زیادی، کوتاهی، بلندی و از این قبیل عناوین سر و کار داریم.

با یک مثال به توضیح کنترل فازی می پردازیم؛

مثال ۱. می خواهیم دمای یک تانکر آب را ثابت نگه داریم؛

یک «فضای ورودی» به شکل زیر برای کنترل دمای آب تانکر تعریف می کنیم:



محور افقی در فضای ورودی فوق، مشخصهٔ درجهٔ آب می باشد.

اولین موضوع مهم در کنترل فازی، فضای ورودی می باشد؛ در این مثال برای فضای ورودی، سه تابع تعریف کرده ایم: سردی، گرمی، ولرمی.

تابع سردی: $y = 1 - \frac{x-0}{50-0}$

تابع ولرمی: $y = 1 - \frac{x-50}{75-50}$; $y = \frac{x-25}{50-25}$

تابع گرمی: $y = \frac{x-50}{100-50}$

فرض کنید دماسنج عدد ۵۵ را نشان دهد؛ در مرحلهٔ آغازین بایستی «درجهٔ عضویت» این مقدار عددی را نسبت به هر یک از توابع تعریف شده در فضای ورودی تعیین کنیم؛

با مشاهدهٔ فضای ورودی و از روی «توابع عضویت» میزان تعلق عدد ۵۵ به هر یک از توابع را تعیین می کنیم:

درجهٔ عضویت به سردی: ۰

درجهٔ عضویت به ولرمی: ۰,۸

درجهٔ عضویت به گرمی: ۰,۱

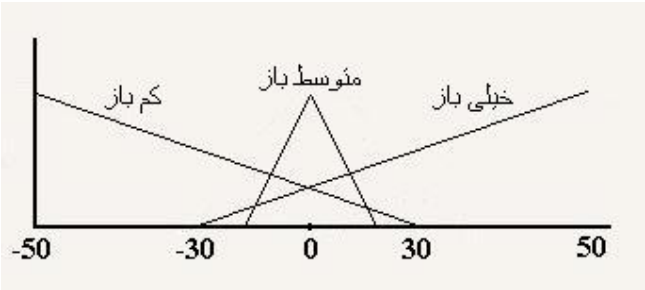
در هر سیستم کنترلی فازی، بازای هر فضای ورودی، تعدادی «قوانین حاکم بر سیستم»، که دومین موضوع مهم در کنترل فازی می باشد، وجود دارند. قوانین حاکم بر فضای ورودی این مثال به شرح زیر تعریف شده اند:

R۱: اگر سرد است، شیر A خیلی باز شود.

R۲: اگر ولرم است، شیر A متوسط باز شود.

R۳: اگر گرم است، شیر A کم باز شود.

در این مثال شیر A یک عامل(محرک) نامیده می شود.
سومین و آخرین موضوع اصلی در کنترل فازی، «فضای خروجی» است، که در این مثال فضای خروجی شیر A مطابق شکل زیر تعریف شده است:



محور افقی در فضای خروجی فوق، مشخصهٔ درجهٔ پیچ شیر A می باشد و لزومی به قرینه بودن آن نیست.
در مرحلهٔ دوم، بایستی با توجه به درجهٔ عضویت عدد ۵۵ در سه تابع ورودی و با توجه به قوانین حاکم و فضای خروجی، مقدار خروجی کنترلر را محاسبه کنیم. به این ترتیب که:
از قانون اول، "ماکزیمم تابع «خیلی باز»" را در "درجهٔ عضویت عدد ۵۵ در تابع «سردی» " ضرب کرده، و از قانون دوم، "ماکزیمم تابع «متوسط باز»" و "درجهٔ عضویت عدد ۵۵ در تابع «ولرمی»" را در هم ضرب می کنیم، و با توجه به قانون سوم، "ماکزیمم تابع «کم باز»" را در "درجهٔ عضویت عدد ۵۵ در تابع «گرمی»" ضرب می کنیم. حال "مجموع سه مقدار حاصله" را بخش بر "مجموع سه درجهٔ عضویت ذکر شده" کرده و جواب را به عنوان خروجی کنترلر(درجه ای که شیر A باید روی آن قرار گیرد) در نظر می گیریم. به این ترتیب داریم:

$$\frac{0(50)+0.8(0)+0.1(-50)}{(0+0.8+0.1)}=\frac{(-5)}{0.9}=-5.5$$

مطالب اخیر را به عبارتی ساده تر بیان می کنیم:
از آنجا که طبق قانون اول داریم: (اگر سرد است، شیر A خیلی باز شود) و از طرفی دمای ۵۵ درجه در فضای ورودی، به تابع «سردی» اصلاً تعلق ندارد(سرد نیست) پس قانون اول در تنظیم شیر A اصلاً تأثیری ندارد.
بر اساس قانون دوم (اگر ولرم است، شیر A متوسط باز شود) و از آنجا که عدد ۵۵ باندازهٔ (۰٫۸) متعلق به تابع «ولرمی» است (٪۸۰ ولرم است)، پس بایستی شیر A، "% ۸۰ متوسط باز شود" که متوسط ترین درجهٔ باز شدن شیر A، درجهٔ (۰) است. پس این قانون به مقدار "% ۸۰ از متوسط ترین درجهٔ باز شدن شیر(۰)"، در تنظیم شیر A تأثیر می گذارد.
با توجه به قانون سوم نیز (اگر گرم است، شیر A کم باز شود) و از آنجا که عدد ۵۵ باندازهٔ (۰٫۱) متعلق به تابع «گرمی» است (% ۱۰ گرم است)، پس بایستی شیر A، "% ۱۰ کم باز شود" که کمترین درجهٔ باز شدن شیر A، درجهٔ (۵۰-) است. پس این قانون به مقدار "% ۱۰ از کمترین درجهٔ باز شدن شیر(۵۰-)"، در تنظیم شیر A تأثیر می گذارد.
در حالت بعد فرض کنید دماسنج عدد ۱۰۰ را نشان دهد، که با توجه به توضیحات فوق و به سبکی مشابه، مقدار خروجی اعمالی به شیر A برابر خواهد شد با:

$$\frac{0(50)+0(0)+1(-50)}{(0+0+1)}=-50$$

زیرا:

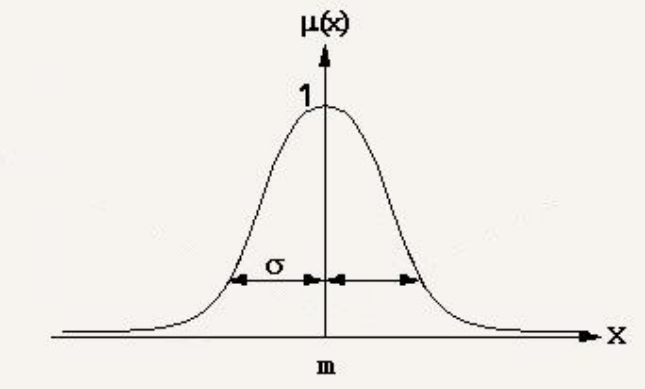
درجهٔ عضویت به سردی: ۰
درجهٔ عضویت به ولرمی: ۰
درجهٔ عضویت به گرمی: ۱
پس درجهٔ شیر A باید روی مقدار (۵۰-) قرار گیرد (کم باز شود).

با کمی دقت در مثال ذکر شده، می توان به خواص کلی کنترل فازی پی برد، در زیر نکات مهم را مرور می کنیم:

- سه موضوع کنترل فازی: فضای ورودی، قوانین حاکم بر سیستم، فضای خروجی.
- هر سیستم کنترل فازی، شامل یک یا چند «فضای ورودی» می باشد.
- هر فضای ورودی، از یک یا چند «تابع عضویت» تشکیل شده است که تعداد آنها معمولاً بین ۳ تا ۵ تابع می باشد و از آنجایی که انسان دارای ۵ حس گوناگون است و در آن واحد بیش از ۵ کار را نمی تواند انجام دهد، نهایتاً ۵ تابع و البته به ندرت ٦ یا ۷ تابع عضویت هم تعریف می شود.
- توابع عضویت هر فضای ورودی باید از یک جنس باشند.
- مجموع توابع عضویت تعریف شده در فضای ورودی، در هر نقطه(عدد crisp ورودی) مقداری بین ۰ و ۱ و یا برابر با آنهاست.
- بازای هر مقدار crisp ورودی، هر یک از توابع عضویت، یک «درجهٔ عضویت» برای آن مقدار، در خود تعریف می کنند؛ که معرف میزان تعلق آن ورودی به مجموعه های فازی می باشد.
- اگر یک فضای ورودی داشته باشیم، به تعداد توابع عضویت ورودی، «قانون حاکم» موجود است. و اگر بیش از یک فضای ورودی داشته باشیم، تعداد قوانین حاکم بر سیستم، برابر است با حاصلضرب تعداد توابع عضویت فضاهای ورودی در یکدیگر.
- در هر قانون، یک یا چند عامل(علت) وجود دارد.
- به تعداد عاملها(محرکها) در قوانین، «فضای خروجی» داریم.
- در فضای خروجی هر یک از عاملها، به تعداد حالتهایی که همان محرک (در قوانین حاکم) می پذیرد، «تابع عضویت» وجود دارد.
- برای محاسبهٔ خروجی کنترلر به سه روش می توان عمل کرد؛ ماکزیمم مقدار، متوسط مقدار، و متوسط ماکزیمم. که به عنوان نمونه، در مثال فوق که از روش ماکزیمم مقدار استفاده شده بود، ماکزیمم هر یک از توابع را در درجهٔ عضویت عدد crisp ورودی ضرب کردیم.

شایان ذکر است که توابع عضویت مجموعه های فازی به فرمهای مختلفی همچون مثلثی، ذوزنقه ای، گوسی، زنگوله ای و ... در نظر گرفته می شوند. در زیر فرم کلی یک تابع گوسی را ملاحظه می کنید:

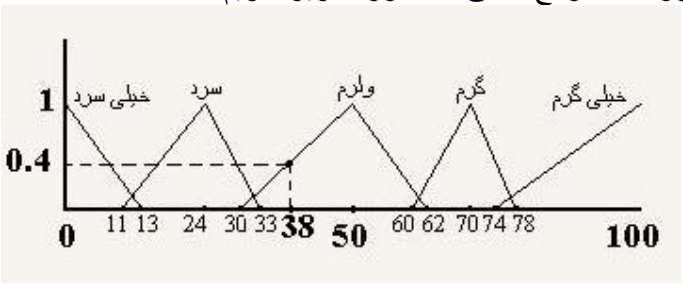
$$\mu(x)=e^{-\left[(x-m)/\sigma\right]^2}$$



با ذکر مثالی دیگر با این خصوصیات بیشتر آشنا می شویم:

مثال ۲. می خواهیم کنترل دمای تانکر آب را به روش فازی و با دو شیر انجام دهیم.

یک فضای ورودی با توابع مثلثی به صورت زیر داریم:



محور افقی در فضای ورودی فوق، نشانگر دمای آب تانکر است و همانطور که در شکل پیداست، لزومی به قرینه بودن توابع عضویت آن نیست.

فضای ورودی از ٥ تابع عضویت تشکیل شده است:

تابع خیلی سرد: $y = 1 - \frac{x-0}{13-0}$

تابع سرد: $y = 1 - \frac{x-24}{33-24}$; $y = \frac{x-11}{24-11}$

تابع ولرم: $y = 1 - \frac{x-50}{62-50}$; $y = \frac{x-30}{50-30}$

تابع گرم: $y = 1 - \frac{x-70}{78-70}$; $y = \frac{x-60}{70-60}$

تابع خیلی گرم: $y = \frac{x-74}{100-74}$

قوانین حاکم بر سیستم، به قرار زیر می باشند:

R۱: اگر خیلی سرد است، شیر A باز شود و شیر B بسته شود.

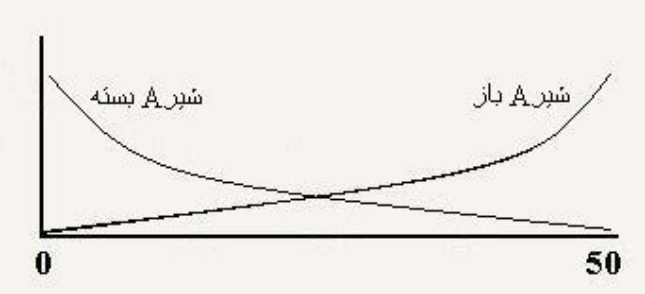
R۲: اگر سرد است، شیر A باز شود و شیر B باز شود.

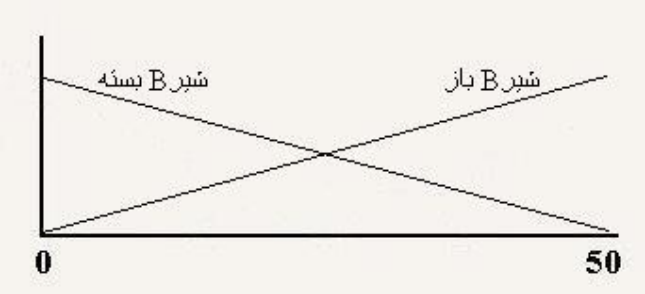
R۳: اگر ولرم است، شیر A بسته شود و شیر B بسته شود.

R٤: اگر گرم است، شیر A باز شود و شیر B باز شود.

R٥: اگر خیلی گرم است، شیر A بسته شود و شیر B باز شود.

همانطور که می بینید، در قوانین دو عامل(شیر A و B) وجود دارند، ازاین رو سیستم ما دو فضای خروجی دارد؛ فضای خروجی شیر A و فضای خروجی شیر B :





محورهای افقی دو فضای خروجی فوق به ترتیب، درجهٔ پیچ شیر A و درجهٔ پیچ شیر B هستند که در اینجا درجهٔ ۰ یعنی شیر کاملاً(% ۱۰۰) بسته و درجهٔ ۵۰ یعنی شیر کاملاً باز.

فرض کنید دماسنج عدد ۳۸ را نشان می دهد؛ درجهٔ عضویت عدد ۳۸ به هر یک از توابع عضویت ورودی را بدست می آوریم:

درجهٔ عضویت در تابع خیلی سرد: ۰
درجهٔ عضویت در تابع سرد: ۰
درجهٔ عضویت در تابع ولرم: ۰,٤
درجهٔ عضویت در تابع گرم: ۰
درجهٔ عضویت در تابع خیلی گرم: ۰

خروجی کنترلر A را به روش متوسط مقدار و خروجی کنترلر B را به روش ماکزیمم مقدار محاسبه می کنیم:

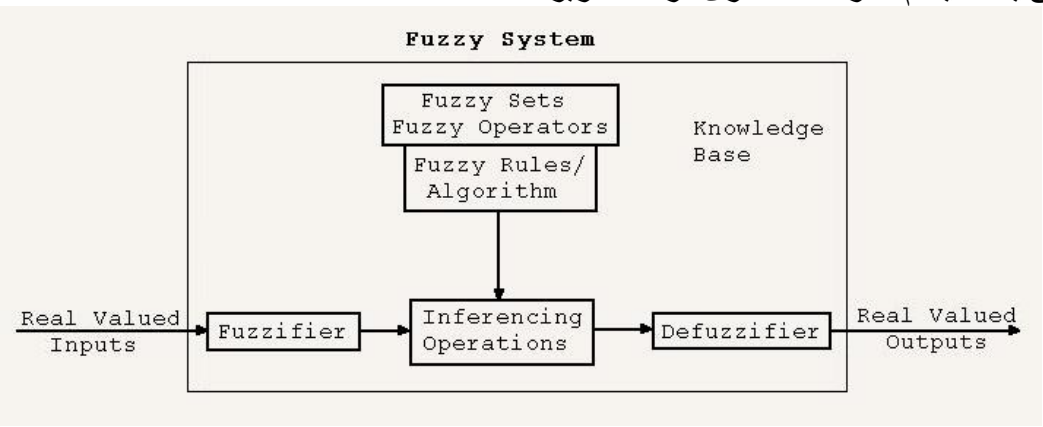
خروجی کنترلر A: $\frac{0.4(5)}{0.4} = 5$

برای محاسبهٔ مقدار متوسط بایستی متوسط انتگرال تابع خروجی مربوطه(مطرح شده در قانون) را بدست آوریم؛ در اینجا فرض کرده ایم که مقدار متوسط تابع "شیر A بسته" در فضای خروجی شیر A، (٥) باشد.

خروجی کنترلر B: $\frac{0.4(0)}{0.4} = 0$

همانطور که در نمودار تابع خروجی "شیر B بسته" در فضای خروجی شیر B پیداست، مقدار ماکزیمم تابع(ماکزیمم مقدار بسته بودن شیر B)، درجهٔ (۰) است.

با توجه به دو مثال فوق نتیجه می گیریم که یک کنترل کنندهٔ فازی با توابع عضویت مجموعه های فازی، تعدادی قوانین فازی و یک روش استنتاج فازی تعریف می شود. دیاگرام بلوکی یک سیستم کنترل کنندهٔ فازی در شکل زیر آمده است:



بنابراین یک کنترل کنندهٔ فازی بر اساس سه مفهوم اصلی مشخص می شود:

۱. فازی کردن مقادیر اندازه گیری شده (Fuzzification)
۲. مکانیزم استنتاج (Decision Making)
۳. فازی زدایی یک مجموعهٔ فازی (Defuzzification)

ورودیها به قسمت فازی گر داده می شوند و توابع عضویت تعریف شده در این قسمت آنها را به ورودیهای مناسب برای ماشین استنتاج فازی تبدیل می کنند. ماشین استنتاج فازی با استفاده از ورودیهای دریافت شده از قسمت فازی گر و با توجه به قواعد فازی که به آن داده شده است، خروجی مناسب را تولید می کند. این خروجی به قسمت فازی زدا رفته و در آنجا خروجی نهایی کنترلر تولید می شود.

نوع طراحی قوانین و توابع عضویت فازی به نوع سیستمی که قرار است کنترل شود، وابسته می باشد. در حالی که جهت استنتاج فازی می توان مستقل از سیستم مورد نظر،از میان روشهای متعدد استنتاج (از قبیل روشهای min, max, max-product, min-max, ...) یکی را جهت استنتاج قوانین انتخاب نمود.

بطور کلی چهار روش برای تعیین قوانین کنترل کننده و توابع عضویت وجود دارد:

۱. استفاده از تجربهٔ فرد متخصص در کنترل پروسهٔ مورد نظر.
۲. از طریق آزمایش و سعی و خطا روی پروسه یا مدل ریاضی آن.
۳. استفاده از مدل فازی پروسه جهت استخراج قوانین کنترلی مناسب برای پروسه.
٤. استفاده از روشهای یادگیری.

در روش آخر که جدید تر از سه روش قبلی است، سعی می شود که با بهره گیری از مکانیزمهای یادگیری، قواعد کنترل سیستم و مجموعه توابع عضویت، بدست آید. این کنترل کننده ها که "کنترل کننده های فازی خود سازمان ده"[1] نامیده می شوند، می توانند بصورت وفقی نیز کار کنند.

کنترل کننده های فازی-عصبی را می توان نمونه ای ازاین نوع کنترل کننده ها دانست که در آنها از شبکهٔ عصبی جهت اصلاح، یادگیری و بهینه سازی قوانین فازی استفاده می شود.

[1] Self-organizing Fuzzy Controllers

۳-٤) کنترل کننده های عصبی[۱]

کنترل کننده های عصبی، کنترل کننده های خودکاری هستند که در آنها از شبکهٔ عصبی جهت محاسبه و پردازش سیگنال کنترلی استغاده می شود. عملکرد شبکهٔ عصبی بر پایهٔ یادگیری بوسیلهٔ مثال می باشد. شبکه های عصبی الگوی ساده شدهٔ شبکه عصبی مغز هستند. هر شبکهٔ عصبی از تعداد زیادی واحدهای مجزای محاسباتی خطی و یا غیر خطی بنام «نرون» تشکیل می شود. هر کدام از این واحدهای محاسباتی، پارامترهایی دارند که اصطلاحاً به آنها وزن (Weight) گفته می شود. با تنظیم این وزنها، اصطلاحاً شبکه آموزش می بیند؛ یعنی می توان توابع غیر خطی مختلفی را بین ورودیها و خروجیهای شبکه ایجاد نمود. در شبکه های عصبی، هدف نهایی انتخاب وزنها است بطوریکه رابطهٔ مورد نظر بین ورودی و خروجی برقرار گردد.

شبکه های عصبی از اطلاعات عددی که توسط حسگرهای ورودی و خروجی به دست می آید، برای تنظیم وزنهای خود (یادگیری)، استفاده می کنند. به بیان دیگر یک شبکهٔ عصبی از تجارب عینی جهت یادگیری استفاده می کند تا چنانچه بعداً مواردی شبیه این مورد پیش بیاید، با یادآوری تجارب قبلی بتواند خروجی مناسب را تولید نماید. از دیدگاه تئوری سیستمها، شبکه های عصبی مصنوعی قابلیت نگاشت غیر خطی از یک فضای با بعد متناهی به فضای دیگر را دارا می باشند. چنین شبکه هایی در حالت ایده آل قادرند بر هر سه نوع مشکل کنترل سیستمهای پیچده (یعنی پیچیدگی محاسباتی، غیر خطی بودن و عدم قطعیت) غلبه کنند. شبکه های عصبی قادر به انجام هر گونه نگاشت غیر خطی می باشند. روشهایی که جهت تنظیم پارامترهای شبکهٔ عصبی بکار می روند همگی متکی به داده های ورودی و خروجی شبکه می باشند و به شرط اعمال داده های متناسب با همهٔ جنبه های ناشناختهٔ سیستم، عدم قطعیت سیستم از بین می رود. نهایتاً به دلیل اینکه نرونها می توانند بصورت موازی با هم کار نمایند، پس می توان با استفاده از توازی، پیچیدگی محاسباتی را کم نموده و بر سرعت پردازش اضافه نمود.

با تفاسیر فوق معلوم می شود که بخش مهمی از طراحی یک کنترل کنندهٔ عصبی، تعیین چگونگی جمع آوری داده های آموزشی و استفاده از آن، جهت تنظیم پارامترهای آزاد شبکهٔ عصبی می باشد. این مکانیزم به کنترل کننده های وفقی که در آن پارامترهای آزاد کنترل کننده در طول زمان تصحیح می شوند، شباهت زیادی دارد. بنابر این کنترلگرهای عصبی، ذاتاً وفقی می باشند؛ یعنی ساختار آنها طوری طراحی شده که قادر به پذیرش اطلاعات جدید در جهت بهبود عملکرد خود می باشند.

با توجه به قابلیت شبکه های عصبی در کنترل سیستمهای پیچیده، می توان از آن جهت کنترل ربات نیز استفاده نمود. تا کنون روشهای متعددی توسط محققین، جهت پیاده سازی کنترلگرهای عصبی پیشنهاد شده است. اکثر این روشها به دنبال این هستند که شبکهٔ عصبی، دینامیک ربات را فرا گیرد، حال به طور کامل یا تقریبی. همین هدف در کنترل کننده های فازی-عصبی به عنوان زیرمجموعه ای از کنترلگرهای عصبی دنبال می شود. بنابراین شناخت روشهای مختلف پیاده سازی و آموزش کنترلگرهای عصبی و بررسی نقاط ضعف و قوت آنها می تواند کمک بزرگی در جهت بهبود طراحی کنترلگرهای عصبی یا فازی-عصبی باشد.

---

[۱] Neuro Controllers

٤-٤) کنترل کننده های فازی-عصبی

در سالهای اخیر تلفیق دو روش فازی و عصبی به عنوان یکی از روشهای مطرح در کنترل سیستمهای مختلف مطرح گردیده است. این روش اولین بار توسط «لی»[۱] در سال ۱۹۷۰ مطرح گردید. در این روش، تجربهٔ شخص خبره و داده های واقعی بدست آمده از سیستم با هم تلفیق می شود و درحقیقت تجارب شخص خبره با استفاده از داده های واقعی تصحیح می شود.

استفاده از این روش جهت آموزش شبکه های عصبی، این حقیقت را روشن ساخت که شبکه های فازی-عصبی زیرمجموعه ای از شبکه های عصبی می باشند که نرونهای آنها قوانین فازی را پیاده سازی می نمایند. در حقیقت می توان به کمک این روش تجربیات شخص خبره را درون نرونهای شبکهٔ عصبی جاسازی نمود و به کمک مشاهدات عینی (داده های واقعی) به اصلاح این تجربیات پرداخت. در حقیقت شبکهٔ فازی-عصبی، خصلت خوانایی و شفافیت[۲] قوانین فازی و دقت[۳] شبکه های عصبی را با هم یکجا گرد آورده است.

امروزه کاربرد سیستمهای فازی-عصبی در کارهایی مانند کنترل سیستمهای غیرخطی، شناسایی الگو، پزشکی، سیستمهای خبره و ... مطرح گردیده است. ربات نیز به عنوان یک سیستم غیرخطی می تواند توسط کنترل کنندهٔ فازی-عصبی کنترل شود. به دلیل سریعتر بودن این کنترلگر نسبت به کنترل کنندهٔ عصبی و دقیقتر بودن آن نسبت به کنترل کننده های فازی، می توان از آن در کنترل بلادرنگ[٤] ربات استفاده نمود.

جهت پیاده سازی یک کنترلگر فازی توسط شبکهٔ عصبی، یک شبکهٔ چهار لایه با مشخصات زیر مورد نیاز است:

۱. لایهٔ ورودی (Input Layer)
۲. لایهٔ فازی کننده (Fuzzification Layer)
۳. لایهٔ تصمیم گیری (Decision Layer)
٤. لایهٔ خروجی (Output Layer)

در طراحی کنترل کنندهٔ فازی-عصبی می توان از دو ساختار استنتاج، یکی «ممدانی»[۵] و دیگری فرم «سوگینو»[٦] استفاده کرد. در ساختار استنتاج ممدانی خروجی هر قانون به صورت فازی بیان شده ولی در ساختار استنتاج به فرم سوگینو، خروجی هر قانون ترکیب خطی از ورودیهای آن قانون و به عبارت دیگر بصورت غیرفازی (Crisp) می باشد. بنابر این فرم کلی قوانین آن به صورت زیر است:

Rule : If $X_۱$ is $A_{r۱}$ and $X_۲$ is $A_{r۲}$ and… and X_n is $A_{r\,n}$ Then $O_r = k_{r۰} + k_{r۱} X_۱ + \ldots + k_{r\,n} X_n$

مزیت استفاده از قوانین نوع سوگینو این است که:

اولاً: خروجی هر قانون به صورت crisp بوده، بنابراین با اندک محاسبه ای می توان خروجی کل قوانین را به دست آورد. پس عملیات محاسباتی ساده تر و سریعتر از فرم ممدانی انجام می شود.

---

[۱] Lee
[۲] Transparency
[۳] Attention
[٤] Real-time Control
[۵] Mamdani
[٦] Sugeno

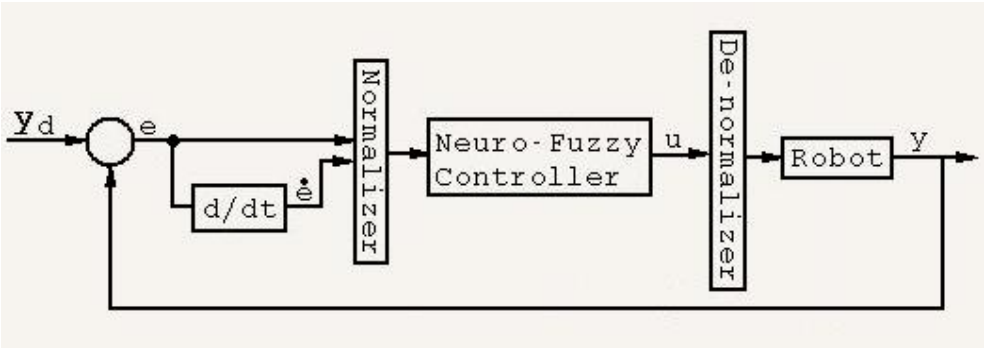
ثانیاً: در طراحی کنترلگر ربات از کنترلگری با دو ورودی و یک خروجی استفاده شده است که ورودیهای آن، خطا (e) و مشتق خطا ($\dot{e}$) می باشند. بنابراین با بهره گیری از فرم سوگینو، خروجی هر قانون به صورت زیر بدست می آید:

$$O_r = k_{r0} + k_{r1}e + k_{rn}\dot{e}$$

خروجی کلی قوانین از نوع سوگینو به فرم زیر تعریف می شود:

$$O = \frac{\sum_{r=1}^{R} \mu_r \left(k_{r0} + k_{r1}X_1 + ... + k_{rn}X_n\right)}{\sum_{r=1}^{R} \mu_r}$$

که در رابطهٔ فوق، μ_r درجهٔ فعالیت پیش شرط قانون r ام است و R نیز تعداد قوانین می باشد.
در شکل زیر دیاگرام کلی یک سیستم کنترلی فازی-عصبی را ملاحظه می نمایید:



هدف اصلی کنترلگرهای عصبی یا فازی-عصبی مستقل از مدل، شناسایی معکوس مدل دینامیکی سیستم و استفاده از آن جهت کنترل سیستم می باشد. الگوریتمهای یادگیری معکوس دینامیکی سیستم معمولاً به صورت نظارت شده[۱] می باشند. به این معنی که سیستم ناظر با محاسبهٔ خطای بین مقدار ایده آل خروجی شبکهٔ عصبی و مقدار واقعی آن، وزنهای شبکه را طوری تنظیم می نماید که خروجی شبکهٔ عصبی بر خروجی ایده آل منطبق[۲] شود. در اغلب سیستمهای کنترل شونده بوسیلهٔ شبکهٔ عصبی، به دلیل در دسترس نبودن خروجی ایده آل کنترلر عصبی، از خطای خروجی سیستم، به عنوان محک نظارت استفاده می شود. در اینگونه روشها که به روشهای "تطبیق خروجی"[۳] معروف هستند با استفاده از خطای خروجی سیستم، وزنهای کنترلگر عصبی را طوری تغییر می دهند که خروجی ایده آل و واقعی سیستم بر هم منطبق شوند.

[۱] Supervised
[۲] Match
[۳] Output Matching

٥-٤) کنترل فازی استفاده شده در ربات هوشمند

در این ربات هوشمند، به هر یک از ۷ سنسور موجود، یک ارزش تعلق گرفته است. این ارزشها به ترتیب از راست ترین سنسور تا چپ ترین سنسور عبارتند از (از راست به چپ):

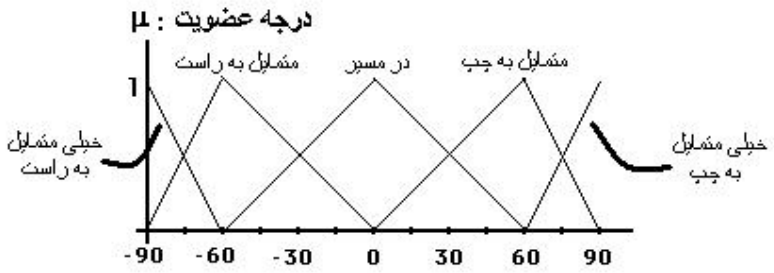
۹۰ ۶۰ ۳۰ ۰ -۳۰ -۶۰ -۹۰

همانطور که در بخش ۳-٤ توضیح داده شد، با قرار گرفتن هر یک از سنسورها روی خط، یک صفر منطقی تولید می شود. از این رو در شرایط طبیعی(۱ یا ۲ سنسور مجاور روی خط)، با توجه به ارزشهای اتلاقی برای هر یک از سنسورها، ۱۳ حالت مختلف خواهیم داشت.

ارزش نهایی سنسورها	راست ترین سنسور	دومین سنسور از راست	سومین سنسور از راست	سنسور وسطی	سومین سنسور از چپ	دومین سنسور از چپ	چپ ترین سنسور
۹۰	۰	۱	۱	۱	۱	۱	۱
۷۵	۰	۰	۱	۱	۱	۱	۱
۶۰	۱	۰	۱	۱	۱	۱	۱
٤۵	۱	۰	۰	۱	۱	۱	۱
۳۰	۱	۱	۰	۱	۱	۱	۱
۱۵	۱	۱	۰	۰	۱	۱	۱
۰	۱	۱	۱	۰	۱	۱	۱
-۱۵	۱	۱	۱	۰	۰	۱	۱
-۳۰	۱	۱	۱	۱	۰	۱	۱
-٤۵	۱	۱	۱	۱	۰	۰	۱
-۶۰	۱	۱	۱	۱	۱	۰	۱
-۷۵	۱	۱	۱	۱	۱	۰	۰
-۹۰	۱	۱	۱	۱	۱	۱	۰

ارزش نهایی در جدول فوق همان میانگین ارزش سنسورهای روی خط می باشد.

به این ترتیب با استفاده از متغیر مستقل "ارزش نهایی سنسورها" (مقدار ورودی فضای ورودی)، فضای ورودی را به همراه توابع عضویتش مطابق نمودار زیر برای ربات هوشمند تعریف می کنیم.



توابع عضویت به قرار زیرند:

تابع خیلی متمایل به راست(f۱): $\mu = 1 - \frac{x+90}{30}$

تابع متمایل به راست(f۲): $\mu = 1 - \frac{x+60}{60}$; $\mu = \frac{x+90}{30}$

تابع در مسیر(f۳): $\mu = 1 - \frac{x}{60}$; $\mu = \frac{x+60}{60}$

تابع متمایل به چپ(f٤): $\mu = 1 - \frac{x-60}{30}$; $\mu = \frac{x}{60}$

تابع خیلی متمایل به چپ(f٥): $\mu = \frac{x-60}{30}$

قوانین حاکم عبارتند از:

R۱: اگر ربات "خیلی متمایل به راست" است؛ موتور راست سریع بچرخد.
موتور چپ سریع و معکوس بچرخد.

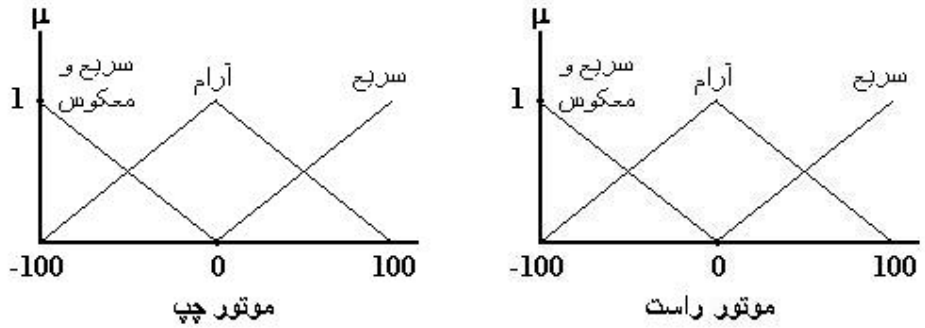
R۲: اگر ربات "متمایل به راست" است؛ موتور راست سریع بچرخد.
موتور چپ آرام بچرخد.

R۳: اگر ربات "در مسیر" است؛ موتور راست سریع بچرخد.
موتور چپ سریع بچرخد.

R٤: اگر ربات "متمایل به چپ" است؛ موتور راست آرام بچرخد.
موتور چپ سریع بچرخد.

R٥: اگر ربات "خیلی متمایل به چپ" است؛ موتور راست سریع و عکس بچرخد.
موتور چپ سریع بچرخد.

دو فضای خروجی "موتور راست" و "موتور چپ" عبارتند از:



محاسبهٔ خروجی ها(سرعت دو موتور) به روش "ماکزیمم مقدار":

مقدار crisp ورودی	سرعت موتور راست (RS)	سرعت موتور چپ (LS)	درجه عضویت (μ) به تابع (f)	
-۹۰	$\frac{1\times100}{1}=100$	$\frac{1(-100)}{1}=-100$	μ=۱	f۱
-۷۵	$\frac{0.5\times100+0.5\times100}{0.5+0.5}=100$	$\frac{0.5(-100)+0.5(0)}{0.5+0.5}=-50$	μ=۰,۵	f۱
			μ=۰,۵	f۲
-٦۰	$\frac{1\times100}{1}=100$	$\frac{1(0)}{1}=0$	μ=۱	f۲
-٤۵	$\frac{0.75\times100+0.25\times100}{0.75+0.25}=100$	$\frac{0.75(0)+0.25\times100}{0.75+0.25}=25$	μ=۰,۷۵	f۲
			μ=۰,۲۵	f۳
-۳۰	$\frac{0.5\times100+0.5\times100}{0.5+0.5}=100$	$\frac{0.5(0)+0.5\times100}{0.5+0.5}=50$	μ=۰,۵	f۲
			μ=۰,۵	f۳
-۱۵	$\frac{0.25\times100+0.75\times100}{0.25+0.75}=100$	$\frac{0.25(0)+0.75\times100}{0.25+0.75}=75$	μ=۰,۲۵	f۲
			μ=۰,۷۵	f۳
۰	$\frac{1\times100}{1}=100$	$\frac{1\times100}{1}=100$	μ=۱	f۳
۱۵	$\frac{0.75\times100+0.25(0)}{0.75+0.25}=75$	$\frac{0.75\times100+0.25\times100}{0.75+0.25}=100$	μ=۰,۷۵	f۳
			μ=۰,۲۵	f٤
۳۰	$\frac{0.5\times100+0.5(0)}{0.5+0.5}=50$	$\frac{0.5\times100+0.5\times100}{0.5+0.5}=100$	μ=۰,۵	f۳
			μ=۰,۵	f٤
٤۵	$\frac{0.25\times100+0.75(0)}{0.25+0.75}=25$	$\frac{0.25\times100+0.75\times100}{0.25+0.75}=100$	μ=۰,۲۵	f۳
			μ=۰,۷۵	f٤
٦۰	$\frac{1(0)}{1}=0$	$\frac{1\times100}{1}=100$	μ=۱	f٤
۷۵	$\frac{0.5(0)+0.5(-100)}{0.5+0.5}=-50$	$\frac{0.5\times100+0.5\times100}{0.5+0.5}=100$	μ=۰,۵	f٤
			μ=۰,۵	f۵
۹۰	$\frac{1(-100)}{1}=-100$	$\frac{1\times100}{1}=100$	μ=۱	f۵

. فصل پنجم .

هوشمندی و کامپیوتر

این سؤال که "هوش چیست؟" تا به حال به طور قطع قابل پاسخگویی نبوده است، چرا که محققان و دانشمندان نیز دربارهٔ نحوهٔ عملکرد مغز انسان نتوانسته اند پاسخی قطعی دهند. با این حال، می توان هوشمندی را با قابلیت های زیر مرتبط دانست:

- یادگیری از طریق تجربه.
- تصمیم گیری منطقی بر اساس تجربه و تعقل (تصمیم گیری انسانی).
- ابراز احساسات.

پیاده سازی این مشخصه ها در یک سیستم الکترونیکی به راحتی میسر نمی باشد. ولی از آنجایی که مدارها و برنامه هایی خاص، قادر به شبیه سازی بعضی از مفاهیم اولیه و اساسی هوشمندی هستند، لذا اینگونه از مدارها و برنامه ها را در زمرهٔ هوش مصنوعی به حساب می آوریم.

امروزه تعیین درجهٔ هوشمندی یک سیستم، خود مشکلی دشوار است و اندازه گیری هوش یک ماشین آسان نیست. حتی نمی توان تعیین کرد که آیا یک ماشین به اندازهٔ یک کرم، هوشمند هست یا خیر!

در هر حال، اگر مداری قادر به انجام کاری از طریق فرآیند تصمیم گیری خودش باشد، آن را در ردهٔ دستگاههای هوشمند قرار می دهیم.

دو روش برای اضافه نمودن هوش به یک ربات یا هر پروژهٔ مکاترونیکی دیگر وجود دارد؛ روش سخت افزاری و روش نرم افزاری.

در روش نخست با استفاده از مداراتی که از طریق سلولهای سخت افزاری، نورون های الکترونیکی و شبکه های عصبی، قادر به یادگیری هستند، می توان قابلیت هوشمندی ایجاد کرد. هرچند پیاده سازی یک نورون الکترونیکی با مدارات الکترونیک دشوار نیست و با ساختاری ساده هم می توان به تقلید عملکرد یک نورون زنده پرداخت، ولی تولید شبکه های عصبی که امروزه موضوع بسیاری از تحقیقات مهم می باشد، کاری بس دشوار است (در مغز موجودات پیچیده نظیر انسان، میلیاردها نورون با هم در ارتباطند و از چنین پیچیدگی شگفت آوری هوش حاصل می گردد). چنین مداراتی از هوش سخت افزاری استفاده می کنند. هوش سخت افزاری خاصیت ذاتی یک مدار می باشد.

دومین روش فراهم آوردن مشخصه های هوش مصنوعی در یک دستگاه، استفاده از برنامه در کامپیوتر یا میکروکنترلرها می باشد. در این حالت، مفهوم هوش در برنامه ای که از فرمانها و دستورها تشکیل شده، وجود دارد. این برنامه قادر است به عنوان واکنشی به سنسورهای ورودی، تصمیمات لازم را اتخاذ نماید و درنتیجه فرامین ضروری را به خروجیها اعمال کند. این سیستمها، سیستمهای دارای هوش نرم افزاری نامیده می شوند. هوش نرم افزاری بایک برنامه پیاده سازی می شود.

در این پروژه از هوش نرم افزاری با تکیه بر منطق فازی برای هوشمند ساختن ربات خط یاب استفاده شده است.

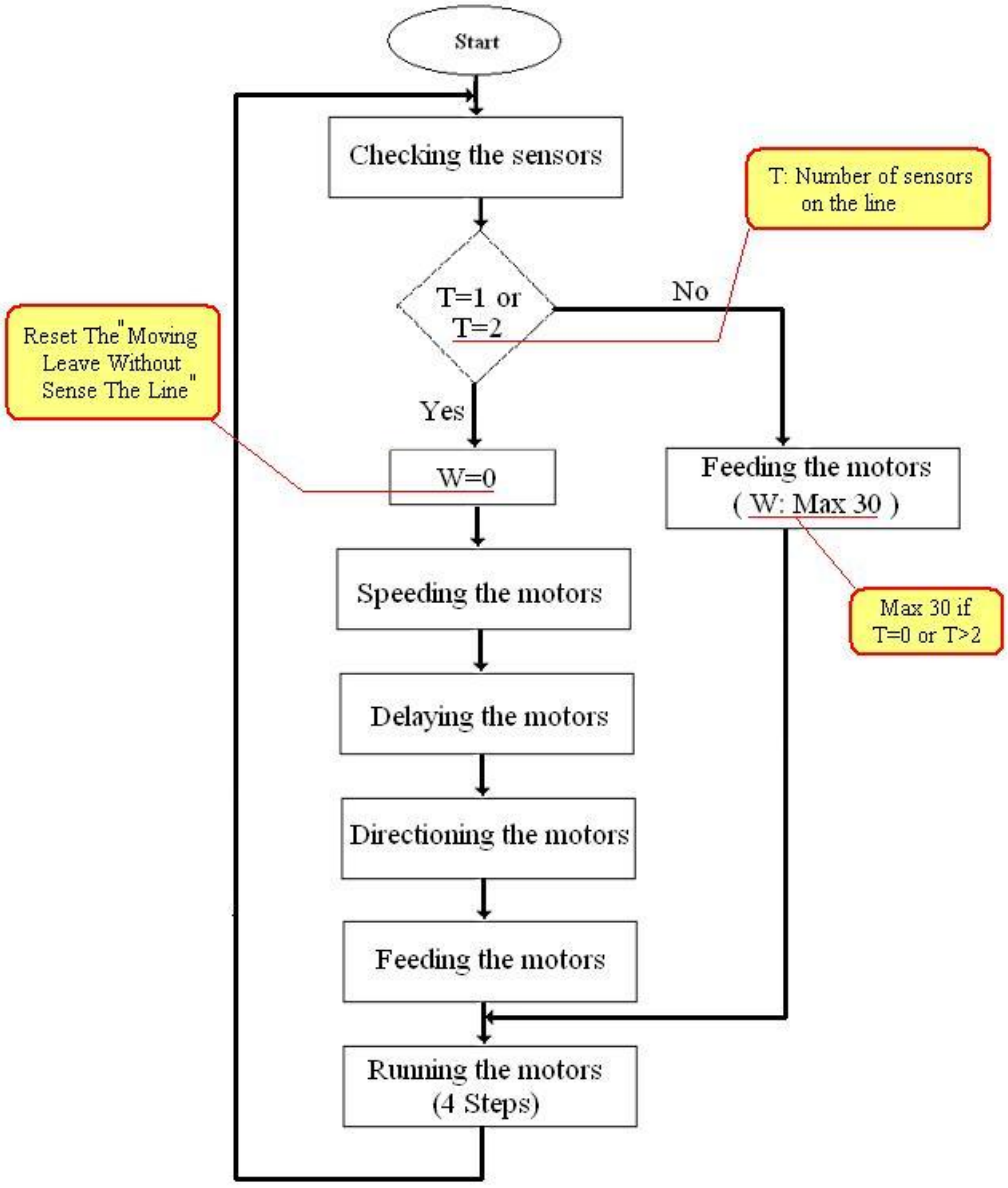
در دو بخش زیر توضیحات لازم برای برنامه ریزی این ربات هوشمند آورده شده است.

۱-۵) فلوچارت برنامه

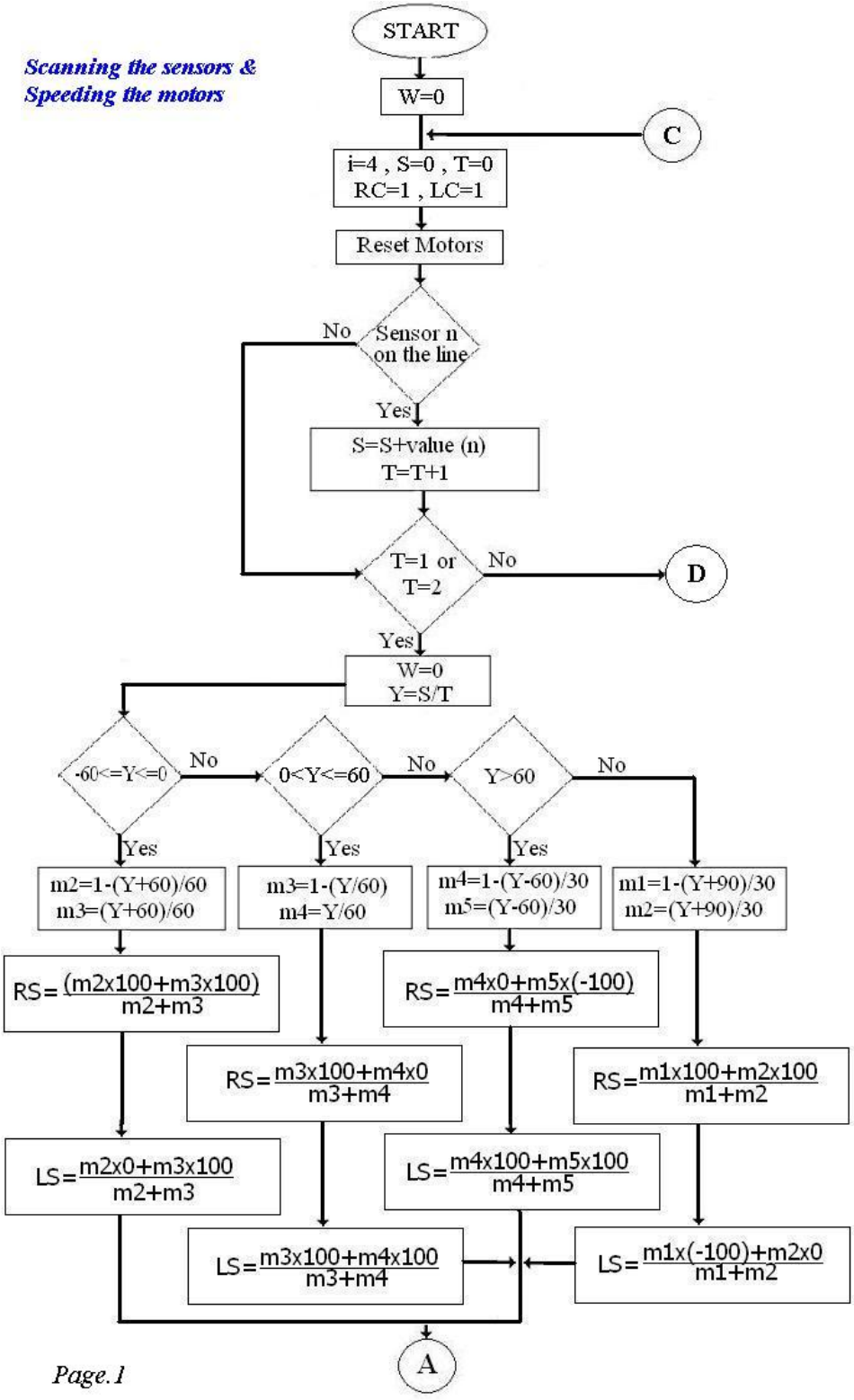
در شکل زیر فلوچارت و روند کلی برنامهٔ ربات هوشمند خط یاب آمده است:

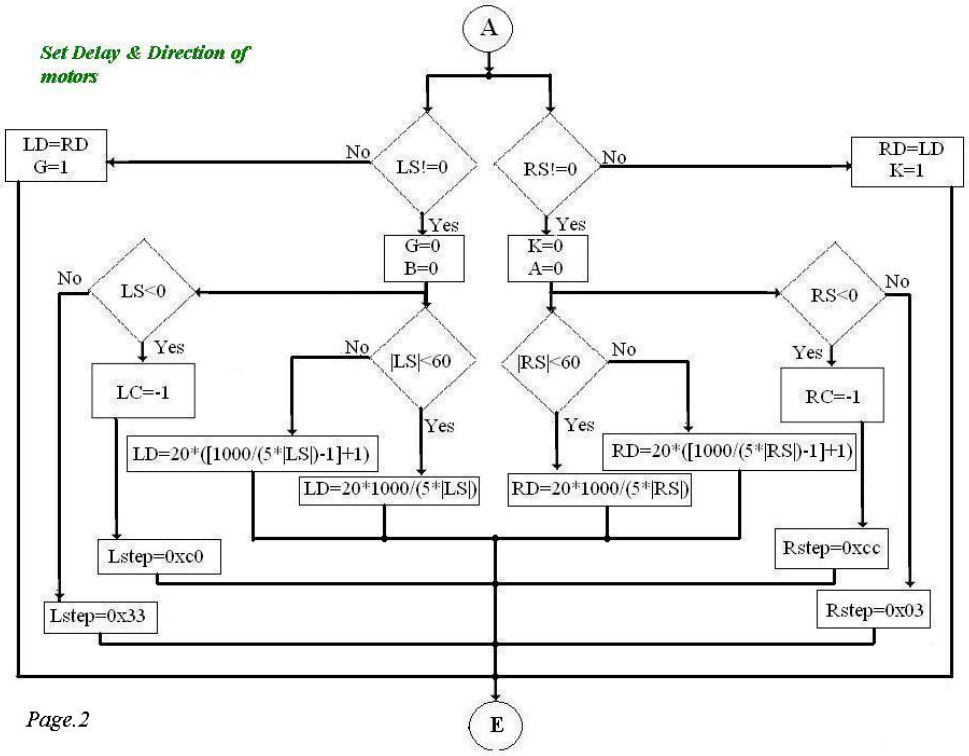
سیستم کنترلی این ربات به این صورت برنامه ریزی شده است که؛

- ابتدا ربات هر ۷ سنسور را چک کرده و بررسی می کند کدام یک از سنسورها خط را می خوانند(روی خط هستند). چیدمان سنسورها به گونه ای است که اگر خط، زیر ربات قرار داشته باشد، در حالت طبیعی ۱ یا ۲ سنسور می توانند خط را ببینند. مگر در پیچهای تند، مسیرهای متقاطع و زاویه های نوک تیز که ممکن است ۳ یا ٤ سنسور همزمان روی خط قرار گیرند.
- در صورتی که ربات خط را در حالت طبیعی مشاهده کند(۱ یا ۲ سنسور روی خط باشند)، به ترتیب مراحل "تعیین سرعت"، "محاسبهٔ زمان تأخیر لازم بین هر دو استپ متوالی" که با توجه به سرعت دو موتور در هر لحظه انجام می پذیرد، "تعیین جهت چرخش دو موتور(ساعتگرد یا پات ساعتگرد)" و "تغذیهٔ موتورها(اعمال قطار پالس مناسب)" برای به حرکت در آوردن دو موتور باندازهٔ ٤ استپ، اجرا می شوند.
- در صورتی که ربات توسط ۳ یا ٤ سنسور خط را مشاهده کند، ابتدا باندازهٔ ٤ استپ در راستا و سویی که اخیراً در حال حرکت بوده است (بدین منظور حافظه ای نرم افزاری در میکرو تعبیه شده است)، به حرکت ادامه می دهد و مجدداً از سنسورها برای ردیابی مسیر اطلاعات می گیرد.
- مراحل فوق به صورت یک چرخه دائماً تکرار می شوند تا زمانی که (پس از ۱۲۰ استپ) ربات هیچ خطی را سنس(حس) نکند.

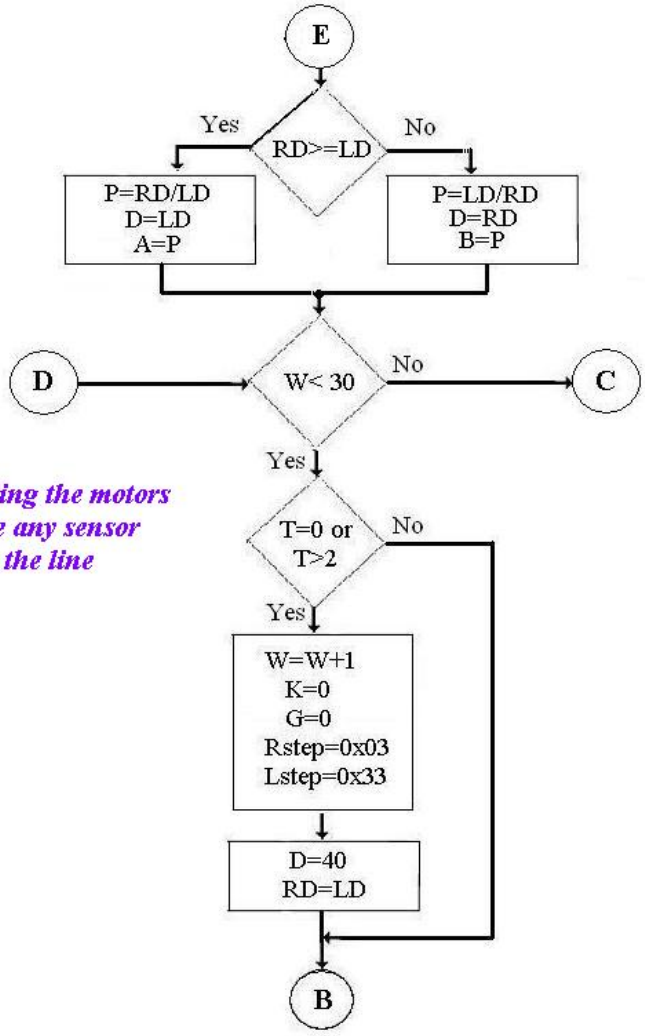

Start
Checking the sensors
T: Number of sensors on the line
T=1 or T=2
No
Yes
Reset The "Moving Leave Without Sense The Line"
W=0
Feeding the motors (W: Max 30)
Max 30 if T=0 or T>2
Speeding the motors
Delaying the motors
Directioning the motors
Feeding the motors
Running the motors (4 Steps)

Scanning the sensors &
Speeding the motors



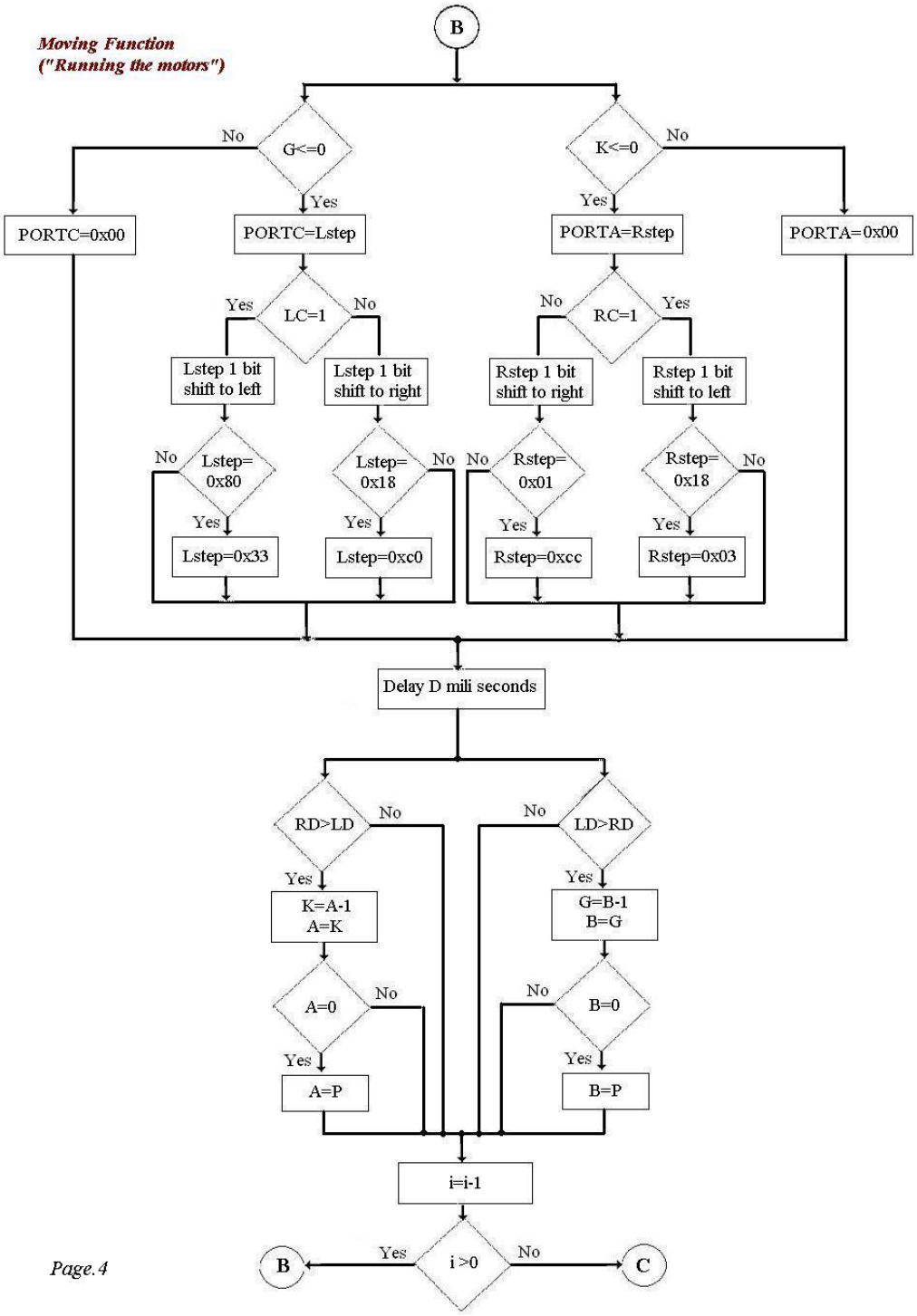

Set Delay & Direction of motors
A
LD=RD
G=1
No
LS!=0
RS!=0
No
RD=LD
K=1
Yes
Yes
G=0
B=0
K=0
A=0
No
LS<0
RS<0
No
Yes
Yes
No
|LS|<60
|RS|<60
No
Yes
Yes
LC=-1
RC=-1
LD=20*([1000/(5*|LS|)-1]+1)
RD=20*([1000/(5*|RS|)-1]+1)
LD=20*1000/(5*|LS|)
RD=20*1000/(5*|RS|)
Lstep=0xc0
Rstep=0xcc
Lstep=0x33
Rstep=0x03
E

Page.2



Feeding the motors while any sensor is on the line

Moving Function
("Running the motors")



٢-٥) برنامه ربات هوشمند به زبان C++

```
/*****************************************
This program was produced by the
CodeWizardAVR V١,٢٣,٨c Standard
Automatic Program Generator
© Copyright ١٩٩٨-٢٠٠٣ HP InfoTech s.r.l.
http://www.hpinfotech.ro
e-mail:office@hpinfotech.ro

Project : Line Follower; Fuzzy Logic
Version :
Date    : ٨/٢٦/٢٠٠٦
Author  : Alireza Erfani
Company :
Comments:

Chip type           : ATmega٣٢L
Program type        : Application
Clock frequency     : ١٦,٠٠٠٠٠٠ MHz
Memory model        : Small
External SRAM size  : ٠
Data Stack size     : ٥١٢
*****************************************/

#include <mega٣٢.h>
#include <delay.h>
#include <math.h>

unsigned char R١(signed char);
unsigned char R٣(signed char);
unsigned char R٥(signed char);

// Declare your global variables here
signed char i,
      W=٠,S=٠,T=٠,Y=١,
      RS,LS,
      RD,K,A,RC=١,
      LD,G,B,LC=١,
      P,D;
unsigned char Rstep,Lstep;
```

```
void main(void)
{
// Declare your local variables here

// Input/Output Ports initialization
// Port A initialization
// Func٠=Out Func١=Out Func٢=Out Func٣=Out Func٤=Out Func٥=Out
Func٦=Out Func٧=Out
// State٠=٠ State١=٠ State٢=٠ State٣=٠ State٤=٠ State٥=٠ State٦=٠ State٧=٠
PORTA=٠x٠٠;
DDRA=٠x٠F;

// Port B initialization
// Func٠=In Func١=In Func٢=In Func٣=In Func٤=In Func٥=In Func٦=In
Func٧=In
// State٠=T State١=T State٢=T State٣=T State٤=T State٥=T State٦=T State٧=T
PORTB=٠x٠٠;
DDRB=٠x٠٠;

// Port C initialization
// Func٠=Out Func١=Out Func٢=Out Func٣=Out Func٤=Out Func٥=Out
Func٦=Out Func٧=Out
// State٠=٠ State١=٠ State٢=٠ State٣=٠ State٤=٠ State٥=٠ State٦=٠ State٧=٠
PORTC=٠x٠٠;
DDRC=٠xF٠;

// Port D initialization
// Func٠=In Func١=In Func٢=In Func٣=In Func٤=In Func٥=In Func٦=In
Func٧=In
// State٠=T State١=T State٢=T State٣=T State٤=T State٥=T State٦=T State٧=T
PORTD=٠x٠٠;
DDRD=٠x٠٠;

// Timer/Counter ٠ initialization
// Clock source: System Clock
// Clock value: Timer ٠ Stopped
// Mode: Normal top=FFh
// OC٠ output: Disconnected
TCCR٠=٠x٠٠;
TCNT٠=٠x٠٠;
OCR٠=٠x٠٠;

// Timer/Counter ١ initialization
// Clock source: System Clock
// Clock value: Timer ١ Stopped
// Mode: Normal top=FFFFh
// OC١A output: Discon.
// OC١B output: Discon.
// Noise Canceler: Off
// Input Capture on Falling Edge
```

```
TCCR١A=٠x٠٠;
TCCR١B=٠x٠٠;
TCNT١H=٠x٠٠;
TCNT١L=٠x٠٠;
OCR١AH=٠x٠٠;
OCR١AL=٠x٠٠;
OCR١BH=٠x٠٠;
OCR١BL=٠x٠٠;

// Timer/Counter ٢ initialization
// Clock source: System Clock
// Clock value: Timer ٢ Stopped
// Mode: Normal top=FFh
// OC٢ output: Disconnected
ASSR=٠x٠٠;
TCCR٢=٠x٠٠;
TCNT٢=٠x٠٠;
OCR٢=٠x٠٠;

// External Interrupt(s) initialization
// INT٠: Off
// INT١: Off
// INT٢: Off
GICR|=٠x٠٠;
MCUCR=٠x٠٠;
MCUCSR=٠x٠٠;

// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=٠x٠٠;

// Analog Comparator initialization
// Analog Comparator: Off
// Analog Comparator Input Capture by Timer/Counter ١: Off
// Analog Comparator Output: Off
ACSR=٠x٨٠;
SFIOR=٠x٠٠;

// Watchdog Timer initialization
// Watchdog Timer Prescaler: OSC/٢٠٤٨
WDTCR=٠x١٠;

while (١)
    {
    // Place your code here
    PORTA=٠x٠٠;                                    // Reset The Right
Motor
    PORTC=٠x٠٠;                                   // Reset The Left Motor

// Scanning The Sensors
      if (PIND.٠==٠)                  // Bit(Sensor) Number n On The Line (٠)
```

```
        {
        S=٩٠;                                   // Sum Of On The Line Bit/Bits Value
        T++;                                    // Number Of On The Line Bits(IR Sensors)
        }
      if (PIND.١==٠)
        {
        S=S+٦٠;
        T++;
        }
      if (PIND.٢==٠)
        {
        S=S+٣٠;
        T++;
        }
      if (PIND.٣==٠)
        {
        S=S+٠;
        T++;
        }
      if (PIND.٤==٠)
        {
        S=S-٣٠;
        T++;
        }
      if (PIND.٥==٠)
        {
        S=S-٦٠;
        T++;
        }
      if (PIND.٦==٠)
        {
        S=S-٩٠;
        T++;
        }

// Speeding The Motors
    if (T==٠)                                   // Any Sensors Are On The Line
      {
      if (W==٠)                                 // Start Of Moving Without Sense The Line
        {
        K=٠;                                    // K<=٠ So Move The Right Motor
        G=٠;                                    // G<=٠ So Move The Left Motor
        Rstep=٠x٠٣;                             // Rstep:Right Motor
Feeding
        Lstep=٠x٣٣;                             // Lstep:Left Motor Feeding
        D=٤٠;                                   // D:Delay Between Per ٢ Steps
        RD=LD;                                  // RD:Right Motor Delay & LD:Left Motor Delay
        }                                       // End Of "if (W==٠)" Term
      else                                      // Enter In The Gap Or Continiue Of Above Term
        Y=Y;                                    // Continiue Moving With The Previous Same Terms
```

```
      W++;                          // Move Max W=٣٠ Times Without Sense The Line
      }
    else                                                          // If (T!=٠)
      {
      W=١;                          // Reset The Moving Leave Without Sense The
Line
      Y=S/T;                                  // Total Value Of On The Line Bit/Bits
      if (T>=٣)                                 // ٣ Or ٤ Sensors Are On The
Line
        {
        if (Y>٣٠ && Y<٦٠)
          Y=٤٥;
        else
          if (Y>-٦٠ && Y<-٣٠)
            Y=-٤٥;
          else
            if (Y>-٣٠ && Y<٣٠)
              Y=٠;
            else
              if (Y>٦٠)
                Y=٩٠;
              else
                if (Y<-٦٠)
                  Y=-٩٠;
        }
      }
    if (Y!=١)                          // The ROBOT Is Not On The Start Position
      {
      if (Y>=-٦٠ && Y<=٠)
        {
        RS=١٠٠;                                        // RS:Right Motor Speed
        LS=R٣(Y)*(١٠٠);                                 // LS:Left Motor Speed
        }
      else
        if (Y>٠ && Y<=٦٠)
          {
          RS=R٣(Y)*(١٠٠);
          LS=١٠٠;
          }
        else
          if (Y>٦٠)
            {
            RS=R٥(Y)*(-٥٠);
            LS=٥٠;
            }
          else
            {
            RS=٥٠;
            LS=R١(Y)*(-٥٠);
            }
```

```
// Delaying & Directioning The Motors
      if (RS!=٠)
        {
        if (abs(RS)<٦٠)
          RD=٢٠*١٠٠٠/(٥*abs(RS));                    // RD:Right
Motor Delay
        else
          RD=٢٠*ceil(١٠٠٠/(٥*abs(RS))-١);
        K=٠;                                  // K<=٠ So Move The Right Motor
        A=٠;                                        // A Set The Amount Of K
        if (RS<٠)
          RC=-١;                              // RC:Right Motor Cycle Direction
        }
      if (LS!=٠)
        {
        if (abs(LS)<٦٠)
          LD=٢٠*١٠٠٠/(٥*abs(LS));                     // LD:Left
Motor Delay
        else
          LD=٢٠*ceil(١٠٠٠/(٥*abs(LS))-١);
        G=٠;                                   // G<=٠ So Move The Left Motor
        B=٠;                                         // B Set The Amount Of G
        if (LS<٠)
          LC=-١;                               // LC:Left Motor Cycle Direction
        }
      else
        {
        LD=RD;
        G=١;                                     // G>٠ So Wait The Left Motor
        }
      if (RS==٠)
        {
        RD=LD;
        K=١;                                    // K>٠ So Wait The Right
Motor
        }
      if (RD>=LD)
        {
        P=RD/LD;                                          // P:Parts Of Delay
        D=LD;                                                 // D:Delay
        A=P;
        }
      else
        {
        P=LD/RD;
        D=RD;
        B=P;
        }
```

```
// Feeding The Motors
      if (RC==١)                                  // Direction Of Right Motor Is "+"
        Rstep=٠x٠٣;              // Set The Right Motor Feeding For Forward Moving
      else                                        // Direction Of Right Motor Is "-"
        Rstep=٠xcc;             // Set The Right Motor Feeding For Backward Moving
      if (LC==١)                                  // Direction Of Left Motor Is "+"
        Lstep=٠x٣٣;               // Set The Left Motor Feeding For Forward
Moving
      else                                        // Direction Of Left Motor Is "-"
        Lstep=٠xc٠;              // Set The Left Motor Feeding For Backward Moving
      }                                           // End Of The "if (Y!=١)" Term
    if (W<٣٠)                          // Move Mux ٣٠ Times Without Sense The Line
      {

// Moving Function
      for (i=٤;i>٠;i--)                     // Per ٤ Times Once Run The Program
        {
        if (K<=٠)                             // K<=٠ So Move The Right Motor
          {
          PORTA=Rstep;              // Rstep Given To The Right Motor Using PortA
          if (RC==١)                  // The Right Motor Want To Move Forward
            {
            Rstep=Rstep<<١;                 // Shift Rstep One Bit To The Left
            if (Rstep==٠x١٨)
              Rstep=٠x٠٣;// Reset The Feeding Of Right Motor For Forward Moving
            }
          else                          // The Right Motor Want To Move Backward
            {
            Rstep=Rstep>>١;                 // Shift Rstep One Bit To The Right
            if (Rstep==٠x٠١)
              Rstep=٠xcc;//Reset The FeedingOf Right Motor For Backward Moving
            }
          }
        else                                  // K>٠ So Wait The Right
Motor
          PORTA=٠x٠٠;                                 // Free The Right Motor
        if (G<=٠)                             // G<=٠ So Move The Left Motor
          {
          PORTC=Lstep;              // Lstep Given To The Left Motor Using PortC
          if (LC==١)                  // The Left Motor Want To Move Forward
            {
            Lstep=Lstep<<١;                 // Shift Lstep One Bit To The Left
            if (Lstep==٠x٨٠)
              Lstep=٠x٣٣;  // Reset The Feeding Of Left Motor For Forward
Moving
            }
          else                          // The Left Motor Want To Move Backward
            {
            Lstep=Lstep>>١;                 // Shift Lstep One Bit To The Right
            if (Lstep==٠x١٨)
```

```
                Lstep=٠xc٠; // Reset The Feeding Of Left Motor For Backward
Moving
              }
            }
          else                                          // G>٠ So Wait The Left Motor
            PORTC=٠x٠٠;                                 // Free The Left Motor
          delay_ms(D);                          // Delay Between Steps D mili seconds
          if (RD>LD)                        // Right Motor Delay = P*Left Motor Delay
            {
            K=A-١;       // Right Motor K Times Wait While The Left Motor Is Driving
            A=K;
            if (A==٠)
              A=P;                                      // Reset The Amount Of A
            }
          if (LD>RD)                        // Left Motor Delay = P*Right Motor Delay
            {
            G=B-١;       // Left Motor G Times Wait While The Right Motor Is
Driving
            B=G;
            if (B==٠)
              B=P;                                      // Reset The Amount Of B
            }
          }                                         // End Of The "for (i=٤;i>٠;i--)" Ring
       }                                              // End Of The "if (W<٣٠)" Term
     T=٠;
     S=٠;
     RC=١;
     LC=١;
     };                                               // End Of The "while (١)" Ring
}                                                     // End Of The "main" Function


// Function R١,R٣,R٥ Definition
unsigned char R١(signed char X)
{
unsigned char M١;
M١=١-(X+٩٠)/٣٠;
return M١;
}

unsigned char R٣(signed char X)
{
unsigned char M٣,Z;
M٣=(X+٦٠)/٦٠;
Z=١-X/٦٠;
if (Z>=٠ && Z<=١)
  M٣=Z;
return M٣;
}
```

```
unsigned char R۵(signed char X)
{
unsigned char M۵;
M۵=(X-۶۰)/۳۰;
return M۵;
}
```

۳-۵) برنامه ریزی میکروکنترلر

برای برنامه ریزی میکروکنترلر، دانش و آگاهی کافی از حداقل یک نرم افزار برنامه نویسی و کامپایلر آن جهت انتقال برنامه به میکرو لازم و ضروری است.

کامپایلرهای مختلفی برای میکروکنترلرهای AVR وجود دارند که از آن جمله می توان به کامپایلرهای IAR, GCC(GNU), ImageCraft, CodevisionAVR به زبان C و C++ و کامپایلرهای FastAVR, Bascom به زبان Basic و کامپایلر E-lab به زبان Pascal اشاره کرد. در این پروژه از نرم افزار CodevisionAVR جهت برنامه ریزی میکرو استفاده شده است. یکی از مزایای استفاده از این نرم افزار قدرتمند، تعلق آن در دسته کامپایلرهای C می باشد؛ استفاده از زبان سطح بالای C، کاهش زمان برنامه نویسی، نگهداری ساده تر و قابلیت حمل آسانتر و بیشتر، استفادهٔ مجدد از کدها و ساده تر شدن فهم برنامه را به همراه دارد. البته عیبی که دارد، افزایش حجم کدها و کاهش سرعت برنامه می باشد. از این رو میکروکنترلرهای AVR جهت کاهش و رفع این عیب، به گونه ای طراحی شده اند تا دستورات تولید شده توسط کامپایلرهای C را به صورت بهینه دیکد و اجرا نمایند.

از آنجا که برای آشنایی و یادگیری نرم افزار CodevisionAVR مراجع و کتب بسیاری موجود هستند لذا از توضیح آن صرفنظر می کنیم. برای آشنایی با این نرم افزار می توانید به مراجع ذکر شده مراجعه نمایید.

. فصل ششم .

مکانیک ربات

مکانیک ربات در اصل سیستم حرکتی و بدنهٔ ربات را شامل می شود. در طراحی ایده آل بیشترین کارایی با حداقل هزینه مورد نظر است. سیستم حرکتی یکی از مهمترین اجزای تشکیل دهندهٔ ربات است و هدف آن بررسی چگونگی انتقال ربات از یک نقطه به نقطهٔ دیگر است. این سیستم شامل جزئیاتی در مورد چگونگی استفاده از چرخ ها است که رایج ترین سیستم حرکتی رباتهای تعقیب خط است.

رباتها و ماشینهای مکاترونیک باید دارای امکاناتی برای سروکار داشتن با اشیاء یا انجام برخی کارها در دنیای خارج باشند. در ادامه این قسمت انواع محرکهایی[۱] که در پروژه های کاربردی یافت می شوند، ذکر شده اند.

رباتها قادرند با استفاده از پا، چرخ یا ریل از یک نقطه به نقطه ای دیگر جابجا شوند. پاهای ربات را می توان با استفاده از موتورها، سلونوئیدها، یا آلیاژهای حافظه دار (SMA)[۲] حرکت داد. SMA ماده ای است که به هنگام عبور جریان الکتریکی از آن تغییر شکل می دهد. از این آلیاژها می توان به عنوان "ماهیچه" در پروژه های مکاترونیک و رباتیک استفاده کرد.

رباتها و دستگاههای مکاترونیکی دارای دست نمی باشند. آنها برای گرفتن اشیاء از چنگک استفاده می کنند و این ابزارها نیز توسط مدارات الکترونیکی کنترل می شوند. حرکت این ابزارها هم می تواند با استفاده از سلونوئیدها، موتورها و یا SMA صورت پذیرد.

نحوهٔ انجام کارها توسط دست را می توان با استفاده از تجهیزاتی که منحصراً برای انجام یک وظیفهٔ خاص طراحی شده اند، تغییر داد. نظیر این حالت در بسیاری از رباتهای صنعتی دیده می شود. در بسیاری از کاربردها، قطعات بهم پیوستهٔ مکانیکی را می توان به گونه ای

---

[۱] Actuators

[۲] Shape Memory Alloys : آلیاژهای حافظه دار مواد شگفت انگیزی هستند که هنگامی که حرارت داده شوند، طی یک فرآیند بنام تبدیل Martensitic به شکل اولیهٔ خود باز می گردند. مادهٔ اصلی ساخت SMA ها، Flexinol می باشد. هنگامی که یک SMA سرد باشد (دمای آن کمتر از دمای تبدیل باشد) می توان آن را به سادگی به هر شکل جدیدی در آورد. زمانی که این ماده تا دمایی بیش از دمای تبدیل حرارت داده شود، دستخوش تغییرات ساختار کریستالی می شود که باعث برگشت آلیاژ به شکل اصلی و اولیهٔ خود می گردد. اگر SMA به هنگام تبدیل با مقاومت روبرو شود، قادر است نیروی بسیار قدرتمندی ایجاد کند. متداول ترین آلیاژ حافظه دار Nitinol نام دارد که آلیاژ ۵۰:۵۰ نیکل و تیتانیوم می باشد.

تنظیم نمود که با اندازه و شکل هر شئ مورد نظر سازگار شوند.
سیتم حرکتی در عملکرد ربات بسیار مؤثر است. در سالهای اخیر رباتهای پیشرفته تر بصورت انسان نما[1] و یا بصورت خزنده (شبیه حرکت مارها بر روی زمین) ساخته می شوند.

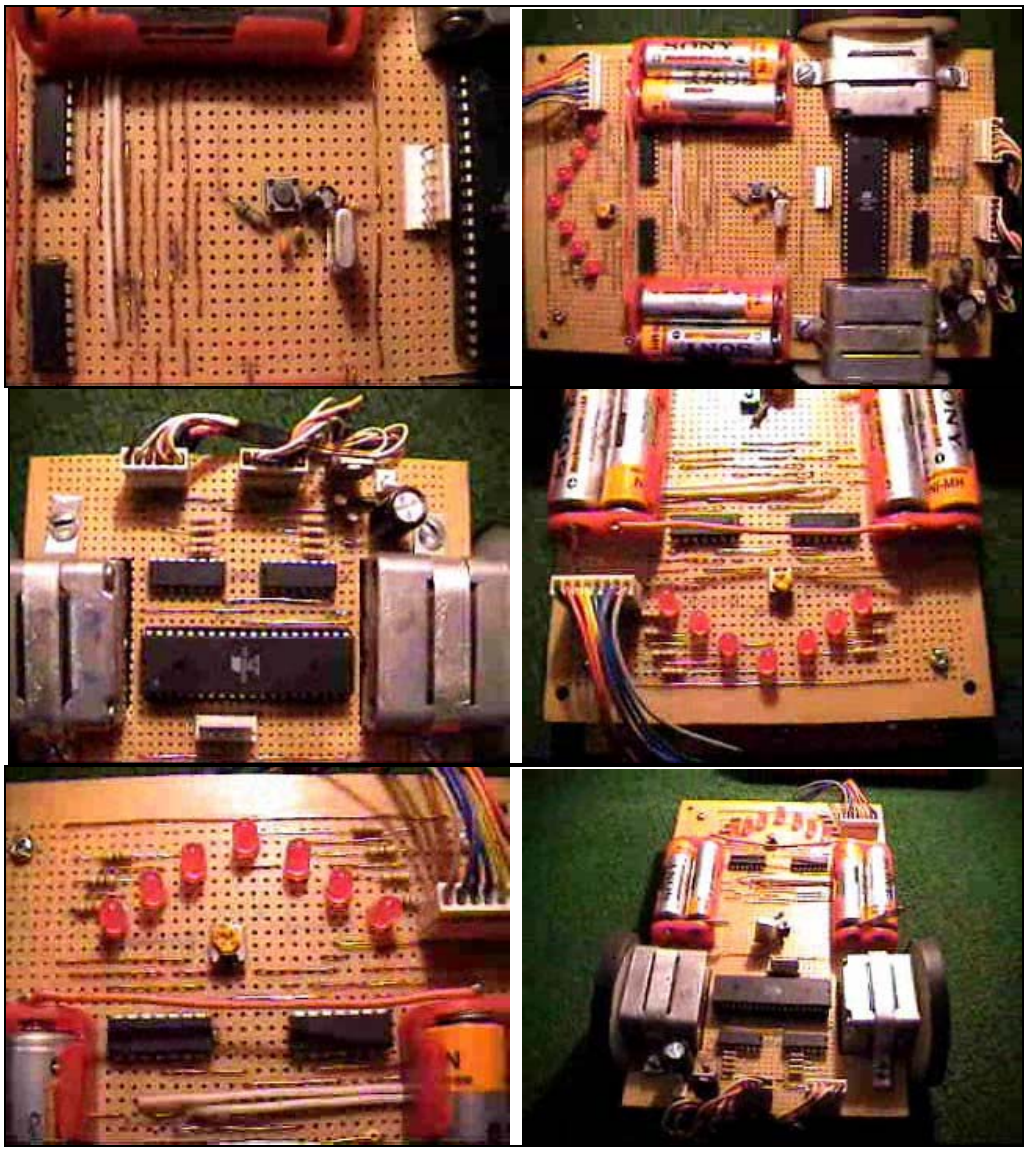
اولین ایده ای که طراحان برای سیستم حرکتی ربات انتخاب می کنند، سیستمهای چرخ دار است که معمولاً از وسایل نقلیهٔ روزمره الهام گرفته شده است. متداول ترین نوع رباتهای تعقیب خط نیز رباتهای چرخ دار هستند. این سیستم قدیمی ترین، ساده ترین و مؤثرترین روش برای حرکت بر روی سطوح صاف است. در این رباتها، چرخها به راحتی ساخته می شوند، قابلیت مانور خوبی دارند و براحتی به شفت موتورها متصل می شوند.
طراحی چرخها و تعداد آنها معمولاً به سلیقهٔ شخصی افراد و چگونگی استفاده از چرخها بستگی دارد. رباتهای دارای دو چرخ و چهار چرخ، متداولترین رباتها هستند، اما در اکثر طراحی های حرفه ای رباتها با شش چرخ یا بیشتر ساخته می شوند. با این وجود بسیاری از رباتهای تعقیب خط، سه چرخ دارند. چرا که هدف اصلی اکثر سازندگان پیروزی در مسابقات است و نه طراحی بهینه و پیشرفته.
ایدهٔ دیگری که در راهبری رباتها مورد استفاده قرار می گیرد، سیستم تانکی نام دارد. سیستمی که معمولاً در بولدوزرها و تانکهای جنگی دیده می شود. در این سیستم بخش اعظم گشتاور موتورها به زمین انتقال می یابد. اغلب این سیستم در طراحی رباتهایی که در سطوح ناهموار حرکت می کنند(مانند ربات امدادگر و ربات مین یاب) و رباتهایی که به قدرت بیشتری نیاز دارند(مانند رباتهای جنگجو وسومو) بکار می رود.

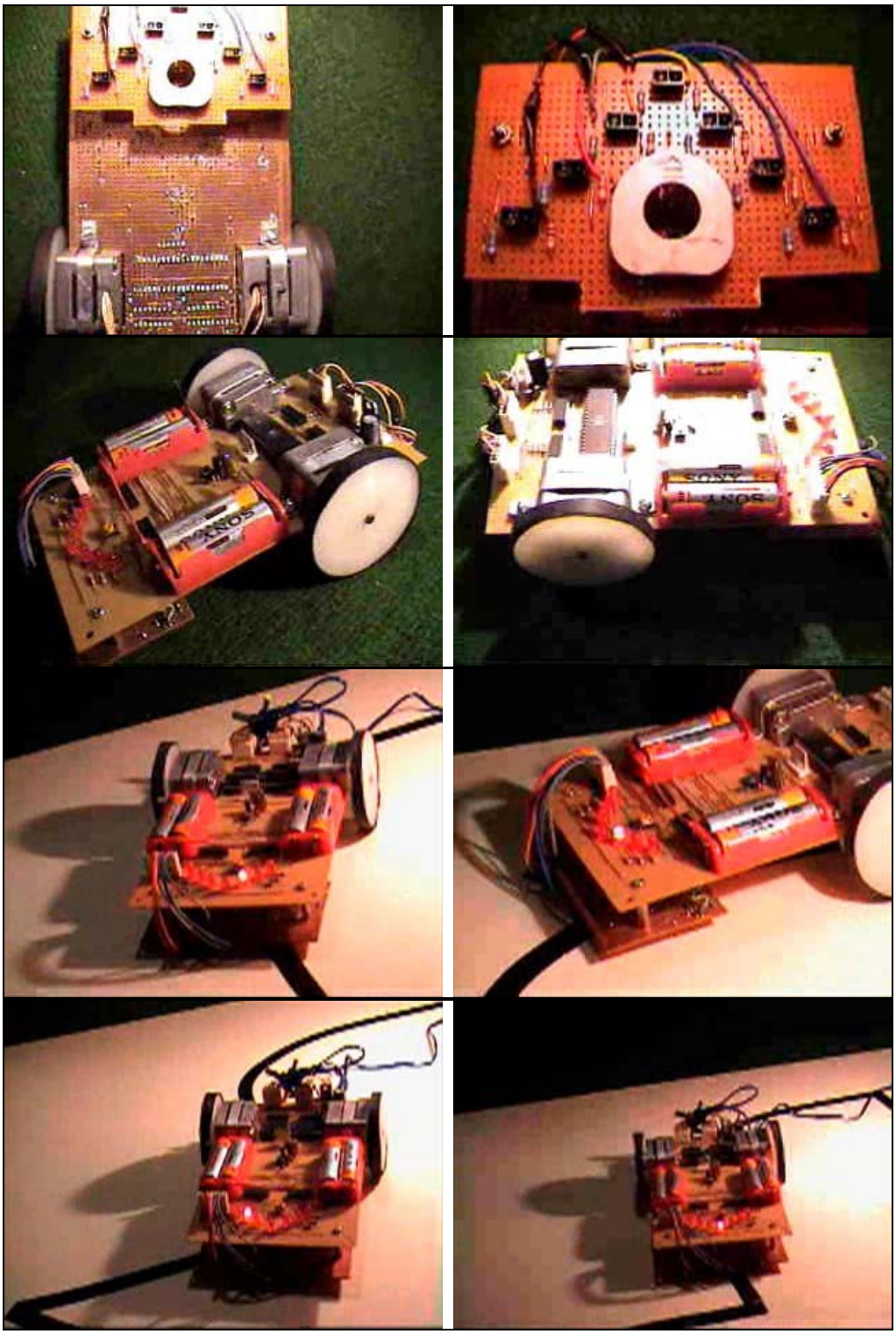
در اکثر رباتها از مواد گوناگونی جهت ساخت بدنه و شاسی استفاده می شود. این مواد عبارتند از چوب، پلاستیک (پلی کربنات و اکریلیک)، آلومینیوم، برنج و ....

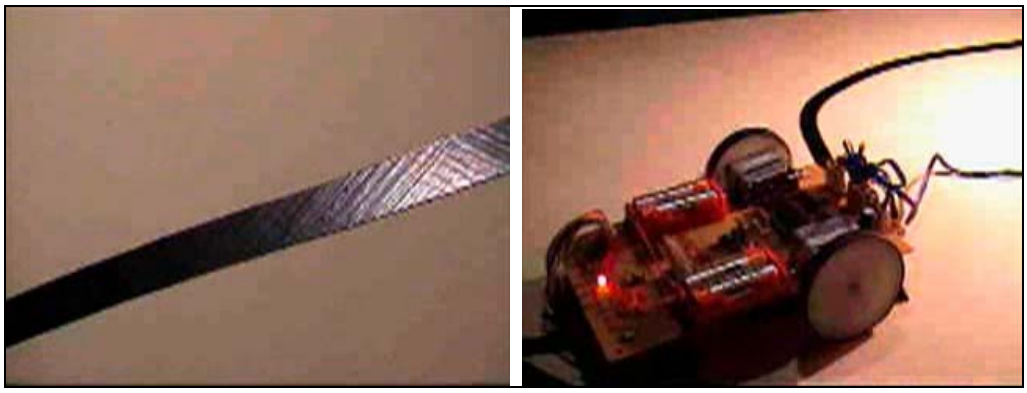
در این ربات خط یاب از هیچ ماده و عنصر اضافی برای بدنه استفاده نشده است، بلکه همانطور که در شکل زیر پیداست، تنها نگه دارندهٔ ربات خود برد مدار به همراه Spacer ها هستند که بر روی چرخ ها سوار هستند.
این ربات خط یاب دارای دو چرخ متصل به دو موتور و یک چرخ هرزگرد می باشد. دو چرخ عقب از جنس "پلی آمید" بوده و قطر آنها ٦,٢ سانتی متر و ضخامتشان ٨ میلیمتر است. جهت ایجاد اصطکاک لازم بین چرخها و سطح پیست نیز از تیوپهای نازکی که روی سطح چرخها کشیده شده، استفاده شده است.
ابعاد این ربات به شرح زیر می باشند:
طول : ١٩,٤ cm
عرض : ١٤,٣ cm

[1] Humanoid

عکسهای ربات خط یاب هوشمند فازی




SONY
SONY



فهرست منابع کتاب شناختی

۱. تفکر فازی-
نوشته بارت کاسکو؛
انتشارات «دانشگاه صنعتی خواجه نصیرالدین طوسی»

۲. میکروکنترلرهای AVR و کاربردهای آنها-
نوشته مهندس امیر ره افروز؛
انتشارات «نص»

۳. روبوتیک، مکاترونیک و هوش مصنوعی-
نوشته نیوتن سی. براگا؛
انتشارات «عبادی»

٤. ربات تعقیب خط-
نوشته عابد ناصری؛
انتشارات «فنی حسینیان»

٥. برنامه نویسی به زبان C++-

نوشته دایتل و دایتل (هاروی دایتل و پل دایتل)؛
انتشارات «شیخ بهایی»

٦. Build Your Own Robot! / Karl Lunt
A K Peters,Ltd. Natick, Massachusetts

قدر دانی

- از زحمات و راهنماییهای مفید استاد ارجمند جناب آقای دکتر محمد رضا نصیری، در کلیهٔ مراحل انتخاب، تصویب، مشاوره، طراحی و اجرای پروژهٔ "ربات خط یاب با کنترل فازی" و همچنین به موجب صرف وقت، حوصله، متانت و دلسوزی ایشان کمال تشکر و قدر دانی را دارم.
- از مساعدت دوستانی که در این هدف بنده را یاری نمودند، صمیمانه سپاسگزاری می نمایم:
  مهندس احسان میهن خواه
  مهندس مرتضی زهیری
  مهندس سهیل یارجانی مقدم
- در پایان نیز از زحمات بیشمار اساتید محترم، جناب آقای دکتر هادی رزاقیان و آقای دکتر محمد رضا نصیری در طول دوران کارشناسی که همواره از دانش و راهنماییهای

ایشان استفاده کرده ام، سپاسگزارم.