بسم الله الرحمن الرحيم

پایان‌نامه دوره کارشناسی

مهندسی برق – الکترونیک

عنوان پروژه:

مشارکت ربات ها در حل مسئله ماز

استاد راهنما:

دکتر نصور باقری

دانشجویان:

محسن رحیمی و امیرحسین رحمانی

بهمن ماه ۱۴۰۰

تشکر و قدردانی:

از همه کسانی که در این پایان نامه ما را یاری رساندند بخصوص دکتر باقری کمال تشکر را داریم.

چکیده:

توسعه سریع تکنولوژی‌ها ما را به برنامه ریزی دقیق برای بهترین انتخاب سوق می‌دهد. این فناوری و نوآوری ها باعث این شده است که کار افراد را در زندگی آسان تر کند. در این مقاله، ربات حل ماز با مهارت نقشه برداری و بومی سازی مستقل توسعه یافته است. درگام اول، ربات حل ماز با سه حسگر فاصله سنج اوولتراسونیک طراحی شده است که برای تشخیص دیوار برای جلوگیری از برخورد و برای تشخیص موانع استفاده می شود. همچنین انتظار می‌رود از ربات در محیطی که برای انسان غیرقابل دسترس است استفاده شود. علاوه بر این، مکان هایی نیز وجود دارد که استفاده از ربات ها تنها راه رسیدن به هدف است. ما با موفقیت توانایی حل ماز را بر روی ربات پیاده سازی کرده ایم. نتیجه آزمایش به‌این صورت بود که ربات می تواند ماز را با موفقیت بدون برخورد با دیوارها حل کند و همچنین مسیر صحیح را در حافظه خود ذخیره کند و در انتهای مسیر یعنی موقعی که به هدف رسید مسیر ذخیره شده را برای ربات همکار ارسال کند. در این طراحی، امکان ارسال موقعیت به ربات های دیگر تعبیه‌شده هست.

کلید واژه ها:

ماز- میکروماوس- ربات حل ماز - الگوریتم - گراف - پیمایش گراف - میکروکنترلر - ATmega 328 - سنسور ها

مقدمه

در این پروژه هدف پیاده سازی الگوریتم برای حل ماز و پیدا کردن مسیر صحیح و ارسال آن به ربات هدف است که در پنج فصل تهیه شده است.

فصل اول:

در فصل اول مقدمه ای از ربات های ماز تاریخچه و همچنین الگوریتم های مختلف و معروف برای حل آن پرداخته شده است.

فصل دوم :

در این فصل راجع به خواست پروژه و هدف آن توضیح داده شده است.

فصل سوم :

در این فصل که تحت عنوان تئوری سازی تنظیم شده راجع به انتخاب الگوریتم برای حل پروژه صحبت می‌شود.

فصل چهارم :

شبیه سازی: پیاده سازی الگوریتم بر روی میکرو ATmega328 و پلتفرم Arduino

فصل پنجم :

پیاده سازی در عمل: پیاده سازی تست الگوریتم نوشته شده در محیط واقعی آخرین فصل را تشکیل می‌دهد.

فهرست عناوین

فهرست تصاویر

فصل اول

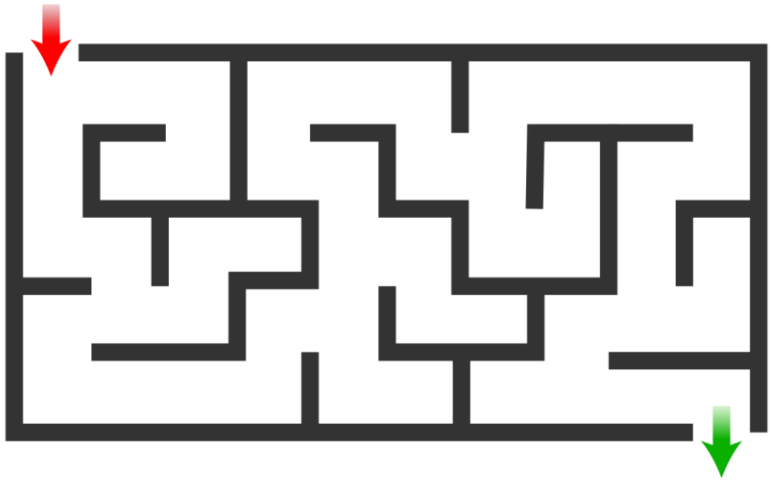
مقدمه‌ای بر ربات های ماز

۱/۱- تاریخچه ربات های حل ماز (میکروموس)

میکروماوس[1] رویدادی است که در آن ربات های کوچک ماز ۱۶*۱۶ را حل می‌کنند. که از اواخر دهه ۱۹۷۰ آغاز شد. رویدادها در سراسر جهان برگزار می‌شوند و در بریتانیا، ایالات متحده، ژاپن،سنگاپور،هند،کره جنوبی محبوبیت بیشتری دارند و در کشور های شبه قاره مانند سریلانکا محبوب هست.

۱/۲- مقدمه ای بر ربات های حل ماز

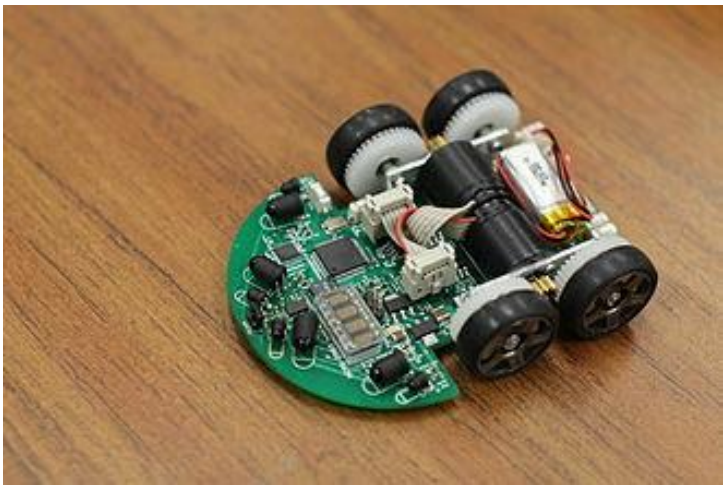
ربات های ماز یا میکرو ماوس ها گروهی از ربات ها هستند که میتوانند یک مسیر ناشناخته را طی کنند و به مقصد برسند.
مسیر ناشناخته در مسابقات و ازمایش ها یک ماز است و در عمل میتواند یک شهر یا یک مکان نامشخص باشد.
در زیر تصویر یک نمونه ماز را مشاهده میکنید.



شکل ۱.۱- مسیر ماز

[1] Micromouse

همچنین در تصویر زیر نمونه ای ربات حل ماز را مشاهده می‌کنید.



شکل ۱.۲- ربات میکروماوس

ماز از سلول ها و شبکه های ۱۶*۱۶ تشکیل شده است که هرکدام ۱۸۰ میلی متر مربع با دیوارهایی به ارتفاع ۵۰ میلی متر است. ربات های ماز، ربات‌های کاملا مستقلی هستند که باید بدون کمک راه خود را از یک موقعیت شروع از پیش تعیین شده به سمت نقطه مرکزی ماز پیدا کنند.

ربات های ماز باید مکان خود را رهگیری کنند، دیواره ها را در حین کاوش کشف کنند، ماز را ترسیم کنند و تشخیص دهد که چه زمانی به هدف رسیده است. پس از رسیدن به هدف، ماوس آن مسیر را در کمترین زمان ممکن اجرا خواهد کرد.

مسابقات میکرو ماوس تحت عنوان IEEE[3] ,UCLA ,UK[2] و کنفرانس ها تحت عنوان Minos[4] به طور منظم برگزار می‌شوند.

میکرو ماوس نیم سایز - Half-Size Micromouse

نسخه جدیدی از میکرو ماوس که نیم سایز نامیده می‌شود که در ۳۰ امین مسابقات میکرو ماوس در سال ۲۰۰۹ در ژاپن با عنوان roboLaboN معرفی شد. در نسخه جدید بجای ماز ۱۶*۱۶، در مسابقات جدید از ماز ۳۲*۳۲ استفاده شد.که ابعاد سلول ها و دیوار ها به نصف کاهش پیدا کرد.

[2] UK Micromouse and Robotics Society
[3] Micromouse USA - USA Micromouse Fans Sit
[4] Minos - UK Micromouse and Robotics Society

۱.۳-الگوریتم های حل ماز

ماوس ها می‌توانند از الگوریتم های مختلف جستجو استفاده کنند. الگوریتم های جستجو متداول از جمله روش Bellman flood-fill ، Dijkstra's الگوریتم معروف جست و جو معروف و صنعتی A^* در میان الگوریتم های مختلف پیمایش گراف ها و پیمایش درخت استفاده کرد.

۱.۴-کارایی

ربات های ماز یا همان ماوس ها بسته به طرح های ماز می‌توانند با سرعت بیش از سه متر در ثانیه حرکت کنند. برخی از بهترین سازندگان میکروموس عبارتند از: یوسوکه کاتو، نگ بنگ کیات و فومیتاکا ناکاشیما هستند. رکورد جهانی کنونی ۳.۹۲۱ ثانیه و در اختیار نگ بنگ کیات است.

عملکرد ربات ها در سال های اخیر به طور قابل توجهی بهبود یافته است. از سال ۲۰۱۵، ماوس های برنده احتمالا با شتاب رو به جلو و ترمز بالای ۱۰ متر بر ثانیه حرکت کنند. Micromice از جمله ربات‌های خودران با بالاترین عملکرد هستند.

فصل دوم

خواست و هدف پروژه

خواست پروژه:

عنوان پروژه: مشارکت ربات ها برای حل مسئله ماز

مشخصات کیفی، بلوک دیاگرام پروژه :

مسئله تعریف شده به این شکل است که دو ربات وجود دارد ۱- ربات اصلی [۵] ۲- ربات همکار[۶]

مسئله تعریف به یکی از دو شکل زیر میتواند انجام شود:

شکل اول:

ربات اصلی باید موقعیت مکانی خود را توسط سنسورها نسبت به موانع بخواند، سپس موقعیت خود را به ربات همکار ارسال کند ربات ها باید به کمک هم ماز از پیش تعیین شده را حل کنند و به مسیر انتهایی برسند.

شکل دوم:

ربات اصلی ابتدا مسیر درست ماز را باید با بهره گیری از هوش خود پیدا کند سپس مسیر درست را برای ربات همکار ارسال کند تا ربات همکار در کوتاه ترین زمان مسیر صحیح را بپیماید و به مقصد تعیین شده برسد.

شرح پروژه :

ربات اصلی با توجه به سنسور هایی که باید برای آن انتخاب شود موقعیت خود را نسبت به موانع میخواند سپس با توجه به الگوریتم انتخابی شروع به حرکت در ماز تعیین شده میکند بعد از کاوش کردن در ماز باید مسیر صحیح را ذخیره کند هم چنین در طول مسیر باید بتواند برروی موتور های خود کنترل داشته باشد تا با شتاب مناسب و ترمز های به موقع بتواند ماز را حل کند و در انتها باید بتواند با ربات همکار ارتباط برقرار کند که برای ارتباط با ربات همکار میتوانیم از ماژول های وایفای و وایرلس برای برقراری این ارتباط استفاده کنیم که ربات اصلی باید مسیر صحیح ذخیره شده بعد رسیدن به مقصد نهایی برای ربات همکار ارسال کند، سپس ربات همکار بعد از دریافت اطلاعات باید شروع به پیمودن مسیری که ربات اصلی فرستاده میکند تا به مقصد هدف برسد.

[5] Main Robot
[6] Fellow Robot

مراحل و فازهای انجام پروژه:

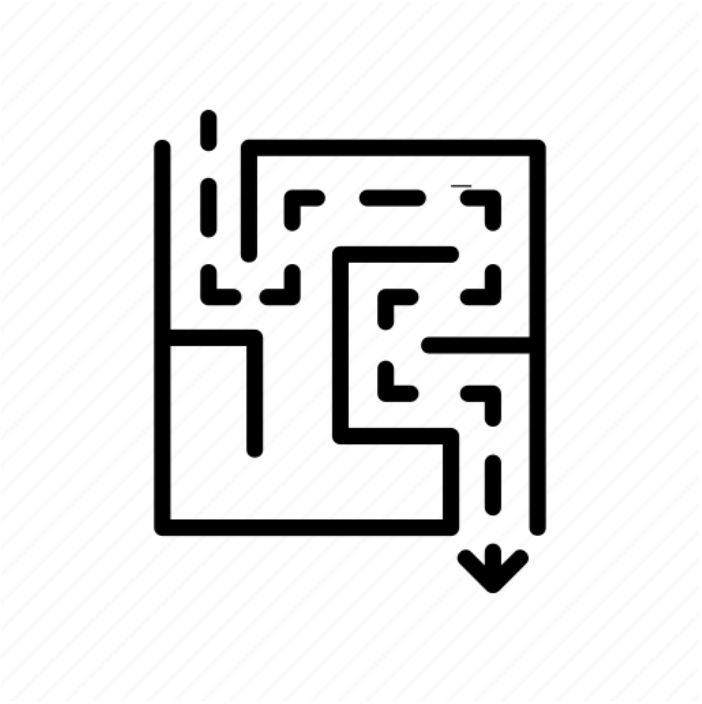
در ابتدا باید مسیر ماز طراحی و انتخاب شود، در مرحله بعد باید انتخاب شود با چه تکنولوژی و الگوریتم و ابزاری قرار است کار کنیم(برای تصمیم گیری در این باره باید به میزان بودجه و وقت خود نگاه کنیم) بعد از تصمیم گیری در این باره باید، ابزارهای لازم و مناسب تهیه شوند. سپس باید ربات طراحی شود که با توجه به مسیری که طراحی شده ابعاد ربات و طراحی آن صورت می‌گیرد. بعد از این مراحل به سراغ پیاده سازی الگوریتم برروی تکنولوژی انتخابی می‌رویم بعد از پیاده سازی الگوریتم و طراحی ربات ها در مرحله آخر باید به سراغ پیاده سازی آن در محیط واقعی برویم بعد از انجام مرحله نتیجه نهایی باید در قالب گزارش پایانی گردآوری و تهیه شود.

فصل سوم

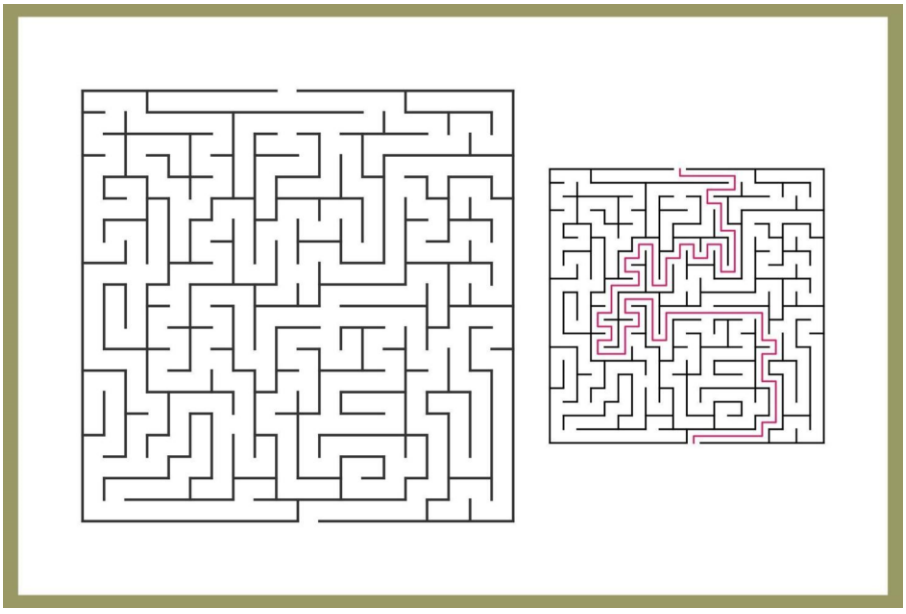
تئوری سازی

۳.۱- انتخاب مسیر

ماز ها میتوانند مسیرهای بسیار پیچیده یا مسیر های ساده باشند که در زیر چند نمونه از مسیر های ماز آورده شده است.



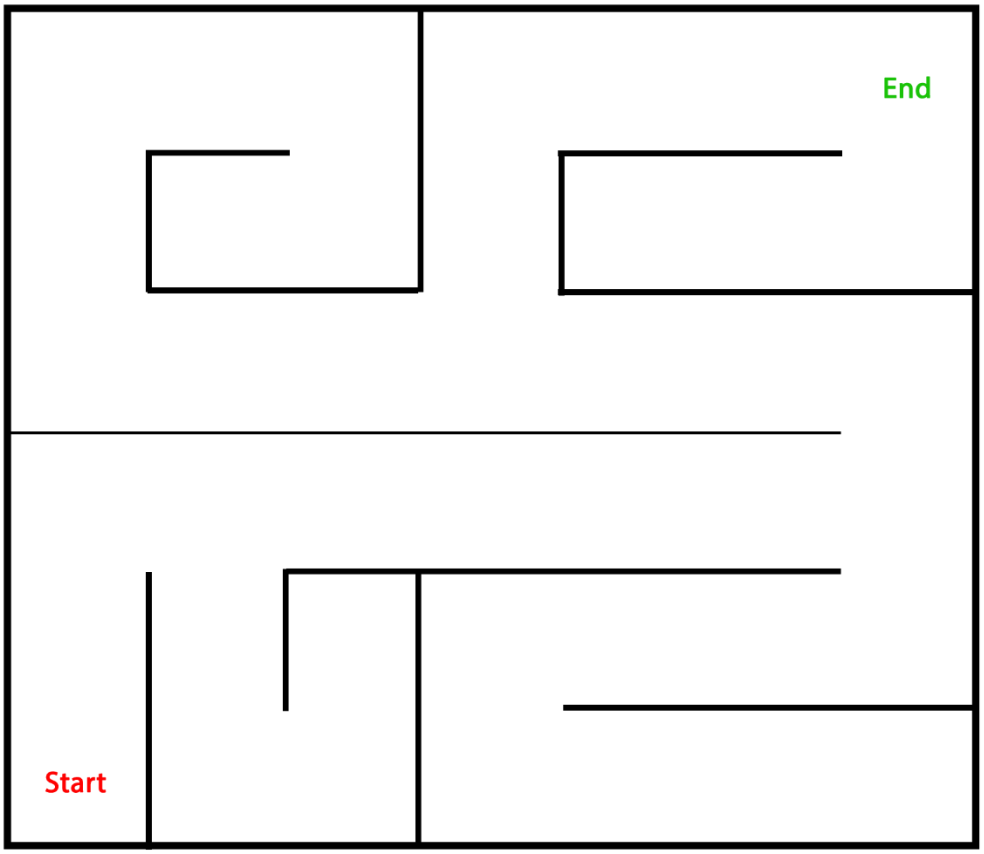
شکل۳.۱- نمونه از مسیر های ماز



شکل ۳.۲- نمونه از مسیرهای ماز

برای این پروژه ماز زیر را به عنوان مسیر انتخابی برای پروژه خود انتخاب شد :

که همانطور که مشاهده میکنید یک ماز ۶ *۶ است. که باید بر اساس مسیر انتخابی و خواسته پروژه، الگوریتم انتخاب و طراحی شود.



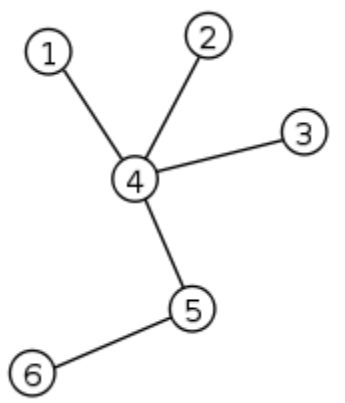
شکل ۳.۳- مسیر ماز انتخاب شده برای حل مسئله ماز

۳.۲- انتخاب الگوریتم برای حل ماز

همانطور که در فصل اول صحبت شد برای حل ماز الگوریتم های زیادی وجود دارد که چند تا از معروف ترین الگوریتم های حل ماز را در قسمت اول نام برده شد و حالا در این قسمت مفصل تر راجع به این الگوریتم ها و هم چنین الگوریتم های پیمایش گراف ها و درخت ها صحبت می‌شود.

الگوریتم های حل ماز ارتباط نزدیکی با نظریه گراف ها دارند. اگر بتوان مسیر های ماز به روش مناسب بکشد می‌توان نتیجه را شبیه درخت کرد.

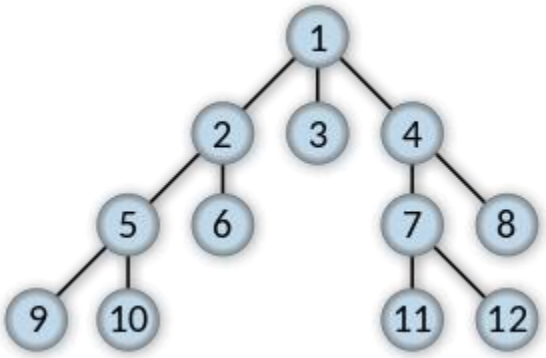
درخت: در نظریه گراف، درخت یک گراف بدون جهت است که در آن هر دو راس توسط یک مسیر به هم متصل شده است یا به به طور معادل یک گراف همبند و بدون دور است.



شکل ۳.۴- نظریه گراف

پیمایش گراف:

پیمایش گراف به معنی بازدید از تک‌تک رأس‌های گراف به نحوی خاص است. مسئله پیمایش درخت حالت خاصی از پیمایش گراف است. برای این چند الگوریتم وجود دارد.

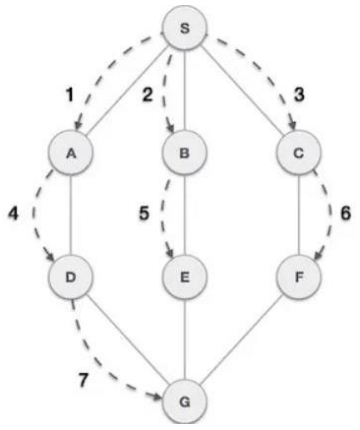


جستجوی اول سطح[7]

الگوریتم جستجوی سطح-اول، گراف را با حرکت در سطح پیمایش می‌کند و از صف برای به‌خاطرسپاری رأس‌های بعدی که می‌خواهد جستجو کند استفاده می‌کند و هر زمان که وارد بن‌بست شود از چرخه استفاده می‌کند.

شکل ۳.۵- جستجوی گراف براساس روش BFS

[7] Breadth_first search



شکل ۳.۶- جستجوی گراف براساس BFS

در مثال فوق الگوریتم سطح-اول ابتدا از رأس A به B به E و به F می‌رود و. سپس به C و در نهایت از G به D می‌رود. قواعد این پیمایش چنین هستند:

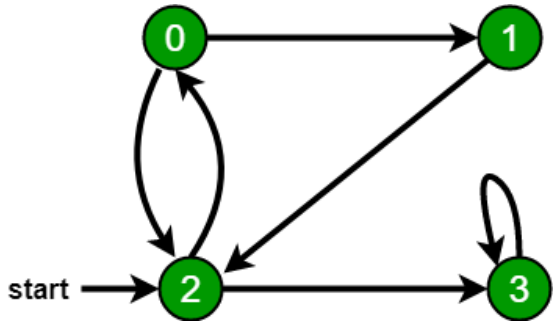
- قاعده اول: از رأس بازدید نشده مجاور بازدید کن. آن را به صورت بازدید شده نشانه‌گذاری کن، آن را نمایش بده و در صف درج کن.
- قاعده ۲: اگر هیچ رأس مجاوری وجود ندارد، نخستین رأس را از صف خارج کن.
- قاعده ۳: قواعد ۱ و ۲ را تا زمانی که صف خالی شود ادامه بده.

پیاده سازی الگوریتم جستجوی سطح اول BFS در زبان C++

```
#include<iostream>
#include <list>

using namespace std;

// This class represents a directed graph using
// adjacency list representation
class Graph
{
    int V;    // No. of vertices

    // Pointer to an array containing adjacency
    // lists
    list<int> *adj;
public:
    Graph(int V);  // Constructor

    // function to add an edge to graph
    void addEdge(int v, int w);

    // prints BFS traversal from a given source s
    void BFS(int s);
```

```
};

Graph::Graph(int V)
{
    this->V = V;
    adj = new list<int>[V];
}

void Graph::addEdge(int v, int w)
{
    adj[v].push_back(w); // Add w to v's list.
}

void Graph::BFS(int s)
{
    // Mark all the vertices as not visited
    bool *visited = new bool[V];
    for(int i = 0; i < V; i++)
        visited[i] = false;

    // Create a queue for BFS
    list<int> queue;

    // Mark the current node as visited and enqueue it
    visited[s] = true;
    queue.push_back(s);

    // 'i' will be used to get all adjacent
    // vertices of a vertex
    list<int>::iterator i;

    while(!queue.empty())
    {
        // Dequeue a vertex from queue and print it
        s = queue.front();
        cout << s << " ";
        queue.pop_front();

        // Get all adjacent vertices of the dequeued
        // vertex s. If a adjacent has not been visited,
        // then mark it visited and enqueue it
        for (i = adj[s].begin(); i != adj[s].end(); ++i)
        {
            if (!visited[*i])
            {
                visited[*i] = true;
                queue.push_back(*i);
            }
        }
    }
}

// Driver program to test methods of graph class
int main()
{
    // Create a graph given in the above diagram
    Graph g(4);
```

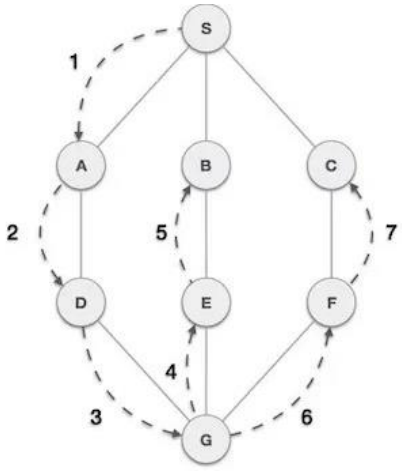
```
    g.addEdge(0, 1);
    g.addEdge(0, 2);
    g.addEdge(1, 2);
    g.addEdge(2, 0);
    g.addEdge(2, 3);
    g.addEdge(3, 3);

    cout << "Following is Breadth First Traversal "
         << "(starting from vertex 2) \n";
    g.BFS(2);

    return 0;
}
```



شکل ۳.۷-نمونه‌ی عملی از پیمایش گراف براساس الگوریتمBFS

خروجی کد برای گراف بالا براساس الگوریتم پیمایش اول سطح BFS به این صورت است:

```
Breadth First Search: 2 0 3 1
```



شکل ۳.۸- جستجوی گراف براساس DFS

پیمایش عمق اول[8]

الگوریتم جستجوی عمق –اول طوری یک گراف را پیمایش می‌کند که اولویت آن با پیمایش عمقی گراف است و از یک پشته برای به‌خاطرسپاری رأس‌های بعدی جهت جستجو استفاده می‌کند و زمانی که وارد یک بن‌بست شود، از تکرار بهره می‌گیرد.

[8] Depth First

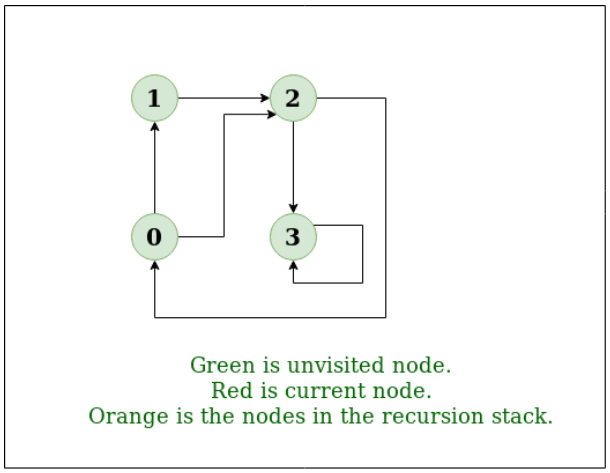
الگوریتم پیمایش عمق-اول همان طور که در شکل فوق مشخص است، ابتدا از S به A به D به G به E به B می‌رود، سپس به F و در نهایت به C می‌رود. قواعد این پیمایش چنین است:

- قاعده ۱: به رأس بازدید نشده مجاور برو. آن را به صورت بازدید شده علامت‌گذاری کن. آن را وارد پشته بکن.
- قاعده ۲: اگر رأس مجاوری نمانده باشد، یک رأس را از پشته pop کنید .همه رئوسی که رأس‌های مجاور ندارند، از پشته pop می‌شوند.
- قاعده ۳: قاعده ۱ و قاعده ۲ را تا زمانی که پشته خالی شود، ادامه بده.

پیاده سازی الگوریتم جستجوی اول عمق در زبان C++

```
// C++ program to print DFS
// traversal for a given given
// graph
#include <bits/stdc++.h>
using namespace std;

class Graph {

    // A function used by DFS
    void DFSUtil(int v);

public:
    map<int, bool> visited;
    map<int, list<int> > adj;
    // function to add an edge to graph
    void addEdge(int v, int w);

    // prints DFS traversal of the complete graph
    void DFS();
};

void Graph::addEdge(int v, int w)
{
    adj[v].push_back(w); // Add w to v's list.
}

void Graph::DFSUtil(int v)
{
    // Mark the current node as visited and print it
    visited[v] = true;
    cout << v << " ";

    // Recur for all the vertices adjacent to this vertex
    list<int>::iterator i;
    for (i = adj[v].begin(); i != adj[v].end(); ++i)
        if (!visited[*i])
```

```
        DFSUtil(*i);
}

// The function to do DFS traversal. It uses recursive
// DFSUtil()
void Graph::DFS()
{
    // Call the recursive helper function to print DFS
    // traversal starting from all vertices one by one
    for (auto i : adj)
        if (visited[i.first] == false)
            DFSUtil(i.first);
}

// Driver  Code
int main()
{
    // Create a graph given in the above diagram
    Graph g;
    g.addEdge(0, 1);
    g.addEdge(0, 9);
    g.addEdge(1, 2);
    g.addEdge(2, 0);
    g.addEdge(2, 3);
    g.addEdge(9, 3);

    cout << "Following is Depth First Traversal \n";
    g.DFS();

    return 0;
}
```



شکل ۳.۹-نمونه‌ی عملی از پیمایش گراف براساس DFS

خروجی کد برای گراف بالا براساس جستجوی اول عمق به صورت زیر است:

Output: DFS from vertex 1 : 1 2 0 3

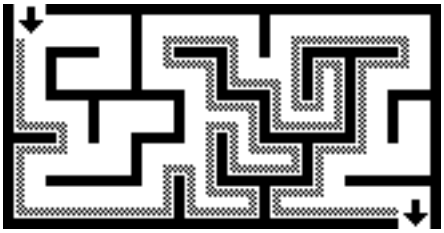
حال که دو نمونه از پرکاربرد ترین الگوریتم های جستجوی گراف‌ها را دیدیم به سراغ الگوریتم های مشهور حل ماز می‌رویم و میبینیم به چه صورت در الگوریتم های معروف و صنعتی ماز از الگوریتم های پیمایش گراف ها استفاده شده برای بهینه کردن الگوریتم برای پیدا کردن کوتاه ترین مسیر برای حل ماز.

چند نمونه از الگوریتم های ماز عبارتند از:

الگوریتم تصادفی ماوس [9]:

این روش یک روش پیش پا افتاده و ساده برای حل ماز است که می‌تواند توسط یک ربات غیر هوشمند پیاده سازی شود. این روش به این صورت است که ماوس باید مسیر جاری را دنبال کند تا به یک تقاطع برسد و سپس تصمیمی تصادفی در مورد مسیر بعدی باید بگیرد. اگرچه چنین روشی همیشه سرانجام مسیر مناسب را پیدا میکند، اما این الگوریتم یه الگوریتم بهینه نیست و بسیار کند است.

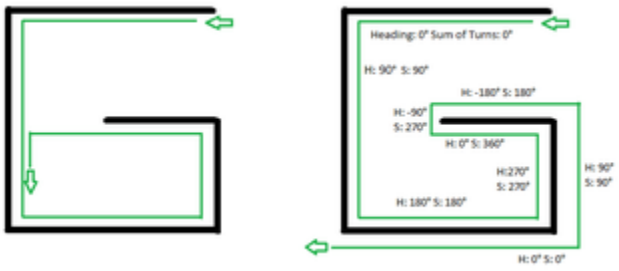
الگوریتم دنبال کردن دیواره ها[10]:



شکل ۳.۱۰-الگوریتم دنبال کردن دیواره‌ها

یکی از متداول ترین و شناخته شده ترین روش برای عبور از ماز ها با دنبال کردن دیواره ها است. که به عنوان قانون دست راست یا دست چپ نیز شناخته می‌شود. اگر ماز ساده باشد، یعنی همه‌ی دیواره‌ها به یکدیگر یا فضای بیرونی ماز متصل باشند، این تضمین وجود دارد که با حفظ تماس یک دست به دیواره‌ای که از آن شروع به حرکت می‌کنیم، بتوان ازMaze خارج شد . ولی Maze ساده نباشد، این الگوریتم به یافتن خروجی در قسمت‌های گسسته کمکی نمی‌کند. این الگوریتم یک پیمایش درخت اول عمق است.

الگوریتم pledge:



شکل ۳.۱۱ – الگوریتم PLEDGE

ماز های گسسته را می‌توان به روش دنبال کردن دیواه ها حل کرد، در صورتی که ورودی و خروجی ماز روی دیواره های خارجی ماز قرار داشته باشند.

[9] Random mouse algorithm

[10] Wall follower

چنانچه از درون ماز شروع به حرکت کنیم، ممکن است الگوریتم دنبال کردن دیواره ها در قسمت گسسته ای که شامل خروجی نیست دائما یک حلقه را طی کند.الگوریتم Pledge می‌تواند این گونه مسائل را حل کند.

الگوریتم Pledge برای رفع موانع، به طی یک مسیر اختیاری نیاز دارد. هنگام مواجهه با مانع، یک دست (مثال دست راست) را در امتداد مانع نگه می‌داریم در حالیکه زوایای چرخش شمرده می‌شود. وقتی دوباره در راستای مسیر اصلی قرار گرفتیم و جمع زاویه ای چرخش ها برابر صفر شد، می‌توان مانع را ترک کرد و در راستای مسیر اصلی حرکت نمود. این الگوریتم به شخص اجازه ی جهت یابی را در شروع از هر نقطه برای خارج از مازهای دوبعدی ، را می‌دهد.

الگوریتم Trémaux's

الگوریتم ترمو، که توسط چارلز پیر ترمو [11]ابداع شد، یک روش کارآمد برای یافتن راه خروج از ماز است که برای مشخص کردن یک مسیر نیاز نیاز به کشیدن خطوط روی زمین دارد و تضمین می‌شود که برای هر ماز خوش تعریف که تقاطع های کاملا مشخص دارد کار کند. اما تضمینی برای یافتن کوتاه ترین مسیر وجود ندارد.

الگوریتم براساس قوانین زیر کار می‌کند:

- هر مسیری را که دنبال میکنید یک بار باید علامت بزنید.علائم باید در هر دو انتهای مسیر قابل مشاهده باشند. بنابراین، اگر آنها به عنوان علائم فیزیکی ساخته می‌شوند، نه این‌که به عنوان بخشی از یک الگوریتم رایانه ذخیره شوند، باید همان علامت در هر دو انتهای مسیر ایجاد شود.
- هرگز نباید وارد مسیری که در آن دو علامت وحود دارد شوید
- اگر به تقاطعی رسیدید که هیچ علامتی ندارد (به جز احتمالا موردی که در مسیری که وارد شده اید)، یک مسیر بدون علامت دلخواه انتخاب کنید، آن را علامت بزنید.
- در غیر این صورت:
  - اگر مسیری که وارد آن شده اید فقط یک علامت داشت، بچرخید و از آن مسیر برگردید و آن را علامت بزنید. به طور خاص باید این مورد هر وقت که بن بست رسیدید رخ دهد.
  - اگر نه، یکی از مسیرهای باقیمانده را با کمترین علامت (در صورت امکان بدون علامت، و در غیر این‌صورت یک) انتخاب کنید، و آن مسیر را دنبال کنید و آن را علامت بزنید.

قانون بچرخ و برگرد[12] به طور موثر هر ماز دارای حلقه را به یک ماز ساده تبدیل می‌کند. هرگاه راهی را پیدا کنیم که که حلقه ای را ببندد، آن را به عنوان یک بن بست در نظر می‌گیریم و برمیگردیم. بدون این قانون

[11] Charles Pierre Trémaux
[12] turn around and return rule

یعنی اگر به جای برگشتن به عقب، خودسرانه مسیر دیگری را دنبال کنیم، ممکن است دستری به بخش های هنوز کشف نشده ماز را قطع کنیم. زمانی که در نهایت به هدف رسیدید، مسیرهایی که دقیقا یکبار علامت گذاری شده اند، درواقع راه بازگشت به نقطه شروع را نشان میدهند. اگر خروجی وجود نداشته باشد، این روش شما را به نقطع شروع باز میگرداند که در آن همه مسیر ها دوبار علامت گذاری شده اند. در این مورد، هر مسیر دقیقا دوبار، یک بار در هر جهت طی می‌شود.

اساسا این الگوریتم که در قرن نوزدهم کشف شد، حدود صد سال بعد به عنوان جستجو در عمق [۱۳]مورد استفاده قرار گرفت. که پیش تر راجع به الگوریتم جستجو در عمق در گراف ها صحبت کردیم.

الگوریتم Dead-end filling

الگوریتم Dead-end filling یک الگوریتم برای حل ماز است که بن بست ها را پر می‌کند، و تنها راه های صحیح را باز می‌گذاریم. می‌توان از این الگوریتم برای حل ماز روی کاغذ یا برنامه های کامپیوتری استفاده کرد. ام این روش برای ماز های نا شناخته مناسب نیست چراکه در این روش باید تمام ماز را بررسی کند. روش بدین صورت است که ابتدا باید همه بن بست ها در ماز را پیدا کنیم و سپس مسیر را از هر بن بست تا رسیدن به اولین تقاطع پرکنیم.

الگوریتم بازگشتی [14]

اگر یک پیشنمایش از ماز به شما داده شود، یک الگوریتم بازگشتی ساده می‌تواند به شما بگوید چگونه به پایان برسد. به الگوریتم یک مقدار X,Y اولیه داده می‌شود. اگر مقدار X,Y روی دیوار نباشند، متد خود را با تمام مقادیر X,Y مجاور بررسی می‌کند و مطمئن می‌شود که قبلا از مقادیر X,Y استفاده نکرده است. اگر مقادیر X,Y مربوط به نقاط انتهایی باشد، تمام نمونه های قبلی را به عنوان مسیر صحیح ذخیره می‌کند.

این روش درواقع یک پیمایش اول عمق از گراف هاست که بر حسب نقاط سلول ها و یا شبکه بیان می‌شود.

در ادامه نمونه کدی از الگوریتم بازگشتی در زبان جاوا آمده است:

```
boolean[][] maze = new boolean[width][height]; // The maze
boolean[][] wasHere = new boolean[width][height];
boolean[][] correctPath = new boolean[width][height]; // The solution to the
maze
int startX, startY; // Starting X and Y values of maze
int endX, endY;     // Ending X and Y values of maze
```

[13] depth-first search.

[14] Recursive algorithm

```
public void solveMaze() {
    maze = generateMaze(); // Create Maze (false = path, true = wall)
    for (int row = 0; row < maze.length; row++)
        // Sets boolean Arrays to default values
        for (int col = 0; col < maze[row].length; col++){
            wasHere[row][col] = false;
            correctPath[row][col] = false;
        }
    boolean b = recursiveSolve(startX, startY);
    // Will leave you with a boolean array (correctPath)
    // with the path indicated by true values.
    // If b is false, there is no solution to the maze
}
public boolean recursiveSolve(int x, int y) {
    if (x == endX && y == endY) return true; // If you reached the end
    if (maze[x][y] || wasHere[x][y]) return false;
    // If you are on a wall or already were here
    wasHere[x][y] = true;
    if (x != 0) // Checks if not on left edge
        if (recursiveSolve(x-1, y)) { // Recalls method one to the left
            correctPath[x][y] = true; // Sets that path value to true;
            return true;
        }
    if (x != width - 1) // Checks if not on right edge
        if (recursiveSolve(x+1, y)) { // Recalls method one to the right
            correctPath[x][y] = true;
            return true;
        }
    if (y != 0)  // Checks if not on top edge
        if (recursiveSolve(x, y-1)) { // Recalls method one up
            correctPath[x][y] = true;
            return true;
        }
    if (y != height - 1) // Checks if not on bottom edge
        if (recursiveSolve(x, y+1)) { // Recalls method one down
            correctPath[x][y] = true;
            return true;
        }
    return false;
}
```

الگوریتم کوتاه ترین مسیر[15]

هنگامی‌که یک ماز چندین راه حل دارد، حل کننده ممکن است بخواهد کوتاه ترین مسیر را از ابتدا تا انتها پیدا کند. الگوریتم های مختلفی برای پیدا کردن کوتاه ترین مسیرها وجود دارد که بیشتر آن ها از نظریه گراف ها می‌آیند که پیشتر راجع به آن ها صحبت کردیم.

[15] Shortest path algorithm

یکی از الگوریتم های کوتاه ترین مسیر را با پیاده سازی یک الگوریتم پیمایش اول-عمق پیدا می‌کند. در حالی که الگوریتم *A از تکنیک اکتشافی استفاده می‌کند. الگوریتم پیمایش سطح اول (BFS) از یک صف برای بازدید از سلول ها به ترتیب فاصله از شروع تا پایان استفاده می‌کند.

هر سلول بازدید شده باید فاصله خود را از ابتدا دنبال کند یا این‌که بررسی کند که کدام سلول مجاور نزدیکتر به شروع باعث اضافه شدن آن به صف شده است. هنگامی که مکان انتهایی یافت شد، مسیر سلول ها را به سمت عقب تا شروع کوتاه ترین مسیر دنبال می‌کند، پیمایش سطح اول در ساده ترین شکل ممکن محدودیت های خود را دارد مانند پیدا کردن کوتاه ترین مسیر در گراف های وزن‌دار.

الگوریتم Flood Fill

الگوریتم Flood Fill یکی از بهترین الگوریتم های حل ماز است. این الگوریتم مقادیری را به هریک از خانه ها یا سلول ها اختصاص می‌دهد که این مقادیر نشان دهنده فاصله هر سلول روی ماز تا سلول مقصد است.

طرزکار الگوریتم Flood Fill

طرزکار الگوریتم بر مبنای BFS

۱. یک صف از جفت اعداد ایجاد کنید.
۲. مقدار دهی اولیه را وارد صف کنید.
۳. یک آرایه دو بعدی برای مکان های بازدید شد ایجاد و مقدار دهی کنید به عنوان مثال [][]vis.
۴. تا زمانی که صف خالی نباشد مراحل ۴.۱ تا ۴.۶ را تکرار کنید.
   - یک عنصر جلویی را از صف بگیرید.
   - آن را از صف پاک کنید.
   - ذخیره کردن مقدار/ رنگ فعلی در مختصاتی که از صف خارج شده است. (مقدار قبلی)
   - مقدار/ رنگ فعلی که از صف خارج شده است را بروز کنید.
   - هر ۴ جهت را بررسی کنید یعنی (x+1,y), (x-1,y) ، (x,y+1) ، (x,y-1) معتبر است یا خیر اگر معتبر بود بررسی کنید مقدار آن در آن مختصات باید برابر با مقدار یا رنگ قبلی باشد و مقدار آن در مختصات در آرایه‌ی [][]vis باید ۰ باشد.
   - اگر همه شرایط بالا درست باشد، مختصات مربوطه را در صف وارد کنید و به عنوان ۱ در آرایه []vis[] قرار دهید.
   - آرایه ۲ بعدی را چاپ کنید.

پیاده سازی این الگوریتم در زبان C++

```
// C++ program for above approach
#include <bits/stdc++.h>
using namespace std;

// Function to check valid coordinate
int validCoord(int x, int y, int n, int m)
{

    if (x < 0 || y < 0) {
        return 0;
    }
    if (x >= n || y >= m) {
        return 0;
    }
    return 1;
}

// Function to run bfs
void bfs(int n, int m, int data[][8],
                     int x, int y, int color)
{

  // Visiting array
  int vis[101][101];

  // Initialing all as zero
  memset(vis, 0, sizeof(vis));

  // Creating queue for bfs
  queue<pair<int, int> > obj;

  // Pushing pair of {x, y}
  obj.push({ x, y });

  // Marking {x, y} as visited
  vis[x][y] = 1;

  // Until queue is empty
  while (obj.empty() != 1)
  {

    // Extracting front pair
    pair<int, int> coord = obj.front();
    int x = coord.first;
    int y = coord.second;
    int preColor = data[x][y];

    data[x][y] = color;

    // Popping front pair of queue
    obj.pop();

    // For Upside Pixel or Cell
    if (validCoord(x + 1, y, n, m)
```

```
        && vis[x + 1][y] == 0
        && data[x + 1][y] == preColor)
    {
      obj.push({ x + 1, y });
      vis[x + 1][y] = 1;
    }

    // For Downside Pixel or Cell
    if (validCoord(x - 1, y, n, m)
        && vis[x - 1][y] == 0
        && data[x - 1][y] == preColor)
    {
      obj.push({ x - 1, y });
      vis[x - 1][y] = 1;
    }

    // For Right side Pixel or Cell
    if (validCoord(x, y + 1, n, m)
        && vis[x][y + 1] == 0
        && data[x][y + 1] == preColor)
    {
      obj.push({ x, y + 1 });
      vis[x][y + 1] = 1;
    }

    // For Left side Pixel or Cell
    if (validCoord(x, y - 1, n, m)
        && vis[x][y - 1] == 0
        && data[x][y - 1] == preColor)
    {
      obj.push({ x, y - 1 });
      vis[x][y - 1] = 1;
    }
  }

  // Printing The Changed Matrix Of Pixels
  for (int i = 0; i < n; i++)
  {
    for (int j = 0; j < m; j++)
    {
      cout << data[i][j] << " ";
    }
    cout << endl;
  }
  cout << endl;
}

// Driver Code
int main()
{
  int n, m, x, y, color;
  n = 8;
  m = 8;

  int data[8][8] = {
    { 1, 1, 1, 1, 1, 1, 1, 1 },
    { 1, 1, 1, 1, 1, 1, 0, 0 },
```

```
    { 1, 0, 0, 1, 1, 0, 1, 1 },
    { 1, 2, 2, 2, 2, 0, 1, 0 },
    { 1, 1, 1, 2, 2, 0, 1, 0 },
    { 1, 1, 1, 2, 2, 2, 2, 0 },
    { 1, 1, 1, 1, 1, 2, 1, 1 },
    { 1, 1, 1, 1, 1, 2, 2, 1 },
  };

  x = 4, y = 4, color = 3;

  // Function Call
  bfs(n, m, data, x, y, color);
  return 0;
}
```

خروجی الگوریتم به صورت زیر در می‌آید:

```
1 1 1 1 1 1 1 1
1 1 1 1 1 1 0 0
1 0 0 1 1 0 1 1
1 3 3 3 3 0 1 0
1 1 1 3 3 0 1 0
1 1 1 3 3 3 3 0
1 1 1 1 1 3 1 1
1 1 1 1 1 3 3 1
```

خب تا اینجا با معروف ترین الگوریتم های حل ماز آشنا شدیم حالا برای قسمت انتهایی یعنی برای زمانی که ربات اصلی مسیر درست را پیدا کرد و قرار است با ربات همکار جهت تبادل اطلاعات ارتباط برقرار کند صحبت میکنیم که چگونه ربات می‌تواند با ربات همکار ارتباط برقرار کند.

۳.۵ روش های ارتباطی

روش‌های مختلفی برای انتقال اطلاعات وجود دارد که هر کدام از آن‌ها در جای خود کاربردی هستند. می‌بایست با توجه به کاربرد و امکانات موجود، بهترین روش را انتخاب کرده تا با بهترین هزینه به بالاترین کیفیت دست یابیم.

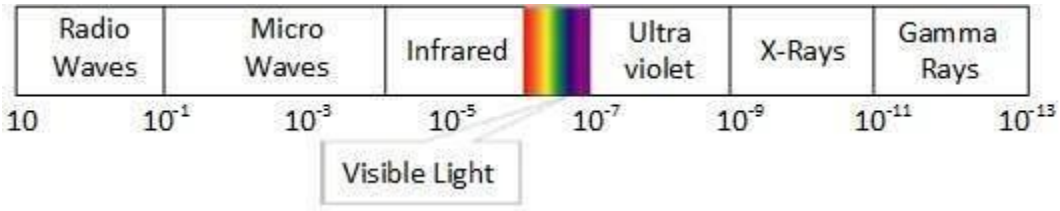
چون در این پروژه اطلاعات باید به روش بی سیم انتقال یابد بنابراین از روش های دیگر ارتباطی نظیر با سیم صحبت نمی‌شود.

انتقال بی سیم:

انتقال بی سیم شکلی از رسانه های هدایت نشده است. ارتباط بی سیم هیچ پیوند فیزیکی بین دو یا چند دستگاه وجود ندارد و بصورت بی سیم این ارتباط شکل می‌گیرد. سیگنال های بی سیم در هوا پخش می‌شوند و توسط آنتن های مناسب دریافت و تفسیر می‌شوند.

زمانی که آنتن به مدار الکتریکی یک کامپیوتر یا دستگاه بی سیم متصل می‌شود، داده های دیجیتال را به سیگنال های بی سیم تبدیل می‌کند و در محدوده فرکانسی خود پخش می‌شود. گیرنده در سمت دیگر این سیگنال ها را دریاقت و آن ها را به داده های دیجیتال تبدیل می‌کند.

یه قسمت کوچکی از طیف های الکترو مغناطیسی را می‌توان برای انتقال بی سیم استفاده کرد.



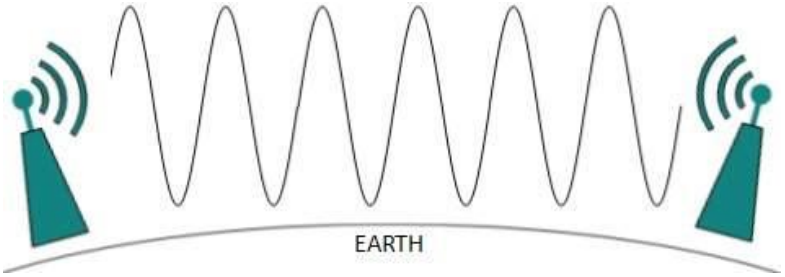
شکل ۳.۱۲- طیف‌های الکترومغناطیسی

انتقال رادیویی[۱۶]

فرکانس های رادیویی آسان تر تولید می‌شوند و به دلیل طول موج بالایی که دارند می‌توانند از طریق دیواره ها و ساختارها به طور یکسان نفوذ کند. فرکانس های رادیویی به شش باند تقسیم می‌شوند.

امواج رادیویی در فرکانس های پایین‌تر می‌توانند از دیوارها عبور کنند، درحالی که فرکانس های رادیویی بالاتر می‌توانند در خط مستقیم حرکت کنند و به عقب برگردند. قدرت امواج فرکانس پایین در مسافت های طولانی به شدت کاهش پیدا می‌کند. امواج رادیویی فرکانس بالا قدرت بیشتری دارند.

فرکانس های پایین تر مانند باند های VLF, LF, MF می‌توانند ۱۰۰۰ کیلومتر را بروی سطح زمین طی‌کنند.



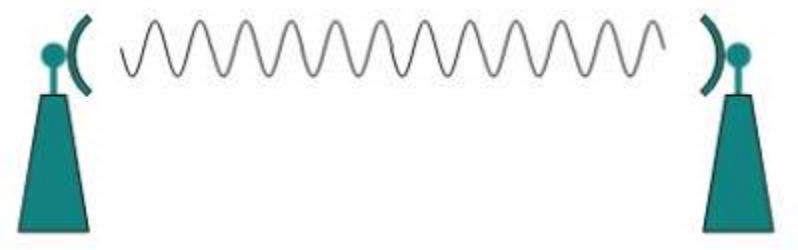
شکل۳.۱۳ – انتقال رادیویی

[16] Radio Transmission

انتقال مایکروویو[17]

امواج الکترومغناطیسی بالای ۱۰۰ مگاهرتز تمایل دارند در یک خط مستقیم حرکت کنند و سیگنال ها را می‌توان با تابش آن امواج به سمت یک ایستگاه خاص ارسال کرد. از آنجایی که امواج مایکروویو در خطوط مستقیم حرکت می‌کنند، فرستنده و گیرنده هردو باید در یک راستا باشند تا کاملا در خط دید هم دیگر قرار بگیرند.

امواج مایکروویو می‌توانند دارای طول موج بین ۱ میلی متر تا ۱ متر و فرکانس بین ۳۰۰ مگاهرتز تا ۳۰۰ گیگاهرتز باشند.



شکل ۳.۱۴- انتقال مایکروویو

آنتن های مایکروویو امواج را متمرکز می‌کنند و پرتویی را از آن می‌سازند. همانطور که در تصویر بالا نشان داده شده است، چندین آنتن را می‌توان برای رسیدن به نقاط دورتر تراز کرد. امواج مایکروویو فرکانس بالاتری دارند و در دیوار نفوذ نمی‌کنند.
انتقال مایکروویو به شدت به شرایط آب و هوایی و فرکانس مورد استفاده بستگی دارد.

انتقال مادون قرمز[18]

موج مادون قرمز بین طیف مرئی و امواج مایکروویو قرار دارد. دارای طول موج ۷۰۰ نانومتر تا ۱ میلی متر و محدوده فرکانس از ۳۰۰ گیگاهرتز تا 430 THZ است.

امواج مادون قرمز برای اهداف ارتباطی با برد بسیار کوتاه همانند تلویزیون و ریموت ها استفاده می‌شود. مادون قرمز در یک خط مستقیم حرکت می‌کند، ازین رو طبیعتا جهت دار است. به دلیل محدوده فرکانس بالا، مادون قرمز نمی‌تواند از موانع مانند دیوار عبورکند.

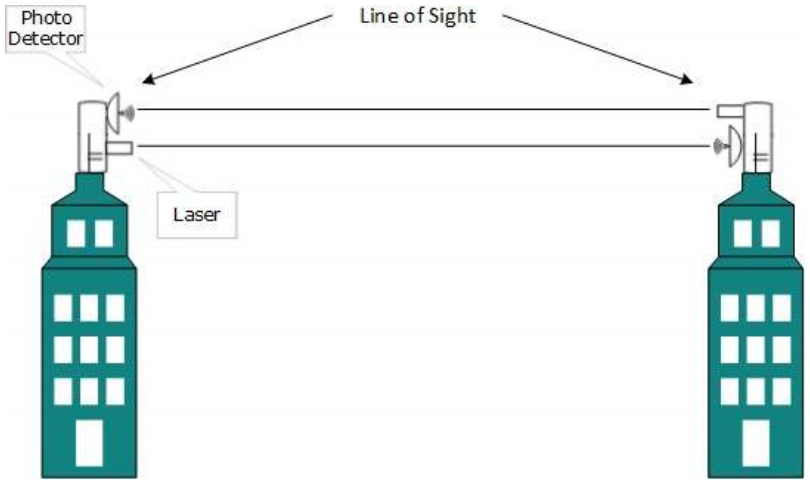
[17] Microwave Transmission

[18] Infrared Transmission

انتقال نوری[19]

بالاترین طیف الکترومغناطیسی که می‌تواند برای انتقال داده استفاده شود، سیگنالینگ اپتیک یا نوری است. که این امر به وسیله لیزر محقق می‌شود.

به دلیل استفاده از نور، فرکانس تمایل دارد که کاملا در خط مستقیم حرکت کند. بنابراین فرستنده و گیرنده باید در خط دید هم دیگر باشند. از آنجایی که انتقال لیزر یک جهته است، در هردو انتهای ارتباط، لیزر و ردیاب باید عکس هم نصب شوند. پرتو لیزر عموما ۱ میلی متر عرض دارد، در نتیجه دقت کار برای تراز کردن دو گیرنده که دور از هم قرار دارند به وسیله منابع لیزر بالاست.



شکل ۳.۱۵-انتقال نوری

لیزر به عنوان فرستنده (TX) و آشکارسازهای عکس به عنوان گیرنده (RX) کار می‌کند.

لیزرها نمی‌توانند از موانعی مانند دیوار ها، باران، مه غلیظ نفوذ کنند. علاوه بر آن، پرتو لیزر توسط باد، دمای اتمسفر یا تغییر دما در مسیر تغییر می‌کند. لیزر یک انتقال داده امن است زیرا آسیب زدن به لیزر با عرض ۱ میلی متر بدون قطع کانال ارتباطی بسیار دشوار است.

خب راجع به روش های انتقال داده بی سیم به صورت کلی صحبت شد. به طور خاص برای این پروژه از روش وایرلس برای ایجاد این ارتباط بین دو دستگاه استفاده شده است .

[19] Light Transmission

برای ارتباط وایرلس بین دو دستگاه از ماژول NRF24L01 استفاده شده است، راجع این ماژول و روش راه اندازی آن به طور مفصل در فصل بعد پرداخته شده است.

شکل۳.۱۶- ماژول وایرلس NRF24L01

این چیپ در واقع یک ماژول بسیار عالی برای ارسال و دریافت اطلاعات از طریق وایرلس با خطای بسیار پایین است. چون فرکانس بالایی در حدود ۲.۴ گیگاهرتز دارد براحتی از اجسامی همانند دیوار و موانع دیگر عبور می‌کند و باعث می‌شود برد بیشتری به ما بدهد.

فصل چهارم

ابزارها و راه‌اندازی آن ها

۴.۱.معرفی میکروکنترلر[20] استفاده شده

ابتدا یه تعریفی از میکروکنترلر داشته باشیم ببینم میکروکنترولر چیست؟

میکروکنترولر چیست؟

میکروکُنترولر (به انگلیسی: Microcontroller) گونه‌ای ریزپردازنده است که دارای حافظه دسترسی تصادفی (RAM) و حافظه فقط‌خواندنی (ROM)، تایمر، پورت‌های ورودی و خروجی (I/O) و درگاه ترتیبی (Serial Port پورت سریال) درون خود تراشه است، و می‌تواند به تنهایی ابزارهای دیگر را کنترل کند. به عبارت دیگر یک میکروکنترلر، مدار مجتمع کوچکی است که از یک CPU کوچک و اجزای دیگری مانند تایمر، درگاه‌های ورودی و خروجی آنالوگ و دیجیتال و حافظه تشکیل شده‌است.

میکروکنترولر یا ریز کنترل گر یک تراشه IC است که درواقع مغز ربات شما است که برای کنترل دستگاه های دیگر برنامه نویسی می‌شود.

انواع میکروکنترلر

میکروکنترلرها انواع گوناگونی دارند. میکروکنترلرهای ARM ، میکروکنترلرهای AVR ، میکروکنترلر atx mega، میکروکنترلرهایplc ، میکروکنترلرهای ۸۰۵۱ و غیره بخشی از انواع میکروکنترلر است.

برای ساخت ربات های میکروماوس بسته به وقت و هزینه و کارایی که از ربات انتظار دارید می‌توانید یکی از میکروکنترلر ها را انتخاب کنید.

خب با توجه به وقت محدودی که در اختیار داشتیم از پلت فرم آردوینو [21] استفاده کردیم که این پلت فرم از میکرو کنترلر AVR Atmega328 ساخته شده است.

حال برای آن که بیشتر با آردواینو آشنا بشیم به تصویر زیر نگاه کنید که یک نمونه از پلت فرم آردوینو است.

شکل ٤،١ – آردوینو مدل اونو

[20] Microcontroller
[21] Arduino

آردوینو

آردوینو (به انگلیسی: Arduino) یک پلتفرم سخت‌افزاری و نرم‌افزاری متن‌باز است. پلتفرم آردوینو شامل یک میکروکنترلر تک‌بردی متن‌باز است که قسمت سخت‌افزار آردوینو را تشکیل می‌دهد. علاوه بر این، پلتفرم آردوینو یک نرم‌افزار آردوینو IDE که به منظور برنامه‌نویسی برای بردهای آردوینو طراحی شده‌است و یک بوت لودر نرم‌افزاری که بر روی میکروکنترلر بارگذاری می‌شود را در بر می‌گیرد. پلتفرم آردوینو به منظور تولید سریع و ساده پروژه‌های سخت‌افزاری تعاملی و ساخت وسایلی که با محیط تعامل داشته باشند طراحی شده‌است،البته بردهای آردوینو اهداف آموزشی را نیز دنبال می‌کنند.

توجه داشته باشید همانطور که در توضیحات بالا اشاره کردیم آردواینو ابزار مناسبی برای ساخت دستگاه های صنعتی و حرفه ای نیست و صرفا به عنوان یک ابزار مناسب و سریع و ارزان برای اهداف آزمایشگاهی و آموزشی و پروژه های ساده و نیمه حرفه ای دانشگاهی و ... مناسب است.

اگر قصد ساخت ربات حرفه ای برای مقاصد تجاری و صنعتی و مسابقات مختلف رسمی را دارید آردوینو گزینه مناسبی نیست.

۴.۲. انتخاب سنسور ها

خب بعد از این‌که میکرو مورد نظر را انتخاب کردیم برای این‌که بتوانیم مانع ها را در مسیر تشخیص دهیم باید سنسور هایی را انتخاب کرد که بتوانند این کار را انجام دهند که در ادامه به چند نمونه ازین سنسور ها اشاره شده است.

سنسور:

سنسور(sensor) یعنی حس کننده و از کلمه sens به معنی حس کردن گرفته شده و می تواند کمیت هایی مانند فشار، حرارت، رطوبت، دما، و ... را به کمیت های الکتریکی پیوسته (آنالوگ) یا غیرپیوسته (دیجیتال) تبدیل کند.

اما سنسور ها در ربات :

سنسورها اغلب برای درک اطلاعات تماسی، تنشی، مجاورتی، بینایی و صوتی بهکار میروند. عملکرد سنسورها بدینگونه است که با توجه به تغییرات فاکتوری که نسبت به آن حساس هستند، سطوح ولتاژی ناچیزی را در پاسخ ایجاد میکنند، که با پردازش این سیگنالهای الکتریکی میتوان اطلاعات دریافتی را تفسیر کرده و برای تصمیمگیریهای بعدی از آنها استفاده کرد.

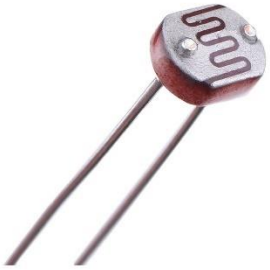
خب در این پروژه چون با سنسور های محیطی سرکار دارد فقط به این سنسور ها اشاره شده است.

سنسور محیطی چیست؟

این سنسور ها درواقع اطلاعات رو از میحط خارج و موقعیت ربات با اشیای اطراف را دریافت می‌کنند.

برای این‌که بتوان موقعیت ربات با موانع اطرافش را دریافت کرد می‌توان از سنسور هایی نظیر سنسور نوری، سنسور حسگر مجاورتی مادون قرمز، آلتراسونیک، سنسور IR و.. استفاده کرد.

سنسور شدت نور



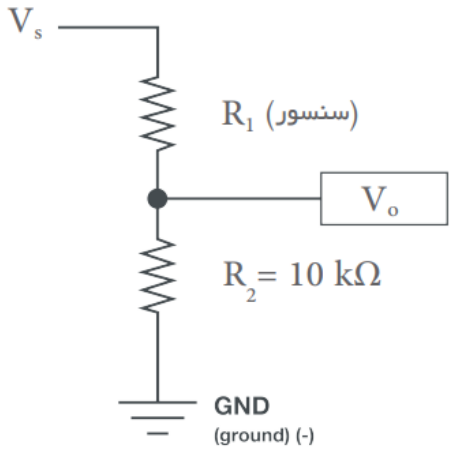
برخی از سنسور ها تنها دو پایه دارند. این سنسور ها درواقع، مقاومت هایی هستند که در اثر قرار گیریدر وضعیت های مختلف، مقاومت آن ها تغییر می‌کند.

سنسور شدت نور یکی از این سنسور هاست. در اثر افزایش شدت نور، مقاومت الکتریکی این سنسور کاهش می‌یابد.

با توجه به این‌که آردواینو نمی‌تواند مقاومت را مستقیما اندازه گیری کند، باید به طریقی تغییرات در مقاومت سنسور را به تغییرات ولتاژ تبدیل کرد.

یکی از رایج ترین روش‌ها برای این کار، استفاده از تقسیم ولتاژ است.

شکل ۴.۲ - سنسور شدت نور



$$Vo = Vs * \frac{R2}{R1+R2}$$

شکل ۴.۳ – مدار راه‌اندازی سنسور شدت نور

۴.۳. سنسور فاصله سنج اولتراسونیک[۲۲]

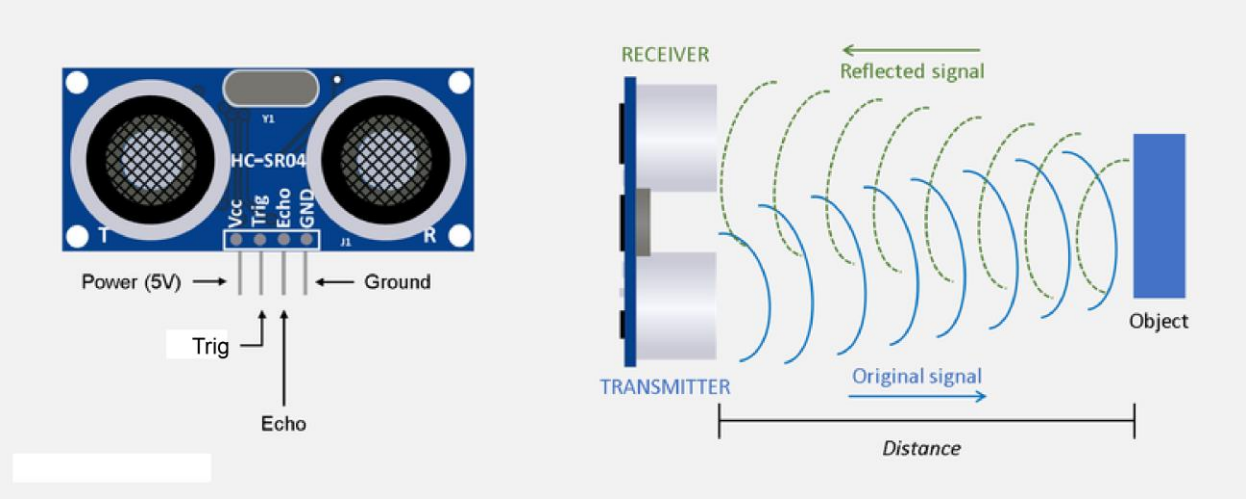
ماژول فاصله سنج اولتراسونیک با ارسال یک موج صوتی و محاسبه زمان بازگشت موج به سنسور پس از برخورد با مانع، فاصله سنسور تا مانع را محاسبه کند.

این سنسور برای ساخت ربات های ماز و ربات های obstacle avoidance و کاربردهایی ازین قبیل استفاده می‌شود.

در این پروژه از این نوع سنسور ها برای ربات استفاده شده که کار با آن آسان است.

[22] Ultrasonic sensor

که در ادامه نحوه راه اندازی این ماژول را توضیح خواهیم داد:



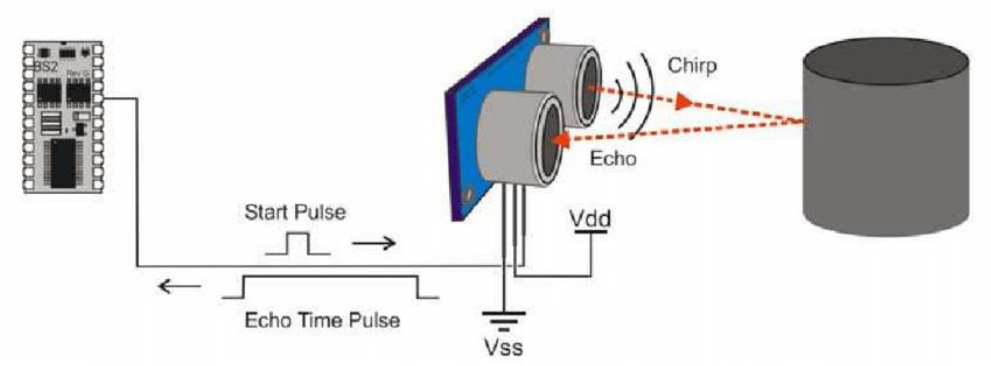
شکل ۴.۴- طرز کار ماژول اوولتراسونیک

همانوطر که در تصویر بالا می‌بینید ماژول HC-SR04 چهار پایه دارد. پایه های VCC و GND به ترتیب به ۵ ولت و GND آردواینو متصل می‌شود. و دو پایه دیگر یعنی Trig و Echo به پین های دیجیتال آردواینو متصل می‌شود هرچند در آردواینو میتواند پایه های آنالوگ را هم به دیجیتال تبدیل کرد بنابراین در صورت کمبود فضا می‌توانید از پایه های آنالوگ آردواینو به عنوان پایه های دیجیتال استفاده کنید

طرز کار اولتراسونیک:

با ارسال موج فراصوتی توسط سنسور فاصله سنج اولتراسونیک و محاسبه زمان بازگشت آن پس از برخورد با مانع، میتوان فاصله سنسور تا مانع را بدست آورد. امواج فراصوتی مانند امواج صوتی، با سرعت ۳۴۰ متر بر ثانیه در فضاحرکت می‌کنند.

لازم به ذکر است که مسافت طی شده توسط موج فراصوتی از ارسال تا برگشت، دو برابر فاصله سنسور از مانع است; در نتیجه برای بدست آوردن فاصله، باید مسافت طی شده توسط موج، در بازه ارسال تا دریافت موج را تقسیم بر دو کرد.



شکل ۴.۵- طرز کار ماژول اوولتراسونیک

باتوجه به سرعت بالای موج فراصوتی، در محاسبات و کدنویسی از واحد میکروثانیه (10^{-6} ثانیه) استفاده می‌شود. در نتیجه سرعت صوت به جای ۳۴۰ متر بر ثانیه ۰.۰۳۴ سانتی متر بر میکروثانیه محاسبه می‌شود.

نحوه ایجاد موج فراصوتی:

برای ایجاد موج فراصوتی، ابتدا ۲ میکروثانیه، پایه Trig در وضعیت LOW قرار داده می‌شود تا اطمینان حاصل شود که پین Trig از قبل در وضعیت HIGH قرار ندارد. سپس به مدت ۱۰ میکروثانیه، این پین در وضعیت HIGH قرار داده می‌شود. با استفاده از دستور pulseIn در آردواینو، زمان رسیدن پالس ارسالی به پین Echo اندازه گیری و به مسافت تبدیل می‌شود.

نحوه عملکرد pulsein

دستور pulseIn منتظر می‌ماند که pin، در وضعیت فعال قرار گیرد به عنوان مثال

```
pulseIn(echoPin,HIGH);
```

در دستور بالا آردواینو منتظر می‌ماند که پین Echo در وضعیت HIGH قرار گیرد وقتی یک پالس HIGH به پین می‌رسد، دستور pulseIn، مدت زمانی که پین در وضعیت LOW بوده تا ابتدای دریافت پالسHIGH را برمی‌گرداند.

نمونه کد نوشته شده برای اندازه گیری مسافت سنسور تا مانع در آردواینو:

```
int trigPin=3;
int echoPin=2;
long duration;
int distance;
int incomingByte = 0; // for incoming serial data
```

```
void setup() {

  pinMode(trigPin,OUTPUT);
  pinMode(echoPin,INPUT);
  Serial.begin(9600);
}
void loop() {
  Serial.print("Distance: ");
  Serial.println(getDistanceFromSensor());
}

int getDistanceFromSensor(){
  digitalWrite(trigPin,LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin,HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin,LOW);
  duration=pulseIn(echoPin,HIGH);
  distance=duration*0.034/2;
  return distance;
}
```

در این پروژه از از سه سنسور اولتراسونیک برای تشخیص موانع در جهات مختلف اسفاده شده. در صورت وقت کافی می‌توانید از ترکیب سنسور های نوری و IR و اولتراسونیک استفاده کنیده برای بالا بردن دقت هرچند باید به سایز ربات خود هم دقت کنید بعد سنسور مورد نظر را انتخاب کنید.

استفاده از سه سنسور برای اینکار اشکالاتی را ایجاد کرد و این حالت استفاده از سه سنسور اولتراسونیک برای ساخت ربات های حرفه ای اصلا توصیه نمی‌شود و این‌که با وجود سه سنسور اما باز هم قسمت هایی از فضا را ممکن است پوشش ندهد. اگر قرار است از این سنسور ها در بهینه ترین حالت استفاده کنید بهتر است از یک سروو موتور ۱۸۰ درجه استفاده کنید که در این حالت ما فقط نیاز به یک سنسور اولتراسونیک داریم و چرخش های ۱۸۰ درجه سروو موتور باعث می‌شود فضای بسیار بیشتری توسط سنسور پوشش داده شود و همچنین

دقت کار بالا رود، اما همیشه دقت، کاهش سرعت را به همراه دارد و با اینکار عملا سرعت را از دست می‌دهید ولی در عوض دقت را بدست می‌آورید این حالت برای ربات هایی که نیاز به سرعت دارند مانند ربات های میکرو ماوس و .. مناسب نیست و برای اینکار باید سنسور های مناسب به تعداد مورد نیاز برای پوشش کل فضا و همچنین دقت کافی استفاده شود.

۴.۴. انتخاب موتور

موتورها یکی از مکانیزم های اصلی حرمت ربات ها هستند. موتور ها انواع مختلفی دارند که هرکدام از آنها وظایف خاصی را انجام می‌دهند. بعضی از موتور ها را می‌توان به چرخ هایی متصل کرد که روبات را به اطراف حرکت دهد. بر اساس پروژه مورد نظر باید موتور خود را انتخاب کنیم که برای اینکار باید یکسری فاکتورها در انتخاب موتور را در نظر بگیریم که در ادامه به آن اشاره می‌کنیم.

عوامل مهم در انتخاب موتور :

۱. سرعت : سرعت همان حداکثر دور موتور است که معمولاً با دور در دقیقه یا RPM اندازه گیری می‌شود. ۱ دور در دقیقه به این معنی است که محور موتور هر دقیقه یک بار کاملاً به دور دایره می‌چرخد که بسیار کند است. حتی یک موتور DC بسیار ارزان نیز دارای سرعت حداقل ۱۰۰۰ RPM است.
۲. گشتاور : گشتاور اندازه گیری میزان قدرت خروجی موتور است. یعنی حداکثر وزنی که موتور می‌تواند بلند کند چقدر است.

به طور معمول ، هرچه دور موتور بیشتر باشد ، گشتاور آن کمتر است و بالعکس. بنابراین در طراحی و ساخت ربات، پیدا کردن موتورهایی با تعادل مناسب سرعت و گشتاور امری بسیار مهم است.

محرک های ربات

سه نوع محرک در ربات ها وجود دارد:

- **Pneumatic actuators**
- **Hydraulic actuators**
- **Electric actuators**

اکثریت قریب به اتفاق ربات ها از موتورهای الکتریکی استفاده می‌کنند که اغلب از موتورهای DC گیربکس دار و بدون جاروبک در روبات های قابل حمل یا موتورهای AC در ربات های صنعتی و ماشین های CNC استفاده می‌کنند. این موتورها اغلب در سیستم هایی با بارهای سبک تر ترجیح داده می‌شوند و شکل غالب حرکت چرخشی است.

که در اینجا به طور خاص روی موتور های الکتریکی تمرکز شده است.

موتورهای الکتریکی: این نوع موتور ها انرژی الکتریکی را به انرژی مکانیکی تبدیل می‌کنند. امروزه این نوع موتور ها رایج ترین نوع موتور در ساخت ربات ها است. آن ها همچنین در ربات هایی با اندازه‌های مختلف به خوبی عمل می‌کنند، نسبت به توان خروجی خود فشرده هستند و دارای آلودگی ناچیز هستند.

اگرچه موتور های الکتریکی رایج ترین نوع موتور در ساخت ربات ها هستند اما مشکلاتی نیز دارند نظیر این‌که سیستم فرمان پیچیده تری در مقایسه با سایر انواع موتورها نیاز دارند.

انواع موتور های الکتریکی

- موتورهای جریان متناوب [۲۳](AC): جریان متناوب در پریزهای دیواری رایج موجود است.موتورهای AC از این منبع انرژی برای تولید القای الکترومغناطیسی استفاده می‌کنند. مهندسان از مکانیسم‌های AC در موقعیت هایی که نیاز به سرعت ثابت دارند، استفاده می‌کنند. با این حال، این موتورها برای استفاده در صنعت رباتیک مناسب نیست و برای مقیاس های صنعتی بزرگ و با گشتاور بالا هستند.
- موتور های جریان مستقیم [۲۴](DC): معمولا این نوع موتورها با باتری تغذیه می‌شوند.مکانیسم‌های DC در طیف وسیعی از اندازه ها ظاهر می‌شوند و محدوده های بار بسیار متغیر، به علاوه زمان پاسخ و تحرک سریع در مقایسه با مدل هایی که از برق شهرتغذیه می‌شوند ارائه می‌کنند.
- سروو موتور [۲۵]ها: این موتورها به طور خلاصه "Servo" نامیده می‌شوند، از جمله سروووهای صنعتی، برخی از دقیق ترین موتور های موجود هستند.آنها در حین کار از تنظیمات خطای فوری پشتیبانی می‌کنند. اگر به موقعیت یابی دقیق به همراه گشتاور بالا در پکیج کوچک نیاز دارید، این موتورها برای بازوهای رباتیک و دیگر انواع ربات و کوبات [۲۶]عالی هستند. هنگام کار با سروووها ممکن است بین حرکات تاخیر وجود داشته باشد.
- موتور های پله ای [۲۷]یا استپر موتور ها : استپر موتور یا استپ موتور یا موتور پله ای یک موتور براشلس الکتریکی dc است که یک دور کامل ۳۶۰درجه را به تعدادی پله یا استپ مساوی تقسیم میکند. این موتور ها می‌توانند تا هنگام نگه داشتن یک جسم، گشتاور بالا و همچنین موقعیت‌یابی دقیق برای هر مرحله را فراهم کنند. دقت این موتورها در حدود ۰.۰۱ درجه و ۰.۱ میلی متر است. گشتاور کلی مشابه موتورهای

[23] Alternating current (AC) motors
[24] Direct current (DC) motors
[25] Servo motors
[26] cobot
[27] Stepper motors

سروو است. کارکرد موتورهای پله ای نسبت به انواع دیگر از نظر تامین انرژی الکتریکی مورد نیاز تا حدودی گران تر است.

چگونه موتور مناسب برای ربات خود انتخاب کنیم؟

برای انتخاب موتور برای پروژه های رباتیک بعدی خود می‌توانید نکات بالا را مرور کنید. با این حال قبل این‌که یکبار دیگر موارد بالا را بخوانید، لیستی از ویژگی هایی که در ربات خود نیاز دارید ایجاد کنید.

میتوانید به عنوان مثال این سوال ها را در لیست خود بپرسید.

ربات چه وظیفه‌ای را انجام خواهد داد، و میزان خطای شما برای حرکات آن چقدر باید باشد؟

محیط مورد نظر را از نزدیک مطالعه کنید تا درک لازم برای آلودگی نیز درک کنید، آلودگی موتور نه تنها عامل اصلی خرابی موتورها است، بلکه تهدیدی برای کیفیت محصول در محیط های تولیدی ریسک گریز هست.

نکات اصلی دیگر در انتخاب موتور که باید به آن دقت شود:

- **نسبت اندازه به توان:**[۲۸] در چه مقیاسی می‌سازید؟ آیا ربات نیاز به حرکت در محیط های باریک دارد یا قدرت موتور مهم تر از اندازه است؟
- **محدودیت های بار**[۲۹]**:** انواع درایوهایی که برای ربات خود انتخاب می‌کنید به گشتاور کافی برای مقابله با بارهای مورد انتظاری که حمل می‌کند نیاز دارند.
- **دقت:** وقتی ربات در حال حرکت هست، تحمل انحراف چقدر است؟ آیا به موتوری نیاز دارید که تصحیح خطا را در لحظه انجام دهد؟
- **قابلیت اطمینان**[۳۰]**:** بخش تعمیر و نگهداری شما چقدر پرسنل دارد؟ هرچند وقت یکبار میخواهید تعمیر و نگهداری ربات خود را انجام دهید؟
- **مصرف منابع:** موتور های پله ای در حالت آماده به کار نیز انرژی مصرف می‌کنند، اما گاهی اوقات آنها تنها انتخاب هستند.

خب راجع موتور های مختلف صحبت شد که بسته به پروژه مورد نظر باید انتخاب شوند چرا که بدون موتور ها عملا ربات هیچ است و باید در انتخاب موتور دقت لازم را بشود.

[28] Size to power ratio
[29] Load limits
[30] Reliability

اما برای این پروژه چون وقت بسیار محدود بود و باید بسرعت تصمیم گیری میشد از موتور های گیربکسی پلاستیکی استفاده شده است.

شکل ۴.۶–موتور گیربکسی پلاستیکی

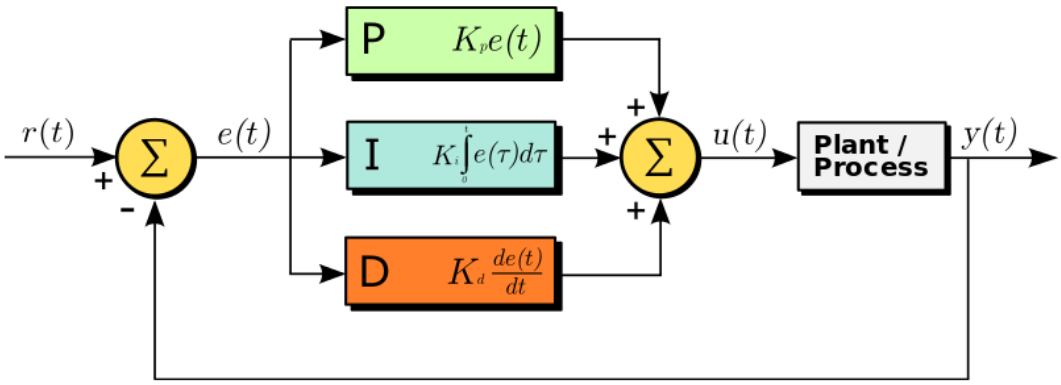
همانطور که در تصویر بالا می‌بینید موتور های گیربکسی پلاستیکی به این شکل هستند، اما متاسفانه این نوع موتور ها که در بازار هست از کیفیت بسیار پایینی برخوردار هستند و تراز کردن موتور ها برای این‌که با سرعت مشابه حرکت کنند کار دشواری است. چرا که این موتور ها رو برد به خوبی عمل نمی‌کنند و در پیاده سازی رباتتان به درستی عمل نمی‌کنند و انتظارات را برآورده نمی‌کند در این پروژه برای هر ربات از دو عدد ازین موتورهای گیربکسی استفاده شده اما روی برد متاسفانه یکی از موتور جریان بیشتری می‌کشد و تند تر می‌چرخد. که برای حل این موضوع می‌توان از ترانزیستور های قدرتی نظیر TIP42 بهمراه خازن عدسی استفاده شود تا جریان و ولتاژ تقریبا یکسان به موتور ها برسد و موتورهایتان بصورت مشابه حرکت کند. اما این موتور ها از دقت خوبی برخوردار نیستند و بعد از تجربه کردن پیشنهاد می‌شود از این نوع موتور ها برای ساخت چنین ربات هایی استفاده نشود تنها مزیت این نوع موتور ها قیمت آن است که ارزان قیمت هستند اما برای مصارف حرفه ای و دقیق مناسب نیستند. در صورت داشتن بودجه کافی از موتورهای باکیفیت تر DC یا استپر موتور ها یا سروو موتور ها با قابلیت تنظیم PWM استفاده کنید تا در اجرا پروژه به مشکل نخورید و ربات بدرستی تست ها را پشت سر بگذارد.

همچنین یکی دیگر از راهکارهایی که می‌توان برای حل این موضوع استفاده کرد، استفاده از PID کنترلر است. که در سیستم های کنترل خطی با این کنترل کننده آشنا شدیم.

PID کنترلر چیست؟

PID مخفف انتگرال [31]و مشتق [32]است. این نام از روش هایی در مورد نحوه برخورد چنین کنترل کننده ای با اغتشاش در سیستم گرفته شده است. با این حال، چنین کنترل کننده ای فقط در سیستم های دارای فیدبک وجود دارد.

سیستم فیدبک دار سیستمی است که بخشی از خروجی به ورودی فیدبک می‌شود.



شکل۴.۷- یک سیستم فیدبک با کنترل کننده PID

عملکرد این کنترل کننده به این صورت است که سیگنال یا سیگنال هایی را به عنوان خطا از ورودی دریافت می‌کند و سپس عملیاتی را روی آن انجام می‌دهد و در نهایت خروجی شان باهم جمع می‌شود. خروجی این مجموعه همان خروجی کنترل کننده PID است که برای اصلاح خطا به سیستم فیدبک داده می‌شود.

فرمول استاندارد PID به شکل زیر است:

$$y(t)= K_p \left(e(t) + \frac{1}{Ti}\int_0^t e(\tau)d\tau + Td\frac{de}{dt}\right) \quad (1)$$

با توجه به رابطه بالا تابع تبدیل به صورت زیر به‌دست می آید:

$$G_c= K_p + \frac{Ki}{S} + K_d s \quad (2)$$

[31] Integral

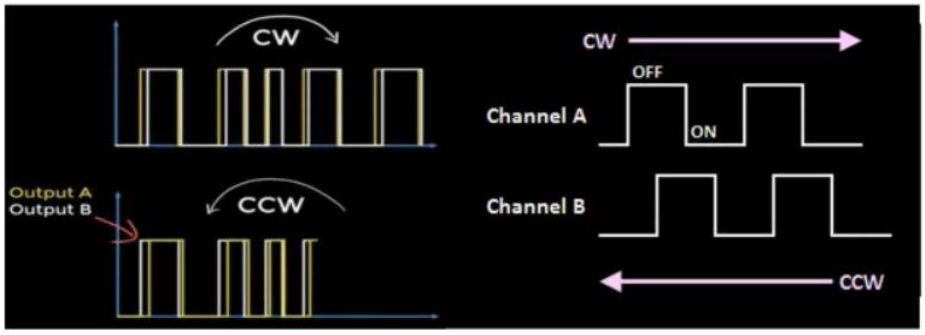
[32] Derivative

با استفاده از PID کنترلر می‌توان خطا را تا حد زیادی کاهش داد و سرعت موتور ها با تقریب زیادی مشابه شوند.

پیاده سازی PID کنترلر با استفاده از Arduino :

برای اینکه بتوان از PID کنترلر استفاد شود باید از موتور های دارای انکودر چرخشی [۳۳]یا رمزگذار چرخشی استفاده شود تا با استفاده از آن بتوان موقعیت شفت و در نهایت موقعیت چرخ های ربات را مشخص کرد.

کاری که باید انجام داد این است که هر زمان که چرخ از موقعیت خود خارج شد، موتور بچرخد، علاوه بر این موتور از طریق مدولاسیون عرض پالس، یا همان PWM کنترل می‌شود هرچه عرض پالس بیشتر باشد موتور تندتر می‌چرخد.



شکل۴.۸ – کنترل موتور ها از طریق PWM با PID کنترلر

کد نوشته شده در نرم افزار Arduino برای پیاده سازی PID کنترلر:

```
#include <PID_v1.h>

#define PIN_INPUT 0
#define PIN_OUTPUT 3

//Define Variables we'll be connecting to
double Setpoint, Input, Output;

//Specify the links and initial tuning parameters
double Kp=2, Ki=5, Kd=1;
```

[33] rotary encoder

```
PID myPID(&Input, &Output, &Setpoint, Kp, Ki, Kd, DIRECT);

void setup()
{
 //initialize the variables we're linked to
 Input = analogRead(PIN_INPUT);
 Setpoint = 100;

 //turn the PID on
 myPID.SetMode(AUTOMATIC);
}

void loop()
{
 Input = analogRead(PIN_INPUT);
 myPID.Compute();
 analogWrite(PIN_OUTPUT, Output);
}
```

۴.۵. درایور موتور

یک کنترل کننده موتور یک دستگاه الکترونیکی است (معمولا به شکل یه مدار بدون پوشش و محفظه است) که به عنوان یک دستگاه واسطه بین میکروکنترلر، یک منبع تغذیه یا باتری و موتورها عمل می‌کند.

اگرچه میکروکنترلر (مغز ربات) سرعت و جهت موتور را مشخص میکند، اما به دلیل محدودیت زیاد در تغذیه خروجی (جریان و ولتاژ) نمی‌تواند آن ها را مستقیما هدایت کند. از طرف دیگر درایور موتور میتواند جریان را در ولتاژ مورد نظر فراهم کند اما نمیتواند تصمیم بگیرد که موتور تا چه میزان سریع بچرخد.

بنابراین، میکروکنترلر و کنترل کننده موتور باید باهم کار کنند تا موتور به طور مناسبی حرکت کند. معمولا میکروکنترلر میتواند از طریق یک روش ارتباطی ساده مانند UART(serial) یا PWMبه کنترل کننده موتور دستوالعملی برای چگونگی تغذیه موتور بدهد. هم چنین برخی از کنترل کننده های موتور را می‌توان به صورت دستی و با استفاده از ولتاژ آنالوگ کنترل کرد (معمولا با یک پتانسیومتر ایجاد می‌شود).

اندازه و وزن فیزیکی کنترل کننده موتور می‌تواند بسیار متفاوت باشد، از یک دستگاه کوچکتر از نوک انگشتتان برای کنترل یک ربات کوچک sumo تا یک کنترل کننده سنگین وزن چند کلیوگرمی. اندازه و وزن کنترل کننده ربات معمولا کمینه تاثیر را روی ربات دارد، تا زمانی که شما با ربات‌های خیلی کوچک و یا هواپیماهای بدون سرنشین کار می‌کنید که در این حالت کوچکترین وزن‌ها هم تاثیرگذار خواهد بود. اندازه کنترل کننده موتور معمولا به حداکثر جریانی که میتواند فراهم کند وابسته است. جریان بیشتر به معنی استفاده از سیم هایی با قطر بزرگتر است.

درایور موتور Motor Driver از مدارهای مهم در رباتیک و پروژه‌های مختلف می‌باشد. دسته بندی **درایور موتور** بر اساس نوع تراشه و میزان ولتاژ و به خصوص جریان عبوری از درایور اهمیت دارد. انواع **درایور موتور Motor Driver**برای فرمان به موتور DC موتور Stepper و یا سرو موتور Servo Motor Driver را بررسی کنید. از آنجایی که راه اندازی موتورهای مختلف نیاز به تامین جریان و ولتاژ مجزا دارد، برای مدیریت موتور Motor هم نیاز به تراشه و مدار **کنترل موتور Motor Driver** می‌باشد. همانطور که توضیح داده شد، بر اساس جریان عبوری و ولتاژ قابل عبور توسط ماژول دسته بندی می‌شوند.

درایور موتور Motor Driver یا راه انداز موتور برای به حرکت در آوردن موتور به کار می‌روند. این نوع مدارها به ازای اطلاعات دریافتی از میکرو و یا کنترل کننده، ولتاژ مورد نیاز موتور را تامین می‌کنند.

درایورهای مختلف موتور برای ربات های خود ران:

همانطور که پیش‌تر اشاره شد ربات های خودران توسط یک میکروکنترلر کنترل می‌شوند. از درایور های Fireblade و Viper Motor می توان برای ارتباط میکرو با موتور استفاده کرد. به غیر از این درایو ها چند درایور موتور با جریان پایین نظیر L298N ، L9110 ، L293D و... وجود دارد.

L293D در شیلدهای مختلف مانند Auton Shield, Xbee Shield, Starter Shield استفاده می‌شوند. چراکه در موتورهای جریان پایین که معمولا در ربات های خودران کوچک استفاده می‌شود بهترین است.

L298N یک درایور دو موتوره است که می‌تواند به شما کمک کند تا موتور هایی که حداکثر ۲ آمپر جریان دارند مانند موتور های BO ، موتور های گیربکسی متصل کند.

برای موتور های رابطی که جریان بالا می‌کشند، می‌توان سری , Fireblade series – Fireblade 10A Fireblade 30A (بدون کنترل سرعت) یا درایور موتور وایپر[۳۴] (کنترل کننده سرعت) با توجه به کاربرد مورد نظر انتخاب کرد.

در این پروژه به طور خاص دو ماژول درایور L298N و L9110 راه اندازی می‌شود.

۴.۶-ماژول درایور L298N و راه اندازی آن :

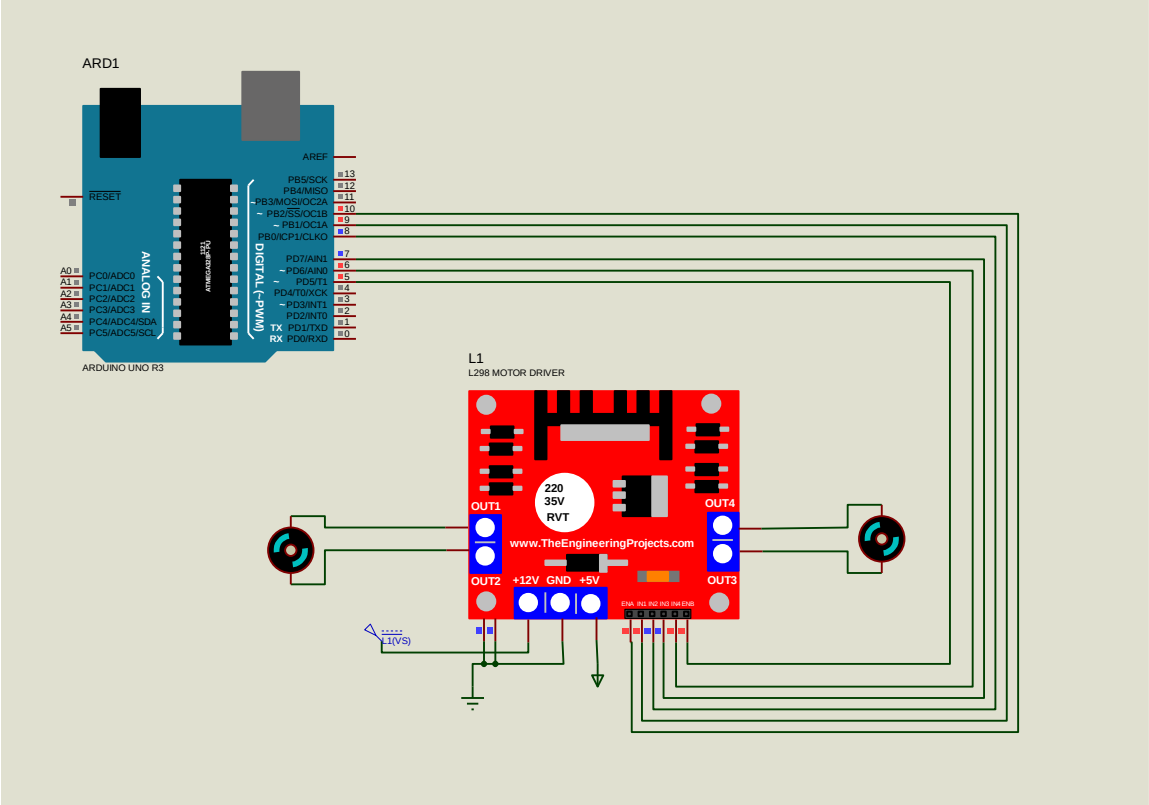
این درایور یکی از ارزان ترین و کاربردی ترین ماژول های کنترل موتور در بازار می باشد که

می توان با آن دو موتورdc و بعضی از استپ موتور ها را با این درایور راه‌اندازی کرد.

شکل ۴.۹-ماژول درایور l298N

[34] Viper motor driver

راه‌اندازی درایور L298N با آردوینو



شکل ۴.۱۰- مدار راه‌انداز موتورهای dc به‌وسیله ماژول Lm298N و آردوینو در نرم‌افزار پروتئوس

همانطور که در تصویر بالا مشاهده می‌شود برای راه اندازی موتور های dc با استفاده از این ماژول باید به‌این‌صورت عمل شود.

موتور ها باید به پایه های Out1 و Out2 و یا Out3 و Out4 متصل شوند. پایه ۱۲ ولت ماژول باید به سر مثبت باتری متصل شود و پایه Gnd باید به سر منفی باتری و آردوینو و پایه ۵ ولت ماژول به ۵ ولت آردوینو متصل شود.

سپس پایه های ورودی ماژول به پایه های دیجیتال آردوینو متصل شود و اینکه برای کنترل سرعت موتور باید پایه های En ماژول به پایه های pwm آردوینو که با علامت ` ~ ` مشخص شده است متصل شود تا به توان از طریق دستور anlogWrite (en , pwm) آردوینو می‌توان سرعت موتور ها را کنترل کرد.

کد نوشته شده برای راه اندازی ماژول و موتور های که در شکل ۴.۸ نشان داده شده است:

```
#define MA1 6  // Motor A pins
#define MA2 7
```

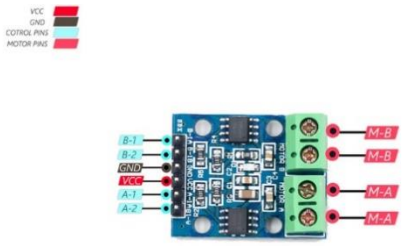
```
#define MB1 8 // Motor B pins
#define MB2 9
#define EnB 10
#define EnA 5
#define PWM_RIGHT 200
#define PWM_LEFT 200
void setup() {
  // put your setup code here, to run once:
  pinMode(MA1, OUTPUT);
  pinMode(MA2, OUTPUT);
  pinMode(MB1, OUTPUT);
  pinMode(MB2, OUTPUT);
  pinMode(EnA, OUTPUT);
  pinMode(EnB, OUTPUT);

}
void loop() {
  forward();
  delay(1000);
  Stop();
}
void  forward() {          //function of forward
  digitalWrite(MA1, HIGH);
  digitalWrite(MA2, LOW);
  digitalWrite(MB1, LOW);
  digitalWrite(MB2, HIGH);
  analogWrite(EnA, PWM_LEFT);
  analogWrite(EnB, PWM_RIGHT);
}

void backward() {         //function of backward
  digitalWrite (MA1, HIGH);
```

```
  digitalWrite (MA2, 0);
  digitalWrite (MB1, HIGH);
  digitalWrite(MB2, 0);
  analogWrite(EnA, PWM_LEFT);
  analogWrite(EnB, PWM_RIGHT);

}
void Stop() {              //function of stop
  digitalWrite(MA1, LOW);
  digitalWrite(MA2, LOW);
  digitalWrite(MB1, LOW);
  digitalWrite(MB2, LOW);
  analogWrite(EnA, PWM_LEFT);
  analogWrite(EnB, PWM_RIGHT);
}
void turnLeft() {          //function of turn left
  digitalWrite(MA1, HIGH);
  digitalWrite(MA2, LOW);
  digitalWrite(MB1, LOW);
  digitalWrite(MB2, HIGH);
  analogWrite(EnA, PWM_LEFT);
  analogWrite(EnB, PWM_RIGHT);
}
void turnRight() {         //function of turn right
  digitalWrite(MA1, LOW);
  digitalWrite(MA2, HIGH);
  digitalWrite(MB1, HIGH);
  digitalWrite(MB2, LOW);
  analogWrite(EnA, PWM_LEFT);
  analogWrite(EnB, PWM_RIGHT);
}
```

۴.۷-ماژول درایور L9110S و راه اندازی آن :

این ماژول یک ماژول درایور دوکاناله جمع وجور و ارزان قیمت برای راه اندازی موتور ربات های کوچک است. این ماژول دارای دو تراشه دارایور موتور مستقل است که هرکدام از آن‌ها می‌توانند تا ۸۰۰ میلی آمپر جریان را افزایش دهند. ولتاژ کاری این ماژول بین ۲.۵ تا ۱۲ ولت است، که این ماژول را قادر می‌سازد با هر میکروکنترلر ۳.۳ و ۵ ولت استفاده شود.

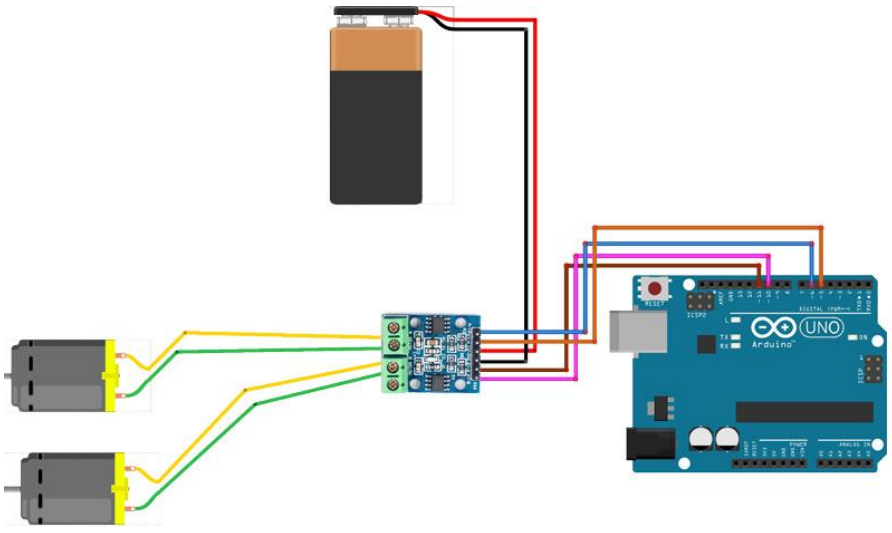
موتورها از طریق دو ترمینال پیچی به ماژول متصل می‌شوند. سیگنال مدولاسیون عرض پالس PWM برای کنترل سرعت موتور و یک خروجی دیجیتال برای تغییر جهت استفاده می‌شود. ماژول همچنین می تواند برای راه اندازی استپر موتورهای ۴ سیمه ۲ فاز نیز استفاده شود. نصب آن روی ربات یا پروژه های دیگر به آسانی انجام می‌شود.



شکل ۴.۱۱ – ماژول درایور L9110S

راه اندازی ماژول درایور L9110S با آردوینو:

راه اندازی این ماژول درایو بسیار شبیه به ماژول L298N است با این تفاوت که این ماژول پایه En ندارد.



شکل ۴.۱۲ – مدار راه انداز ۲ موتور DC به وسیله درایور L9110S با آردوینو

کد نوشته شده برای راه اندازی ماژول و موتور های که در شکل ۴.۱۱ نشان داده شده است:

```
#define MA1 ۵  // Motor A pins
#define MA2 ۶
#define MB1 ۱۰ // Motor B pins
#define MB2 ۱۱
#define PWM_RIGHT 200
#define PWM_LEFT 200
void setup() {
  pinMode(MA1, OUTPUT);
  pinMode(MA2, OUTPUT);
  pinMode(MB1, OUTPUT);
  pinMode(MB2, OUTPUT);
}
void loop() {
  forward();
  delay(1000);
  turnLeft();
  delay(1000);
  turnRight();
  delay(1000);
  Stop();
}
void  forward() {          //function of forward
  analogWrite(MA1, PWM_RIGHT);
  digitalWrite(MA2, LOW);
  digitalWrite(MB1, LOW);
  analogWrite(MB2, PWM_Left);
}


void backward() {         //function of backward
```

```
  analogWrite(MA1, PWM_RIGHT);
  digitalWrite (MA2, 0);
  analogWrite(MB1, PWM_RIGHT);
  digitalWrite(MB2, 0);
}
void Stop() {              //function of stop
  digitalWrite(MA1, LOW);
  digitalWrite(MA2, LOW);
  digitalWrite(MB1, LOW);
  digitalWrite(MB2, LOW);
}
void turnLeft() {         //function of turn left
  analogWrite(MA1, PWM_RIGHT);
  digitalWrite(MA2, LOW);
  digitalWrite(MB1, LOW);
  analogWrite(MB2, PWM_LEFT);
}
void turnRight() {         //function of turn right
  digitalWrite(MA1, LOW);
  analogWrite(MA2, PWM_RIGHT);
  analogWrite(MB1, PWM_LEFT);
  digitalWrite(MB2, LOW);
}
```

۴.۹ – انتخاب منبع تغذیه

منبع تغذیه ربات های کوچک را باتری ها تشکیل می دهند. باتری ها انواع مختلفی دارند که باید براساس پروژه موزد نظر انتخاب شوند. یکیاز نکاتی که باید در انتخاب باتری ها دقت شود، قابل شارژ بودن یا نبودن آن است. برای صرفه جویی در هزینه انتخاب باتری قابل شارژ می‌تواند بهترین گزینه باشد، درست است در ابتدا هزینه بیشتری پرداخت خواهید کرد اما با چند بار شارژ کردن این باتری هزینه ها جبران خواهد شد. یکی دیگر از مواردی که اهمیت دارد، اندازه است. به طور کلی باتری‌ها با سلول های بزرگ تر عمر طولانی تری دارند. ولتاژ های بالاتر به سلول های بیشتری نیاز دارد. در حالی که ممکن است پردازنده شما احتمالا فقط به منبع تغذیه ۵

ولت در چند ۱۰ میلی آمپر نیاز داشته باشد درحالی که موتور ها و سنسور ها به ولتاژ های بالا و ظرفیت زیاد نیاز دارند.

به عنوان مثال یک ربات میکروماوس با موتور پله‌ای[۳۵]برای اینکه بتواند بهترین عملکرد دینامیکی خود را داشته باشد، نیاز به یک یا چند جفت سلول برای دادن ولتاژ ۱۵ یا بیشتر داشته باشد، هنگامی که استپرها کاملا انرژی دارند. ممکن است ۲ آمپر بکشند ربات شما باید حداقل ۱۵ دقیقه در حالت آماده به کار بماند. که در این شرایط مدیریت منابع تغذیه مهم خواهد شد. موتورها باید در صورت عدم نیاز خاموش شوند، سنسور ها باید برای مدت زمان کمتری پالس شوند و از بیش از حد چشمک زدن چراغ ها جلوگیری شود. این یک پیکره‌بندی آسان برای مدیریت بهتر منبع تغذیه بود. به‌عنوان مثال افت ۸ ولت در ۲۰۰ میلی آمپر رگولاتورها را بسیار داغ می‌کند که برای جلوگیری از داغ شدن رگولاتور ها باید از هیت سینک استفاده شود.

پارامترهای عملکرد باتری

ولتاژ پایانه :

ولتاژ بین پایانه‌های باتری را ولتاژ پایانه می‌نامند که با ولت اندازه گیری می‌شود. معمولا یک سلول، ولتاژی از ۱ تا ۲ ولت دارد (اگر دانشمندان از گاز فلورین با پتانسیل کاهش ۲.۸۷ و فلز لیتیوم با -۳.۰۵ برای گرفتن ۵.۹۲ ولت استفاده کنند، چه می‌شود؟ فقط یک خیال پردازی ...) . با **اتصال چندین باتری (در واقع سلول) به صورت سری، ولتاژ بالاتر**ی را می‌توان به دست آورد .

ولتاژ مدار باز :

وقتی باتری نه در حال **شارژ** است و نه خالی شدن، در این هنگام ولتاژ پایانه به عنوان ولتاژ مدار باز شناخته می‌شود .

منحنی ولتاژ :

منحنی ولتاژ یک باتری افت ولتاژ پیشرونده را هنگام تخلیه نشان می‌دهد .

منحنی تخلیه :

باتری‌ها معمولا تمایل دارند هنگام کار از لحاظ ولتاژ افت کنند. تعداد کمی از باتری‌ها ولتاژ اولیه خود را تا زمان تخلیه کامل حفظ می‌کنند. این تخلیه در ولتاژ به صورت نموداری در برابر زمان نشان داده می‌شود. هر چه منحنی صاف‌تر باشد، باتری بهتر است؛ اکثر باتری‌های قدیمی دارای تخلیه شیب‌دار بودند و باتری‌های جدیدتر

[35] Stopper Motor

البته صاحب برند معتبر، منحنی تخلیه صاف دارند. خیلی بعید است بتوانید برای یک باتری قلمی عادی که از بازار می خرید، خصوصاً در ایران انتظار دریافت منحنی باتری داشته باشید. چرا که اغلب فروشندگان قطعات تخصصی متاسفانه و با ابراز ناراحتی زیاد از این مساله، سواد و تخصص کافی در ارائه اطلاعات فنی محصول خود را ندارند! غیر از این مورد تولید کننده هم ممکن است برای یک باتری با کاربرد ساده مثلاً اسباب بازی و مشابه آن منحنی ارائه ندهد. اما در لوازم حساس تر مثلاً یک ربات پرنده، یک دستگاه نظامی خرید باتری خیلی شوخی به نظر نمی رسد .

ظرفیت ذخیره سازی :

مقدار جریانی است که باتری می‌تواند در واحد زمان تأمین کند که با آمپر-ساعت اندازه‌گیری می‌شود (برای باتری‌ها معمولاً **میلی آمپر ساعت** است). مثلا اگر باتری ۲۰۰۰ میلی آمپر ساعت است، یعنی باتری می‌تواند ۲ آمپر یا ۲۰۰۰ میلی آمپر جریان را برای مدت یک ساعت تأمین کند. اگر ربات فقط ۱۰۰۰ میلی آمپر جریان مصرف می‌کند، باتری حدودا ۲ ساعت کار می‌کند. حالا دیگر باید تفاوت بین mA و mAh این است که در مقایسه باتری ها با یکدیگر آمپر ساعت از آمپر تنها خیلی مهم تر است. معمولاً روی تمام باتری‌ها عدد **میلی آمپر** نوشته شده. با **اتصال موازی باتری‌ها** (در واقع سلول‌ها) می‌توان **جریان خروجی بالاتر** را بدست آورد .

C-rate :

این مورد برای یک طراح ربات خیلی مهم نیست، اما خُب C-rate ... میزان **شارژ و دشارژ باتری** است که با توجه به **ظرفیت ذخیره سازی** آن که معمولا mAh یا Ah است، بیان می‌شود.۱C به معنای تخلیه کل انرژی ذخیره شده در ۱ ساعت است و۰.۵C به معنای تخلیه کل انرژی در ۲ ساعت است. مثلاً، یک باتری با ۱.۵ میلی آمپر ساعت، در صورت تخلیه با ۱C، جریان ۱۵۰۰ میلی آمپر، برای یک ساعت می‌دهد. اگر همان باتری ۰.۵C باشد، ۷۵۰ میلی آمپر در ۲ ساعت تخلیه می‌شود. معمولا اکثر باتری‌ها ۱C هستند .

توان :

مقدار توان باتری در واحد حجم است که با وات بر متر مکعب اندازه گیری می‌شود .

تعداد دوره :

به تعداد دفعاتی گفته می‌شود که باتری شارژ و خالی (قابل استفاده برای **باتری‌های شارژشونده**)می‌شود، پیش از اینکه عملکرد به زیر حد انتظار برسد .

ماندگاری :

مدت زمانی که باتری می‌تواند بدون استفاده در قفسه یا فروشگاه سالم بماند .

طول عمر :

مدت زمان تا پیش از **افت عملکرد باتری**؛ چه استفاده شده چه استفاده نشده .

دما :

عملکرد اکثر باتری‌ها با تغییر دما کاهش می‌یابد. بهتر است که باتری انتخاب کنید که با کمی تغییر دما عملکرد آن افت نکند .

انواع باتری ها

- لیتیم
- نیکل ـ کادمیم
- باتریهای هیبرید نیکل ـ فلز[36]
- باتریهای لیتیم ـ یون[37]
- ...

برای اینکه ربات عملکرد بهتری داشته باشد و موتور ها و سنسور ها در بهترین عملکرد خود قرار داشته باشند بهتر است منبع تغذیه میکروکنترولر را از سنسورها و موتور ها مستقل شود.

دراین پروژه برای میکرو از باتری کتابی ۹V و منبع تغذیه موتور ها وسنسور ها نیز از باتری لیتیوم یونی ۱۸۶۵۰ استفاده شده است.

از مزایای باتری لیتیوم یونی میتوان به موارد زیر اشاره کرد:

- باتری های لیتیوم یونی عمر طولانی دارند
- قابل شارژ می‌باشند
- چگالی انرژی بالایی دارند
- و ...

[36] nickel-metal hybride
[37] Lithium-Ion

در کنار مزیت هایی که دارند معایبی نیز دارا هستند از جمله این‌که این باتری ها در مقایسه با سایر باتری ها:

- از وزن بالایی برخوردارند
- فضای زیادی را اشغال می‌کنند
- و همچنین تخلیه شارژ نسبتا بالایی دارند.

خب تا اینجا با وسایلی که می‌توان یک ربات میکروماوس و ربات حل ماز هوشمند با ویژگی هایی که در پروژه تعریف شده ساخت و نحوه راه‌اندازی آن ها صحبت شد.

در بخش بعدی راجع به پیاده سازی الگوریتم و پیاده سازی نهایی ربات صحبت خواهد شد.

فصل پنجم

پیاده سازی در عمل

۵.۱. ساخت مسیر

خب در بخش های قبل راجع‌به کلیات پروژه و شیوه انجام آن به صورت تئوری صحبت شد، حالا در این قسمت یعنی بخش آخر راجع پیاده سازی عملی مطالب گفته شده در بخش های قبل پرداخته می شود.

خب حالا باید ماز طراحی شده را واقعیت پیاده سازی شود. شکل ۵.۱ پیاده سازی ماز انتخابی در محیط واقعی می باشد.

شکل ۵.۱ – پیاده سازی ماز طراحی شده

۵.۲. ساخت ربات

در قسمت های قبل راجع به ابزارهایی که می‌شود یک ربات میکروماوس یا ربات حل ماز را ساخت صحبت شد.

در این قسمت به صورت عملی ربات حل مسئله ماز برای این پروژه طراحی و ساخته می‌شود.

وسایل مورد نیاز:

- موتور گیربکسی ۱۰۰RPM
- باتری لیتیوم یونی ۱۸۶۵۰
- باتری کتابی ۹V
- ماژول درایور L9110S
- پلتفرم آردوینو
- ماژول اوولتراسونیک
- ماژول وایرلس nRF24L01

شکل ۶.۱- پیاده سازی نهایی ربات در محیط واقعی

۵.۳. الگوریتم نوشته شده در نرم افزار Arduino

چون در پروژه هدف ارسال مسیر صحیح به ربات همکار است، بنابراین از الگوریتم هایی که نمی‌تواند مسیر را رهگیری کند نمی‌توان استاده کرد الگوریتم هایی نظیر دنبال کردن دیواره‌ها الگوریتم های تصادفی و...

همانطور که در قسمت های قبل گفته شد، ربات باید بتواند مسیر را رهگیری کند و در پشته [۳۸]ذخیره کند تا در انتهای مسیر ماز یعنی وقتی‌که به هدف رسید بتواند مسیر ذخیره شده در پشته را به ربات همکار ارسال کند.

برای همین برای این پروژه از ترکیب الگوریتم پیمایش گراف ها به روش BFS و الگوریتم FLOOD FILL استفاده شده است.

کد نوشته شده در نرم‌افزار Arduino

کد نوشته شده در پیوست ۱ آورده شده است.

۵.۴. نتایج بدست آمده

الگوریتم در نرم افزارArduino نوشته شده و از طریق شبیه ساز و ترمینال نرم افزار تست شده و نتایج بدست آمده با نتایج مورد انتظار پروژه همخوانی داشته است.

[38] Stack

پیوست ۱: الگوریتم نوشته شده در نرم افزار Arduino

کد مربوط به ربات اصلی[۳۹]:

```
/**
 *  Main Robot
 *  Cooperation of Multiple Robots to Solve Maze Tasks
 *  The circuit:
 *    1.Arduino Uno
 *    3.Dc Motor 100RPM
 *    2.L9110S
 *    3.Ultrasonic HC-SR04
 *    4.Nrf24l01
 *  Created At 20/12/2021
 *  By Amirhossein Rahmani & Mohsen Rahimi
 */
/*Initilize WIFI Madule*/
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>
#include <printf.h>

RF24 radio(9, 8); // (CE, CSN) for ARD Nano
#define MA1 5  // Motor A pins
#define MA2 6
#define MB1 10 // Motor B pins
#define MB2 11

#define PWM_RIGHT 200
#define PWM_LEFT 200
#define DST 11
```

[39] Main Robot

```
/**
*initialize all ultrasonics
*/
int trigPinLeft=A5;
int echoPinLeft=A4;
int trigPinRight=A1;
int echoPinRight=A0;
int trigPinFront=A3;
int echoPinFront=A2;
//
long duration;
int  distance;
int cell=-1;
bool isFindCorrectPath=false;
bool deadend=false;
int send_count=0;
uint32_t path[36],two_way[36]={0,0,0,0,0,0,
                               0,0,0,0,0,0,
                               0,0,0,0,0,0,
                               0,0,0,0,0,0,
                               0,0,0,0,0,0,
                               0,0,0,0,0,0};
/**
 * All modes that the robot must choose between that
 * mode : null,A:RL(Right and Left open),B:FR(Front and Right
open),C:LF(Front and Left open)
 * curr_dir: `F`:Forward,`R`:turn Right,`L`:turn Left
 */


char mode[36],curr_dir[36]={'\0','\0','\0','\0','\0','\0',
```

```
                              '\0','\0','\0','\0','\0','\0',
                              '\0','\0','\0','\0','\0','\0',
                              '\0','\0','\0','\0','\0','\0',
                              '\0','\0','\0','\0','\0','\0',
                              '\0','\0','\0','\0','\0','\0'};

const byte address[6] = {'R','x','A','A','A'};
uint32_t correct_path[36];
void setup() {
  Serial.begin(9600); // opens serial port, sets data rate to 9600 bps
  pinMode(trigPinFront,OUTPUT);
  pinMode(echoPinFront,INPUT);
  pinMode(trigPinRight,OUTPUT);
  pinMode(echoPinRight,INPUT);
  pinMode(trigPinLeft,OUTPUT);
  pinMode(echoPinLeft,INPUT);

  pinMode(MA1, OUTPUT);
  pinMode(MA2, OUTPUT);
  pinMode(MB1, OUTPUT);
  pinMode(MB2, OUTPUT);

  analogWrite(MA1, LOW);
  analogWrite(MA2, LOW);
  analogWrite(MB1, LOW);
  analogWrite(MB2, LOW);

  //setup wifi
  radio.begin();
  radio.openWritingPipe(address);
  radio.setPALevel(RF24_PA_MAX);
  radio.stopListening();
```

```
}
void loop() {
  Stop();
  delay(5000);

  //just front open
  if(isLeftWall() && isRightWall() && !isFrontWall()){
    forward();
    path[++cell]=1;
    Serial.println("forward first");
    delay(100);
    Stop();
  }

  //just right open
  if(isLeftWall() && !isRightWall() && isFrontWall()){
    path[++cell]=2;
    turnRight();
    delay(50);
    forward();
    delay(100);
    Serial.println("right");
  }

  //just left open
  if(!isLeftWall() && isRightWall() && isFrontWall()){
    path[++cell]=3;
    turnLeft();
    delay(50);
    forward();
    delay(100);
    Serial.println("left");
```

```
}

//left and right open
if(!isLeftWall() && !isRightWall() && isFrontWall()){
  int rand_=random(1,3);
  //choose random way between left and right
    if(rand_==1){
      turnRight();
      delay(50);
      forward();
      delay(100);
      path[++cell]=2;
      curr_dir[cell]='R';
      Serial.println("right 1");
    }
    if(rand_==2){
      turnLeft();
      delay(50);
      forward();
      delay(100);
      path[++cell]=3;
      curr_dir[cell]='L';
      Serial.println("left..");
    }
     two_way[cell]=1;
     mode[cell]='A';//LR
     Stop();
     delay(70);
}

//left and front open
if(!isLeftWall() && isRightWall() && !isFrontWall()){
```

```
 int rand_=random(1,3);
  //choose random way between left and front
  if(rand_==1){
      forward();
      delay(100);
      path[++cell]=1;
      curr_dir[cell]='F';
       Serial.println("forward..");
    }
    if(rand_==2){
      turnLeft();
      delay(50);
      forward();
      delay(100);
      path[++cell]=3;
      curr_dir[cell]='L';
       Serial.println("left..");
    }
    two_way[cell]=1;
    mode[cell]='C';//LF
    Stop();
    delay(70);
}

//right and front open
if(isLeftWall() && !isRightWall() && !isFrontWall()){
 int rand_=random(1,3);
  //choose random way between right and front
  if(rand_==1){
      forward();
      delay(100);
      path[++cell]=1;
```

```
      curr_dir[cell]='F';
      Serial.println("forward..");
    }
    if(rand_==2){
      turnRight();
      delay(50);
      forward();
      delay(100);
      path[++cell]=2;
      curr_dir[cell]='R';
      Serial.println("right 2");
    }
    two_way[cell]=1;
    mode[cell]='B';//FR
    Stop();
    delay(70);
}

//Deadend left and right and front wall
if(isDeadend()){
  Serial.println("Deadend");
  turnRight();
  //turn back 180deg
  delay(2000);
  Serial.println("turn back 180deg");
  Stop();
  deadend=true;
}
//turn back if is deadend
if(deadend){
  turn_back();
  delay(1000);
```

```
  }
  //right and left and front open all dir open
  if(isAllDirOpen()){
    Serial.println("all dir open");
    Serial.println("forward");
    forward();
    delay(4000);
    Stop();
    path[++cell]=1;
    if(isAllDirOpen()){
      Stop();
      delay(2000);
      Serial.println("End maze..");
      isFindCorrectPath=true;
      //End maze and send path to another robot
      copy(path,correct_path,36);
    }
  }

/**
 * after solve maze and find correct path send correct path to Fellow robot
 */
  if(isFindCorrectPath){
    if(send_count==0){
      int result=send_correct_path_to_fellow_robot();
      if(result){
        send_count=1;
      }
    }else{
       Serial.println("Already Sended.");
    }
  }
```

```
}

void turn_back(){
  for(;two_way[cell]!=1;cell--){
    if(path[cell]==1){
      forward();
      Serial.println("turn back :forward");
      delay(100);
      Stop();
    }else if(path[cell]==2){
      turnLeft();
      delay(50);
      forward();
      delay(100);
      Serial.println("turn back: left");
    }else if(path[cell]==3){
      turnRight();
      delay(50);
      forward();
      delay(100);
      Serial.println("turn back: right");
    }
    path[cell]=0;
    delay(5000);
    Stop();
}
   //two way
   if(two_way[cell]==1){
      if(mode[cell]=='A' && curr_dir[cell]=='L'){
      forward();
      Serial.println("two turn back: forward");
```

```
  delay(100);
  Stop();
  path[cell]=2;
}
if(mode[cell]=='A' && curr_dir[cell]=='R'){
  forward();
  Serial.println("two turn back: forward");
  delay(100);
  Stop();
  path[cell]=3;
}
//left and front open
if(mode[cell]=='C' && curr_dir[cell]=='F'){
  turnRight();
  delay(50);
  forward();
  delay(100);
  path[cell]=3;
  Serial.println("two turn back : right");
}
if(mode[cell]=='C' && curr_dir[cell]=='L'){
  turnLeft();
  delay(50);
  forward();
  delay(100);
  Serial.println("turn back: left");
  path[cell]=1;
}
//right and front open
if(mode[cell]=='B' && curr_dir[cell]=='F'){
  turnLeft();
  delay(50);
```

```
      forward();
      delay(100);
      Serial.println("two turn back: left");
      path[cell]=2;
    }
    if(mode[cell]=='B' && curr_dir[cell]=='R'){
      turnRight();
      delay(50);
      forward();
      delay(100);
      Serial.println("two turn back: right");
      path[cell]=1;
    }
   }
   Stop();
   delay(180);
   deadend=false;
   exit;
}


void  forward() {          //function of forward
  analogWrite(MA1, LOW);
  analogWrite(MA2, PWM_LEFT);
  analogWrite(MB1, LOW);
  analogWrite(MB2, PWM_RIGHT);
}

void backward() {         //function of backward
  analogWrite(MA1, HIGH);
  analogWrite(MA2, 0);
  analogWrite(MB1, HIGH);
```

```
  analogWrite(MB2, 0);
}

void Stop() {              //function of stop
  analogWrite(MA1, LOW);
  analogWrite(MA2, LOW);
  analogWrite(MB1, LOW);
  analogWrite(MB2, LOW);
}

void turnLeft() {         //function of turn left
  analogWrite(MA1, PWM_LEFT);
  analogWrite(MA2, LOW);
  analogWrite(MB1, LOW);
  analogWrite(MB2, PWM_RIGHT);
}

void turnRight() {         //function of turn right
  analogWrite(MA1, LOW);
  analogWrite(MA2, PWM_LEFT);
  analogWrite(MB1, PWM_RIGHT);
  analogWrite(MB2, LOW);
}

/**
 * get distance(cm) by ultarsonic sensor
 */
int getDuration(int trigPin,int echoPin){
  digitalWrite(trigPin,LOW);
  delayMicroseconds(5);
  digitalWrite(echoPin,LOW);
  digitalWrite(trigPin,HIGH);
```

```
  delayMicroseconds(10);
  digitalWrite(trigPin,LOW);
  duration=pulseIn(echoPin,HIGH);
  return duration;
}

int rightSensor(){
  long duration=getDuration(trigPinRight,echoPinRight);
  return (duration*0.034/2);
}
int leftSensor(){
  long duration=getDuration(trigPinLeft,echoPinLeft);
  return (duration*0.034/2);
}
int frontSensor(){
  long duration=getDuration(trigPinFront,echoPinFront);
  return (duration/40);
}
bool isLeftWall(){
  if(leftSensor()<DST){
     return true;
  }else{
    return false;
  }
}

bool isRightWall(){
  if(rightSensor()<DST){
    return true;
  }
  return false;
}
```

```
bool isFrontWall(){
  if(frontSensor() < DST){
     return true;
  }
  return false;
}

bool isAllDirOpen(){
  if(!isLeftWall() && !isRightWall() && !isFrontWall())
    return true;
   else
    return false;
}

bool isDeadend(){
  if(isFrontWall() && isRightWall() && isLeftWall()){
    return true;
  }else{
    return false;
  }
}

bool send_correct_path_to_fellow_robot(){
  Serial.println("Sending...");
  int result=radio.write(&correct_path, sizeof(correct_path));
  if(!result){
   Serial.println("oops!, send failed!!");
   return false;
  }
   Serial.println("sended successed");
   return true;
```

```
  delay(200);
}

// Function to copy 'len' elements from 'src' to 'dst'
void copy(uint32_t* src, uint32_t* dst, int len) {
    memcpy(dst, src, sizeof(src[0])*len);
}
```

کد مربوط به ربات همکار[40]:

```
/**
 *  Fellow Robot
 *  Cooperation of Multiple Robots to Solve Maze Tasks
 *  The circuit:
 *    1.Arduino Uno
 *    3.Dc Motor 100RPM
 *    2.L9110S
 *    3.Ultrasonic HC-SR04
 *    4.Nrf24l01
 *  Created At 20/12/2021
 *  By Amirhossein Rahmani & Mohsen Rahimi
 */
/*Initilize WIFI Madule*/
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>
#include <printf.h>

RF24 radio(9, 8); // (CE, CSN) for ARD Nano
const byte address[6] = {'R','x','A','A','A'};
#define MA1 5  // Motor A pins
#define MA2 6
```

[40] fellow-robot

```
#define MB1 10 // Motor B pins
#define MB2 11

#define PWM_RIGHT 200
#define PWM_LEFT 120

uint32_t path[36];
bool start_=false;
bool is_read_data=false;
/**
 * path=1 --> move forward
 * path=2 --> turn right
 * path=3 -->turn left
 */
void setup() {
  Serial.begin(9600); // opens serial port, sets data rate to 9600 bps
  pinMode(MA1, OUTPUT);
  pinMode(MA2, OUTPUT);
  pinMode(MB1, OUTPUT);
  pinMode(MB2, OUTPUT);

  radio.begin();
  radio.openReadingPipe(1,address); // for Ard. NANO
  radio.setPALevel(RF24_PA_MAX);
  radio.startListening();
}

void loop() {
  if(!is_read_data){
    while(isDataAvailable()){
      get_data();
      if(isValidData()){
```

```
        start_=true;
        is_read_data=true;
      }
    }
}

 if(start_){
    for(int i=0;i<36;i++){
      if(path[i]==1){
        forward();
        delay(100);
        Stop();
      }

      if(path[i]==2){
        turnRight();
        delay(50);
        forward();
        delay(100);
        Stop();
      }

      if(path[i]==3){
        turnLeft();
        delay(50);
        forward();
        delay(100);
        Stop();
      }

      delay(4000);
      Stop();
```

```
    }
    start_=false;
    //end maze
  }
}

void  forward() {          //function of forward
  analogWrite(MA1, LOW);
  analogWrite(MA2, PWM_LEFT);
  analogWrite(MB1, LOW);
  analogWrite(MB2, PWM_RIGHT);
}

void backward() {         //function of backward
  analogWrite(MA1, HIGH);
  analogWrite(MA2, 0);
  analogWrite(MB1, HIGH);
  analogWrite(MB2, 0);
}

void Stop() {              //function of stop
  analogWrite(MA1, LOW);
  analogWrite(MA2, LOW);
  analogWrite(MB1, LOW);
  analogWrite(MB2, LOW);
}

void turnLeft() {         //function of turn left
  analogWrite(MA1, PWM_LEFT);
  analogWrite(MA2, LOW);
  analogWrite(MB1, LOW);
  analogWrite(MB2, PWM_RIGHT);
```

```
}

void turnRight() {          //function of turn right
  analogWrite(MA1, LOW);
  analogWrite(MA2, PWM_LEFT);
  analogWrite(MB1, PWM_RIGHT);
  analogWrite(MB2, LOW);
}

bool isDataAvailable(){
  if (radio.available()) {
    Serial.print("Received Data : ");
    return true;
  }
  Serial.println("Not Receiving !!!");
  return false;
}

void get_data(){
  radio.read(path, sizeof(path));
  // check
}

bool isValidData(){
  int count=0;
  for(int i=0;i<36;i++){
    if(path[i] > 0 && (path[i]==1 || path[i]==2 || path[i]==3)){
      return true;
    }
  if(path[i]==0){
    count++;
  }
```

```
 }
 if(count>30){
  return false;
 }
}
```

Abstract:

The quick development of technology leads us to careful planning for best selection. These technologies and innovations in the lives of individuals making their work easier. in this paper,autonomous maze solving robot is developed with independent mapping and localization skill. in first step, the maze solving bot is designed with three ultrasonic distance sensor which is used for wall detection to avoid collision and obstacle detection. Also, it expect to use robot where an environment unreachable for human. In addition, there are also places where use of robots is the only way to achieve a goal. We have successfully implemented the maze solution ability on the robot. The result of the experiment was that the robot could successfully solve the maze with the walls and also save the correct path in its memory and send the saved path to the fellow robot at the end of the path.In this design, the possibility of sending positions to other robots is embedded.

Keywords:

Maze – Micro mouse – Algorithm – Graph – Graph navigation – Micro Controller – ATmega 328 – Sensors

منابع

[1] https://en.wikipedia.org/wiki/Maze-solving_algorithm
[2] https://www.geeksforgeeks.org/flood-fill-algorithm/
[3] https://en.wikipedia.org/wiki/Graph_theory
[4] https://www.tutorialspoint.com/data_communication_computer_network/wireless_transmission.htm
[5] https://blog.faradars.org/graph-data-structure/
[6] https://roboworld.com/how-to-select-the-right-motor-for-your-robot/
[7] https://www.baeldung.com/java-solve-maze
[8] https://lastminuteengineers.com/nrf24l01-arduino-wireless-communication/

Faculty of Electrical Engineering

Project Title:

Cooperation of Multiple Robots to Solve Maze Tasks

Supervisor:

Dr Nassor Bagheri

By:

Mohsen Rahimi & Amirhossein Rahmany

A thesis submitted to the Graduate Studies Office in partial fulfillment of the requierments for the degree of Masters in the Electronic Engineering

February / 2022