به نام خدا

بازی Snake ، پروژه درس ریزپردازنده

رسول کامکار – سپهر شیرانی – علی ملاحسینی

استاد فرزانه شایق

فهرست

فهرست.....2

مقدمه 3

اهمیت و جایگاه بازی ها 3

علت انتخاب بازی مار 3

تاریخچه 4

نسخه های بعدی.....5

قطعات استفاده شده.....6

منطق، مکانیک ها و نرم افزار بازی 7

صحنه های بازی و تفاوت های آن ها 7

جدول و کاراکتر های بازی 7

درجه های سختی بازی 7

محاسبات فریم به فریم و تایمر دار.....8

گرفتن ورودی از بازیکن.....8

پردازش ورودی های جهتی و حرکت مار 8

پیشرفت در بازی و رشد مار 9

نحوه قرار گیری میوه در زمین بازی 9

پایان بازی (مردن مار).....9

نمایش بازی بر روی ماتریکس.....10

سخت افزار و اتصالات.....10

مقدمه

اهمیت و جایگاه بازی ها

بازی ها در دنیای امروزی نقش جدایی ناپذیری را دارند. و جمعیت بزرگی از پیر و جوان مشغول به آن هستند که چرا باعث سرگرمی و لذت می‌شود همچنین در طی این راه آموزش هایی هم به آن ها می دهد که باعث می شود تا حدودی شبیه سازی از زندگی واقعی تجربه کنند و این تجارب در زندگی عادی و روزمره به کمک آن ها می آید.

بازی کردن فواید و مضراتی دارد همانطور و استفاده درست و کافی از آن می تواند تاثیر های بسیاری برای بازیکن ها به وجود بیاورد. از مضرات آن به آسیب های عضلانی و یا اذیت شدن چشم ها می توان اشاره کرد که در مدت زمان زیاد بازی کردن پیوسته به وجود می‌آید ولی از فواید آن می توان به رشد مغز بازیکن و هماهنگی بیشتر و سریع تر اعضای بدن بازیکن اشاره کرد و همچنین سرعت انتخاب و تصمیم گیری در لحظه را افزایش می‌دهد. همچنین خیلی از بازی ها بستری هستند که بازیکن را با دانش و یا انواع چیز های جدید آشنا می‌کند و چه بسا بازیکن پس از این تجربه به دنبال آن ها برود و بر دانش خود بیافزاید.

پس می توان گفت بازی ها علاوه بر مضراتی که دارند فواید سودمندی دارند که بر ذهن و بدن بازیکن ها تاثیر می گذارد و باعث رشد و پیشرفت آن ها می‌شود.

علت انتخاب بازی مار

از دلایلی که باعث شد ما به سراغ این پروژه برویم می توان در ابتدا به حس نوستالژی آن اشاره کرد و همچنین علاقه اعضا تیم به ساختن بازی که سبب می شود که یک پروژه کاملا تعاملی به وجود بیایید و بشدت برای کاربر سرگرم کننده باشد.

برای انتخاب بازی ما به سراغ بازی های قدیمی معروفی رفتیم و از بین گزینه هایی همچون Hangman، Tetris و .... در نهایت بازی کلاسیک Snake انتخاب شد.

طی امکان‌سنجی هایی که تیم ما انجام داد که معیار های آن می توان به سطح سختی پیاده سازی بازی، استفاده از ابزار های اشاره شده در درس و همچنین امکان پیاده سازی آن در بازه مشخص، ما این بازی را انتخاب کردیم.

تاریخچه

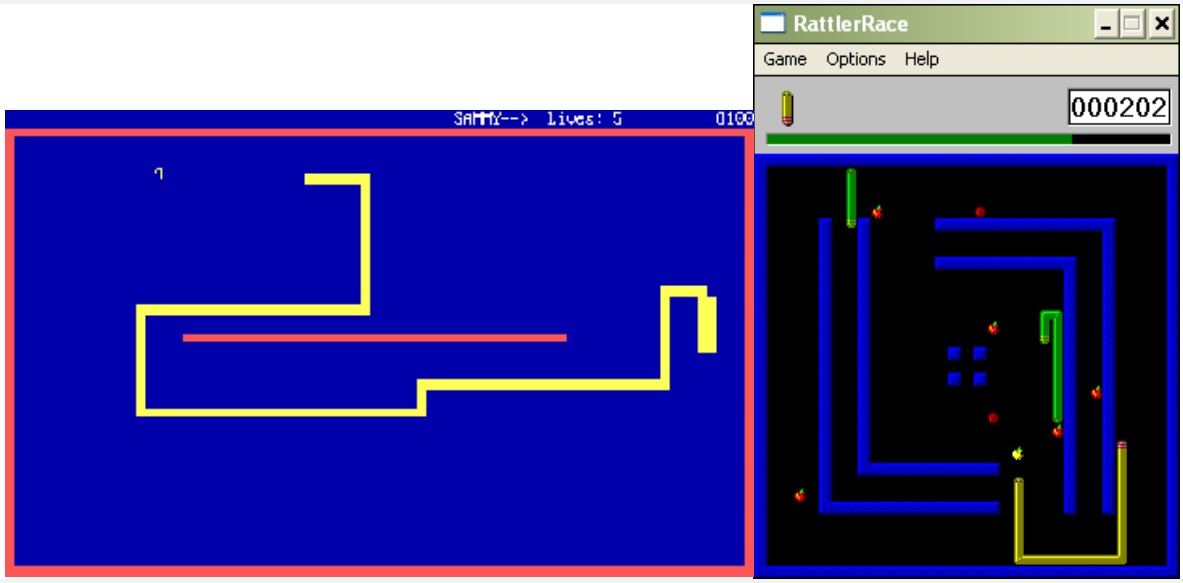
اولین نسخه بازی مار به سال ۱۹۷۶ باز میگردد که برای اولین بار تحت نام Blockade توسط شرکت Gremlin طراحی و منتشر شد. در سال ۱۹۷۷ دو بازی الهام گرفته شده از بازی Blockade توسط شرکت Atari منتشر شد. بازی آرکید Dominos و Surround.

اولین باری که این بازی برای کامپیوترهای خانگی طراحی شد به سال ۱۹۷۸ باز میگردد که توسط پیتر ترفونز برای TRS-80 طراحی کرد، که بعدا همین بازی برای Apple II، VIC-20، Atari 8-bit و دستگاه‌های دیگر نیز منتشر شد.

بازی تک نفره Nibbler که در سال ۱۹۸۲ منتشر شد بازی تک نفره آرکیدی بود که مار در آن به سختی در یک ماز جا میشد و گیم پلی آن سریع تر از همتاهای خود بود. تصاویری از گیم پلی آن در پایین نمایش داده شده است.

شکل 1 تصاویری از بازی Nibbler

از سال ۱۹۹۱ بازی Nibbles در پک آموزشی زبان برنامه‌نویسی QBasic به عنوان یک برنامه سمپل قرار داده شد. در سال ۱۹۹۲ Rattler Race به عنوان بخشی از Microsoft Entertainment Pack منتشر شد. این بازی مارهای دشمن را به گیم پلی معروف مار و سیب اضافه کرد.



شکل 2 تصاویری از بازی Rattler Race

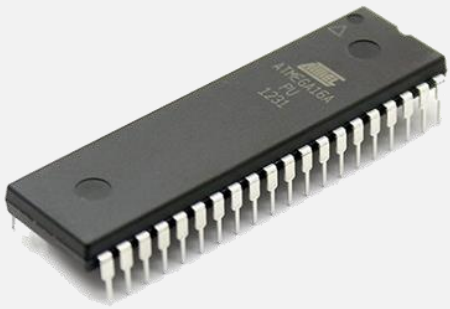
نسخه های بعدی

از نسخه های معروف دیگر این بازی میتوان به لیست زیر اشاره کرد:

- Serpent که برای Game Boy منتشر شد.
- Meerca Chase نسخه دیگری است که برای Neopet ارائه شد.
- Slither.io که برداشت چند نفره ای از بازی مار است.
- نسخه easter egg بازی که توسط گوگل در ۲۰۱۷ معرفی شد.

قطعات استفاده شده

Atmega32 (in proteus), Atmega16 (for implementation)



Resistor

Momentary push to ON switch

8x8 LED Matrix

منطق، مکانیک ها و نرم افزار بازی

صحنه های بازی و تفاوت های آن ها

در نرم‌افزار سه صحنه (مشخص شده با enum Scene) پیاده سازی شده، منو، بازی و مرگ.

در منو، بازیکن می‌تواند با دکمه های چپ و راست درجه سختی بازی را انتخاب کند و بر روی ماتریکس، حرف H برای حالت سختتر و حرف E برای حالت آسان بازی نمایش داده می‌شود. پس از انتخاب بازیکن می‌تواند با کلید Ok درجه سختی را انتخاب کند، صحنه به بازی تغییر پیدا می‌کند و بازی شروع می‌شود.

در صحنه بازی، خانه های جدول بازی که مار و میوه در آن قرار گرفته اند نمایش داده می‌شود و بازیکن می‌تواند با کلید های جهت، مار را کنترل کند.

در صحنه مرگ، کلمه GAMEOVER به صورت حرف به حرف به کاربر نمایش داده می‌شود و بازیکن می‌تواند با کلید Ok، به صحنه منو بازگردد.

جدول و کاراکتر های بازی

بازی به صورت یک Grid 8 در 8 در نظر گرفته شده تا نمایش آن بر روی ماتریکس امکان پذیر باشد

هر سلول این جدول با یک طول و عرض مشخص می‌شود. ساختار Point برای ذخیره این طول و عرض در نظر گرفته شده، آرایه اصلی Snake و همینطور متغیر میوه از این ساختار تشکیل شده اند.

درایه 0 آرایه مار، سر مار و دیگر درایه ها بدن مار را نشان میدهند و طول مار با متغیر length مشخص می‌شود، هر کدام از بخش های مار و همینطور میوه مختصات مربوط به خود را دارند که بر اساس آن محاسبات مورد نیاز و نمایش بر روی ماتریکس انجام می‌شود.

درجه های سختی بازی

بازی شامل دو درجه سختی Hard و Easy است که به صورت Difficulty enum در نرم‌افزار مشخص شده و در یک متغیر به نام diff ذخیره می‌شود، تفاوت این دو سختی بازی در سرعت حرکت مار است. در درجه سخت، آپدیت های مربوطه هر نیم ثانیه یک بار انجام شده در حالیکه در درجه آسان بازی، آپدیت ها هر یک ثانیه یک بار انجام می‌شوند. بازیکن در منو بازی می‌تواند با کلید راست درجه را بر روی سخت و با کلید چپ بر روی آسان تنظیم کند

محاسبات فریم به فریم و تایمر دار

در نرم‌افزار با الهام از موتور های بازی سازی، محاسبات به طور کلی به دو بخش فریم به فریم و تایمر دار تقسیم شده است.

محسابات فریم به فریم شامل گرفتن ورودی، تغییر درجه سختی در منو بازی و نمایش بازی بر روی ماتریکس، در یک لوپ در تابع main اجرا میشوند (توابع inputKey, updateDifficulty, draw)

محاسبات تایمر دار که مربوط به آپدیت شدن اصلی بازی هستند، شامل حرکت مار و تغییر جهت، خوردن و قرار گیری دوباره سیب و چک کردن اتمام بازی پس از حرکت مار همگی در یک تایمر اجرا می‌شوند.

لازم به ذکر است که این تایمر به صورت وقفه نیم ثانیه ای overflow بر روی Timer1 است و اگر درجه سختی بازی بر روی آسان باشد، آپدیت های ذکر شده پس از دوبار وقفه (یعنی هر یک ثانیه) انجام می‌شوند. متغیر updateCounter وظیفه شمارش وقفه ها را بر عهده دارد

آپدیت های تایمر دار تنها در صحنه Game انجام می‌شوند.

گرفتن ورودی از بازیکن

در ابتدا، 5 پایه اول پورت C به صورت ورودی و pull-up تظیم شده اند.

5 دکمه که از یک سمت زمین شده و از سمت دیگر به این پایه ها متصل شده که در صورت فشار دادن، آن پایه ها مقدار 0 منطقی می‌گیرند. پایه 0 برای جهت بالا، 1 جهت راست، 2 جهت چپ، 3 جهت پایین و 4 برای دکمه Ok در نظر گرفته شده اند.

خواندن ورودی از این دکمه ها در تابع inputKey انجام می‌شود، به این صورت که مقدار این لحظه پایه با مقدار قبلی آن مقایسه شده، اگر لبه پایین رونده شناسایی شد، متغیر inputDir (برای ورودی های جهتی) تنظیم میشود.

در صورتی که دکمه Ok فشار داده شده بود و صحنه، منو یا مرگ بود بازی به صحنه بعدی می‌رود.

پردازش ورودی های جهتی و حرکت مار

جهت حرکت مار با استفاده از enum Dir و متغیر moving تعیین می‌شود.

در تابع updateMovingDirection، اخرین ورودی جهتی کاربر که در متغیر inputDir ذخیره شده در متغیر moving ذخیره میشود، به شرطی که باعث چرخش 180 درجه مار نشود (به طور مثال ورودی پایین هنگام حرکت مار به سمت بالا نامعتبر است)

حرکت مار به این صورت است که با شروع از دم، هر قسمت بدن مار در جای قسمت جلویی قرار میگیرد، در نهایت سر مار با توجه به متغیر moving تغییر مکان میدهد (توابع updateSnake و updateHead)

لازم به ذکر است که محاسبات ذکر شده در بخش تایمر دار قرار دارند.

پیشرفت در بازی و رشد مار

پیشرفت در بازی به این صورت است که مار باید به سمت میوه قرار گرفته شده در زمین بازی حرکت کند، هنگامی که سر مار و میوه در یک خانه قرار گرفتند (تابع checkCollectedFruit) مختصات میوه تغییر کرده و مار رشد می‌کند.

رشد مار به این صورت است که یک قسمت جدید به آرایه مار اضافه شده، مختصات آن برابر دم مار تنظیم شده و در نهایت یک واحد به متغیر طول مار (length) اضافه می‌شود.

نحوه قرار گیری میوه در زمین بازی

هنگام شروع بازی و همینطور هنگام خورده شدن میوه، مختصات میوه به صورت رندم از خانه های آزاد زمین انتخاب می‌شود (تابع pickRandomFruit).

به این صورت که تمامی 64 خانه بازی مورد بررسی قرار می‌گیرند که قسمتی از بدن مار در آن خانه قرار دارد یا خیر. در صورت آزاد بودن، آن خانه به آرایه possiblePoints اضافه می‌شود، در نهایت یکی از خانه های معتبر به صورت رندم انتخاب میشود و میوه در آن خانه قرار می‌گیرد.

پایان بازی (مردن مار)

پس از هر بار حرکت مار، شرط های مرگ مار بررسی می‌شود (تابع checkForDeath).

در صورتی که سر مار با یکی از بخش های بدن آن برخورد کرده باشد

(تابع checkHeadCollidedWithBody)

در صورتی که سر مار از زمین بازی خارج شود (تابع checkIfHeadOutOfBoundries)

در صورتی که یکی از شرط ها برقرار باشد، صحنه بازی به DEAD تغییر می‌کند.

نمایش بازی بر روی ماتریکس

همانطور که گفته شد، تابع draw به صورت مکرر صدا زده می‌شود.

در این تابع، بر اساس صحنه فعال در بازی، یکی از توابع drawMenu, drawGame, drawDead صدا زده می‌شود.

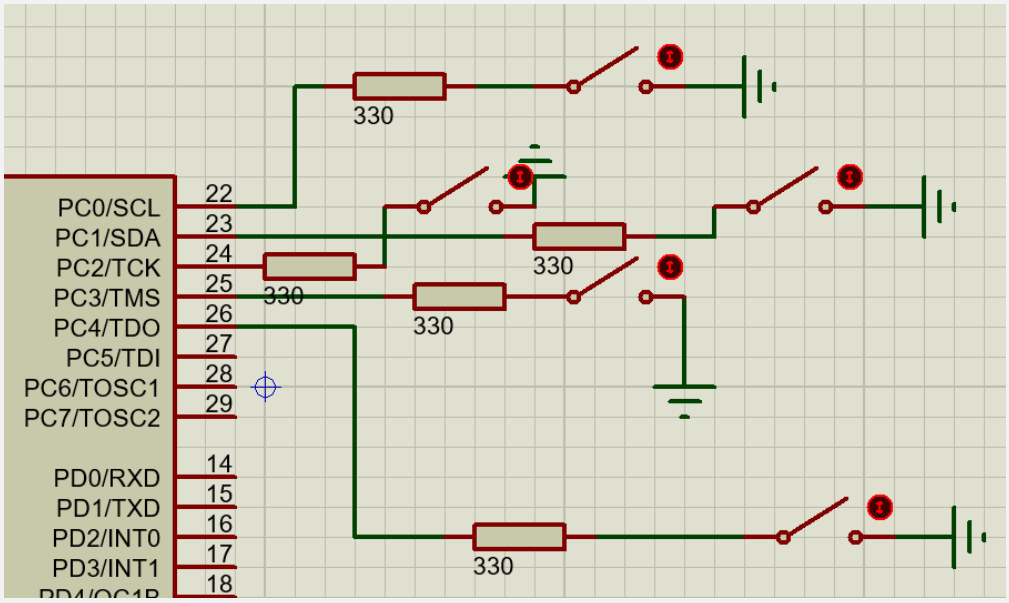
در تابع نمایش منو، مقدار سختی بازی در متغیر diff بررسی میشود، در صورت سخت بود بازی، کاراکتر H و در صورت آسان بود، کاراکتر E از لیست الفبا انتخاب شده و مقادیر متناظر آن بر روی ماتریکس قرار می‌گیرد.

در تابع نمایش بازی، 8 متغیر 8 بیتی، که هر بیت آن متناظر با یک خانه بازی و یک LED در ماتریکس است در نظر گرفته شده. سپس بر اساس مختصات هر قسمت از بدن مار و همینطور مختصات میوه، مقادیر بیت های این 8 متغیر ست شده و در نهایت نمایش بر روی ماتریکس انجام می‌شود.

در تابع نمایش پایان بازی، کاراکتر های کلمه GAMEOVER به ترتیب انتخاب شده و مقادیر آن بر روی ماتریکس قرار می‌گیرد.

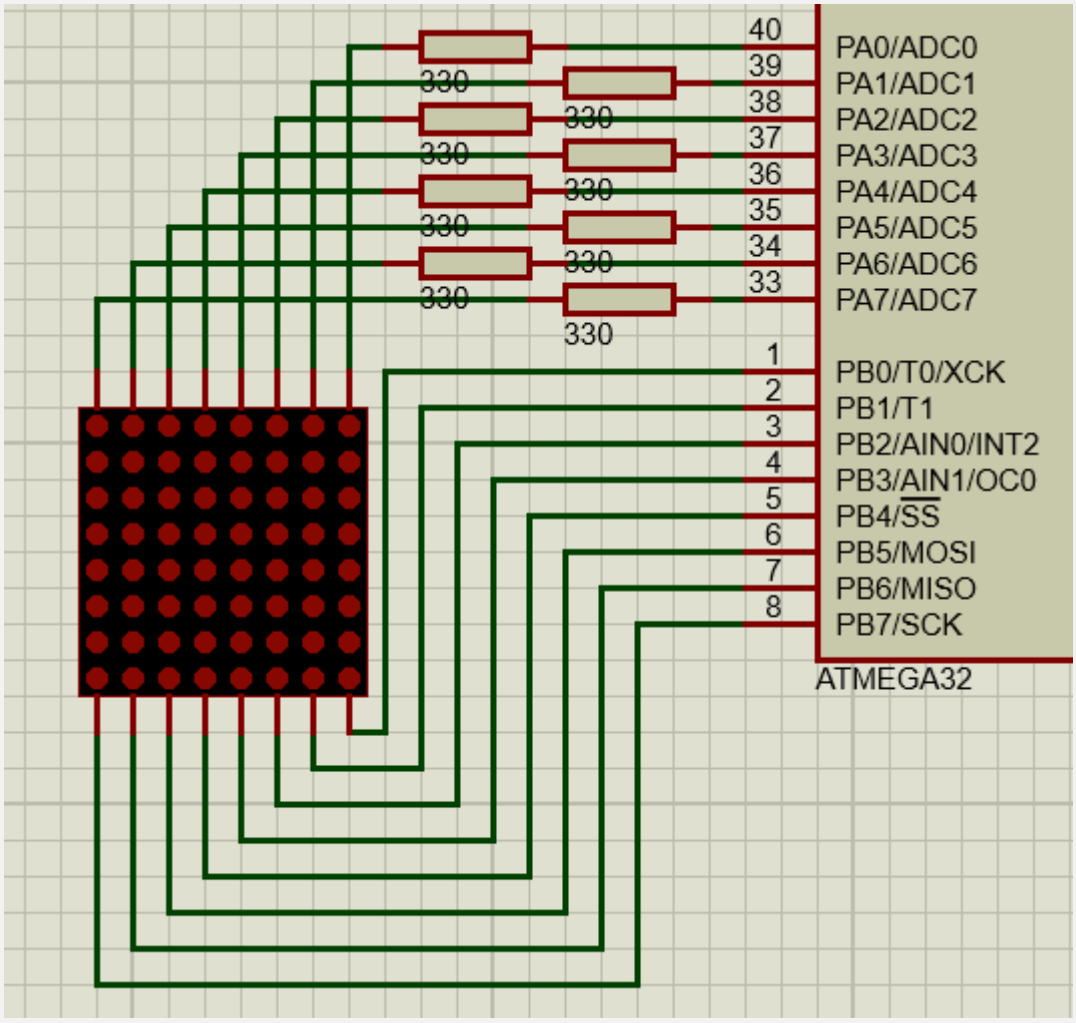
سخت افزار و اتصالات

یک سمت سوییچ های push to ON به زمین و سمت دیگر آن ها با واسطه مقاومت به 5 پایه اول پورت C کنترلر متصل اند



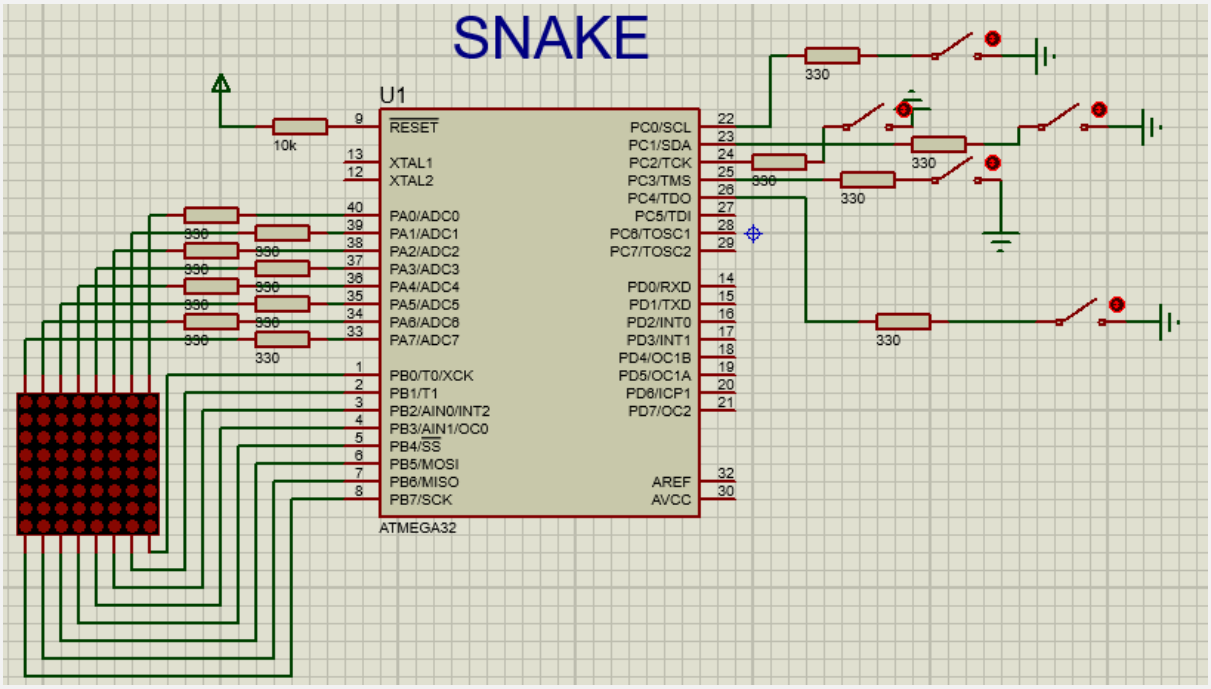
شکل 3 بخش ورودی مدار

ماتریس 8 در 8 به پورت های A (با مقاومت 330) و B متصل است (پورت A به عنوان منبع و پورت B به عنوان زمین ال ای دی ها عمل می‌کنند)

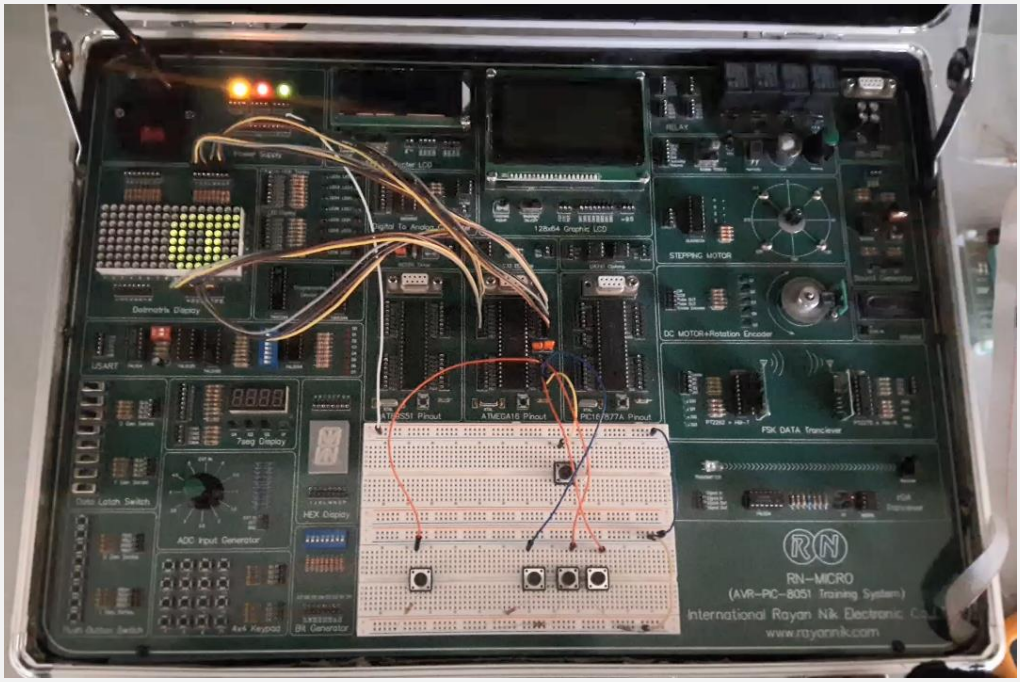


شکل 4 بخش ماتریکس مدار

پایه RESET کنترلر هم به منبع با مقاومت 10 کیلو متصل است



شکل 5 مدار کامل در Proteus



شکل 6 مدار کامل پیاده سازی شده