# Софийски университет „Св. Климент Охридски”

## Факултет по математика и информатика

## Проект по дисциплината

## “Размити множества и приложения”


Проект на:
Мария Пламенова Паскова фн. 25311

# 1. Увод

Преди да се изучава каквото и да е по програмиране, се използват блок схеми. Те са в основата на много от операторите и алгоритмите в математиката и съответно в програмирането. В блок схемите има няколко вида фигури - условие и различни случаи в зависимост от условието. Най-простият оператор, който описва тези блок схеми е операторът - if - then - else. Чрез него се дефинират и правила. Пример:
“Ако температурата е 0 градуса водата замръзва.”
“Ако температурата е 20 градуса, водата е в течно състояние.”

Тези правила могат да бъдат приложени и в теорията на размитите множества. Ако величините, които участват в тези правила са размити числа свеждаме задачата до задача за размити множества. Пример за такава задача може да бъде:

“Ако времето е много хубаво, ще спортувам малко”
“Ако  времето е хубаво, ще спортувам много ”
“Ако времето е хладно ще спортувам евентуално”
“Ако времето е лошо няма да спортувам”

В този пример величините са размити стойности като хубаво, хладно и т.н. За такъв тип данни е много подходящо да се използват размити множества.
Целта на тези правила е при дадени нови данни, системата да вземе решение за неизвестната променлива.

# 2. Теоретична постановка и използван алгоритъм

Задачата, която съм разгледала е реализация на система базирана на размити правила и вземане на решения на база тези правила.
Правилата се дефинират по следния начин :

If A1, B1, then C1
If A2, B2, then C2
If A3, B3, then C3
If A4, B4, then C4
If A5, B5, then C5

...
If A, B, then C - където C е неизвестно и целта на алготъма е да го изчисли на база предишните правила.

Първата задача, която си поставих за решаване с този алгоритъм е движение на моторно средство, което да спира пред препятствия и да не се блъска с тях.
Има възможност да се сложи сензор на превозното средство. Първоначалната концепция беше правилата да изглеждат по следния начин:

“ Ако разстоянието в ляво е малко, разстоянието в дясно е малко, скоростта е ниска.”

В тези правила променливите са следните:

- растояние в ляво
- разстояние в дясно
- скорост

Друг вариант е да се гледа от пред, в ляво и в дясно - така променливите стават четири.
При реализирането на тази задача превозното средство ще спре пред дадено предпятствие и няма да направи нищо друго. В този момент се замислих, че ако успява да избяга от препятствията ще бъде по-добро и реално. По този начин се промениха и първоначалните условия. Променливите са:

- разстояние
- скорост

Правилата стават по-прости, но работят много по-добре и независимо от посоката.
Разстоянието е дадената величина, а скоростта е величината, която трябва да бъде извод от системата. Правилата са от тип:
“Ако разстоянието е много малко, скоростта е отрицателна”
“Ако разстоянието е малко, скоростта е ниска (отрицателна и положетелна)”
“Ако разстоянието е средно, скоростта е средна”
“Ако разстоянието е голямо, скоростта е висока”
Тези правила са обвити в размити числа и чрез тях е реализиран алгоритъмът за вземане на решение.

Конкретното превозно средство, за което беше реализиран алгоритъма има две вериги, които го задвижват. За двете вериги са използвани два сензора, които измерват разстоянието в две посоки:

- напред - ляво
- напред - дясно

Двата сензора са свързани независимо един от друг към алгоритъма, който измерва в противоположна посока разстоянието до предмет.
Левият сензор контролира дясната верига и десният сензор - лявата верига.
Всеки сензор определя разстояние, което изпраща на алгоритъма и то пресмята с каква скорост трябва да се движи съответната верига. По този начин един и същи алгоритъм се използва два пъти за двете вериги. Чрез тази реализация на задачата може да се реализира движение в много посоки при добавяне на още сензори.

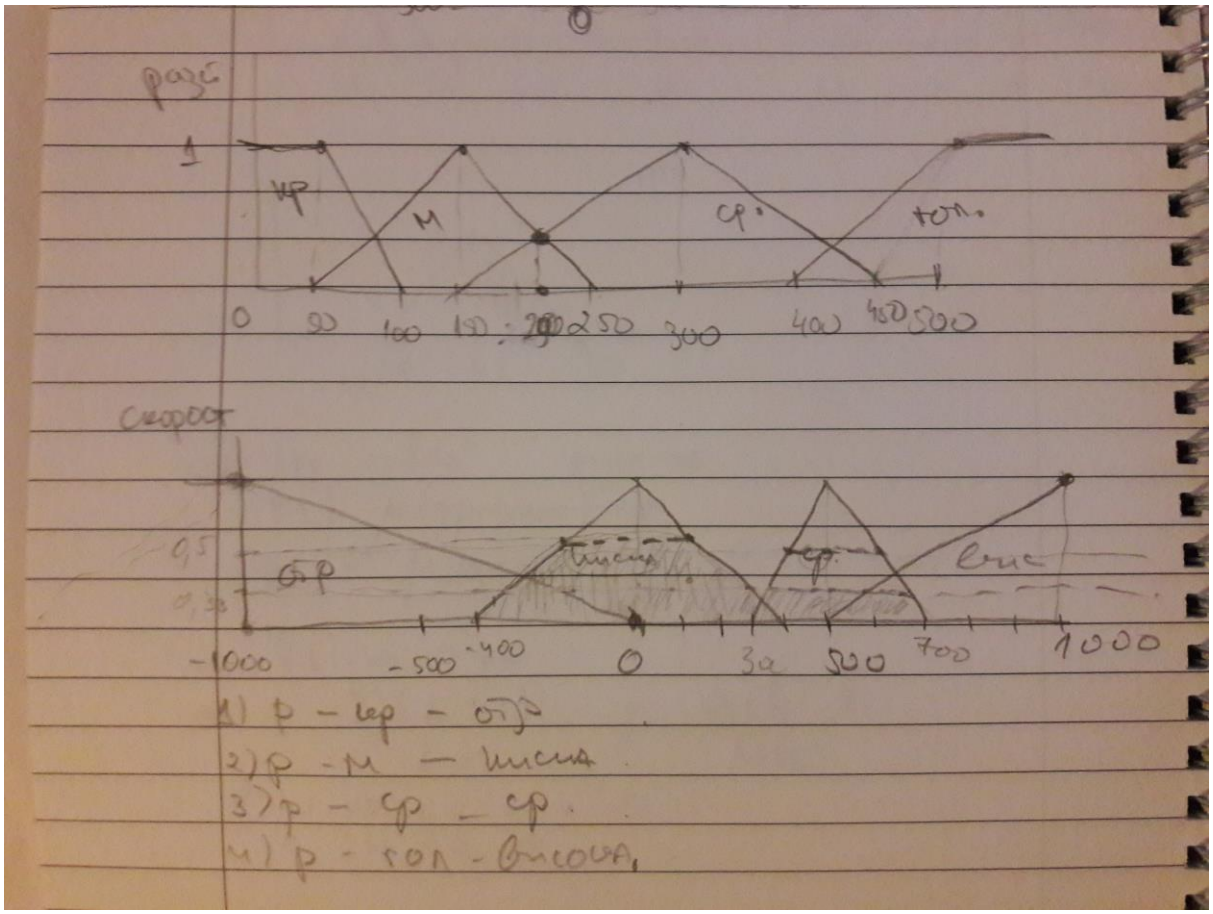
Предварителна подготовка: Анализиране на променливите и дефиниране на размити стойности.

Стъпки на алгоритъма:

- Фъзификация
- Прилагане на правилата - пресмятане на степен на принадлежност
- Дефъзификация

## 2.1. Фъзификация

След определяне на правилата и променливите беше нужно да се определят самите размити стойности на променливите.

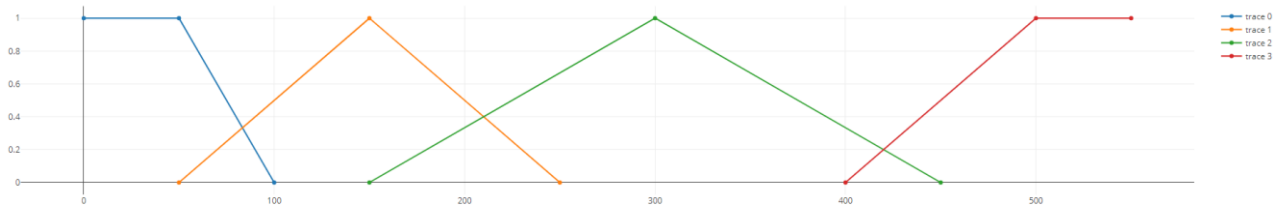



Първоначално стойностите на лингвистичните променливи са:
Разстояние -

- vshort - {0,50,100}
- short - {50,150,250}

- medium - {150,300,450}
- large - {400,500,500}

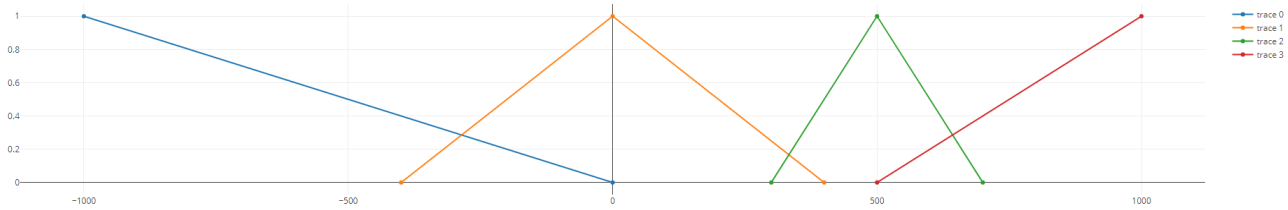
Визуално изглеждат по следния начин:




В последствие след свързването на алгоритъма с танка - “Станка”, бяха направени промени в зависимост от поведението ѝ.

Скорост -

- negative - {-2000,-1000,0}
- low - {-400,0,400}
- medium - {300,500,700}
- high - {500,1000,1500}

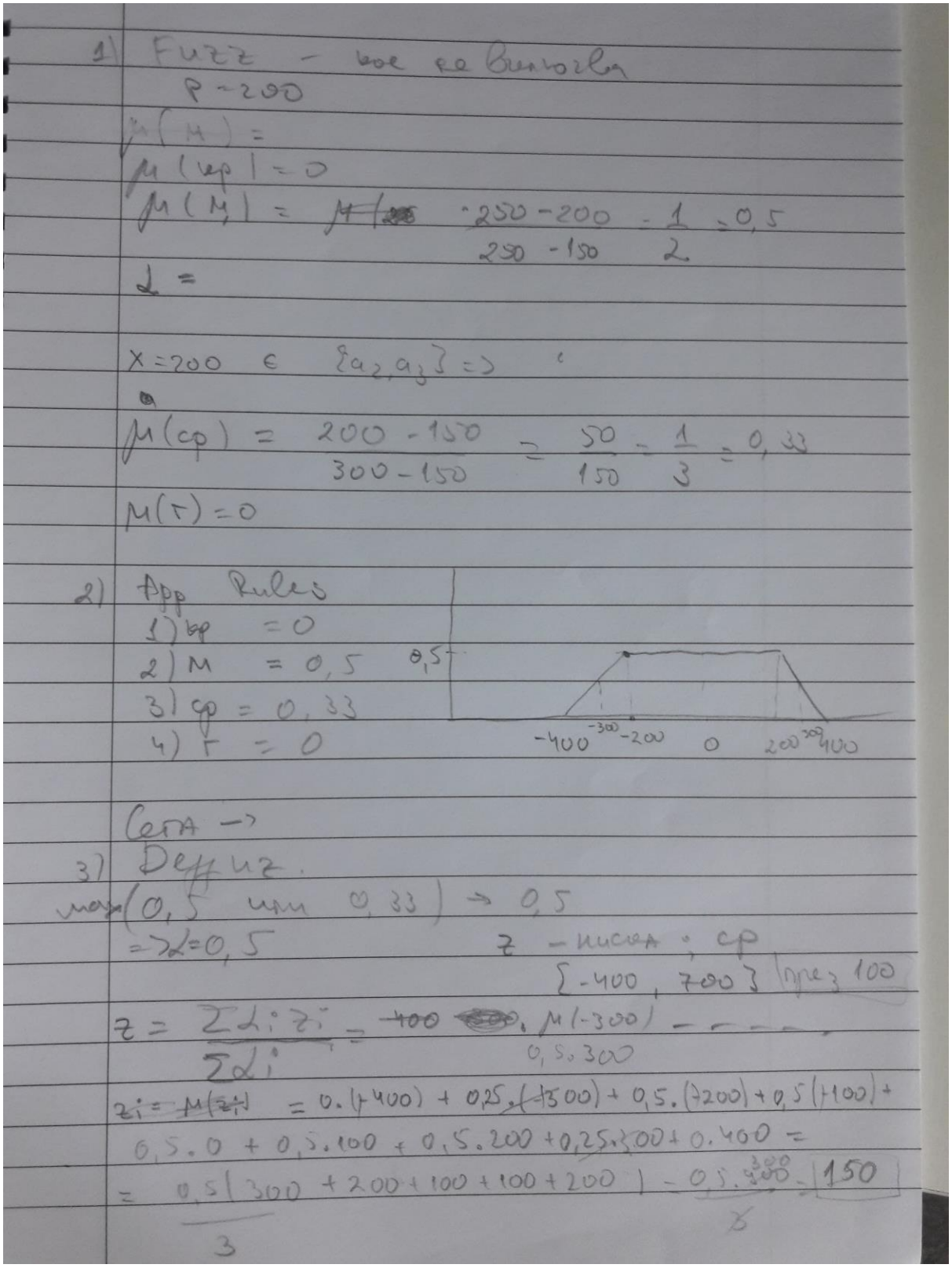



За определянето на тези размити стойности бяха използвани следните факти:

- Над 50 см е достатъчно голямо разстояние за да има нужда от намаляне на скоростта
- Има възможност за отрицателна скорост - движение назад
- Скоростта на танка е нормализирана между -1000 и 1000

(Тези диаграми могат да бъдат разгледани във файла variables.html)

## 2.2. Прилагане на правилата




## 2.3. Дефъзификация

Пресмятане на Z - център на тежестта чрез формулата:

$$CG = \frac{W_1 d_1 + W_2 d_2 + W_3 d_3 \ldots}{W}$$

# 3. Експериментални резултати

## 3.1. Реализиране на алгоритъма

Реализацията на алгоритъма е написана на JavaScript, за да бъде верифицирана идеята.
За да се пресмята целия алгоритъм от превозното средство беше нужно да се сложат няколко хардуерни части:

- Node MCU - opensource firmware and development kit - платка, която има процесор, WiFi и се програмира на Lua - подходяща платка за малки IOT продукти
- платка, която свързва моторите и изпраща сигнали до Node MCU - драйвер за моторите
- сензори за разстояние - в първоначалната идея беше нужен един, в последствие бяха добавени два сензора за двете посоки
- захранване на платките - в началото чрез свързване в компютър, в последствие батерия за зареждане и в последната итерация презареждащи се батерии

След реализиране на алгоритъма на JS, същия алгоритъм бе написан на Lua - ези за програмиране, който се използва в NodeMCU.
Освен основният алгоритъм - (fuzzy.lua)  реализирането на система за решаване на размити правила, бяха реализирани и следните модули:

- server.lua - създаване на сървър и сервиране на файлове
- speed.lua - Свързване на двете платки и задаване на конкретна скорост на двата мотора
- autoSpeed.lua - Вземане на дистанциите от сензорите на всеки интервал от време и задаване на скорост спрямо тях ( тук се извиква резултата от алгоритъма за размити правила)
- distance.lua - модул за изчисляване на дистанцията на набор от сензори
- init.lua - файл, който сервира всички файлове и се стартира при пускане на mcu-то
- index.html - страничка, която се достъпва чрез телефон/компютър за превключване на ръчно управление на превозното средство
- app.js - файл, който изпраща данните от потребителя към сървъра

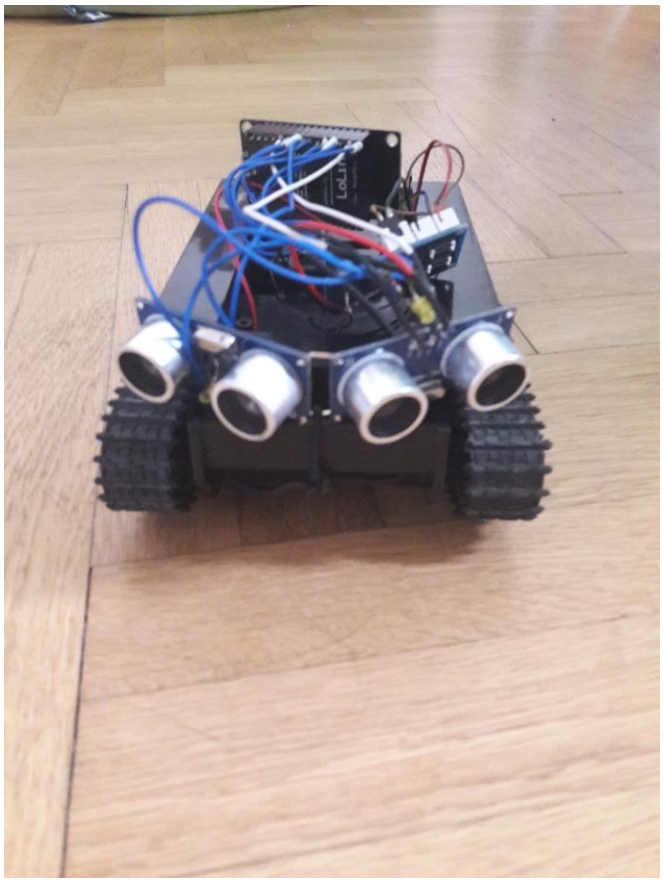
## 3.2. Промяна на входните данни

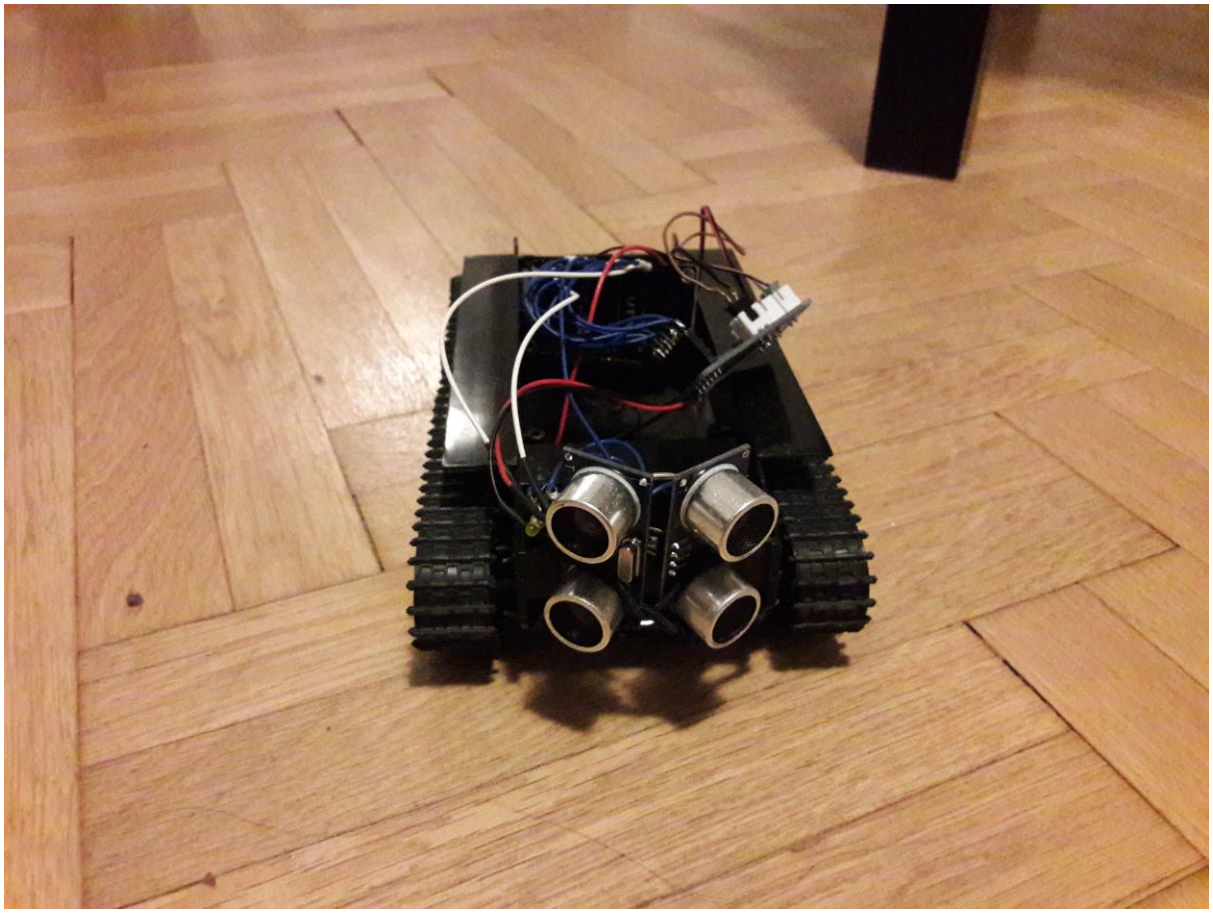
След реализиране на алгоритъма и свързване на всички сензори, жички и наблюдение на танка “Станка”, бяха направени някои модификации за по-плавно спиране. Модификациите бяха в началните стойности на разстоянието.

Визуално промяната може да се види във файла function.html.

## 3.3. Промяна на местоположението на сензорите

При свързване на първи сензор към танка, се наблюдаваше, че алгоритъма работи коректно, но след добавяне на втория сензор се видя, че е много важна позицията им, за да може да заобикаля предмети и да вижда предмети точно отпред.
Сензорите преминаха през няколко състояния като основните бяха:





След няколко размествания последната конфигурация работи най-добре от наблюдения.

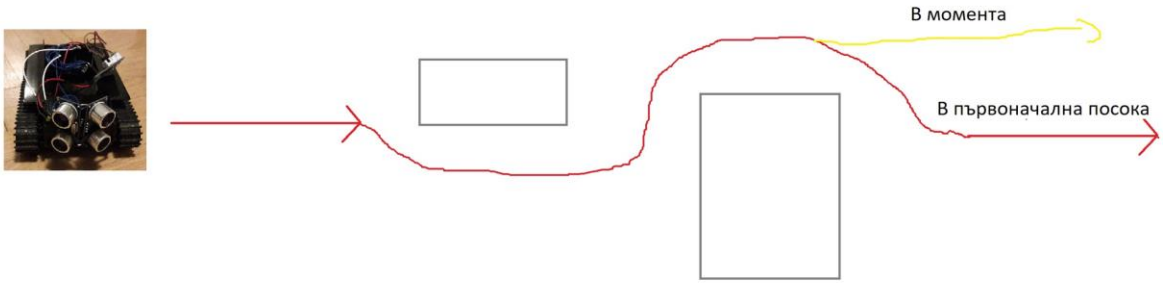
# 4. Заключение и бъдещи идеи

Има възможност да се включи като още една променлива данните от другия сензор с по-малък коефициент. Например:

Дясна верига: “Ако разстоянието от левия сензор е средно и разстоянието от десния сензор е малко , скоростта е ниска”. По този начин ще се вкара като втора променлива и вторият сензор.

Може да се прави още проучване в посока промяна на началните стойности на скоростта и разстоянието.

Друга възможна идея за развитие е завръщане на същото място след като заобиколи дадено препятствие. Визуално изглежда по този начин:




Тази идея тепърва предстои да бъде проучвана.

# 5. Литература


- http://www.nodemcu.com/index_en.html#fr_54745c8bd775ef4b99000011
- https://nodemcu.readthedocs.io/en/master/
- https://www.lua.org/cgi-bin/demo
- https://erelement.com/sensors/HC-SR04
- https://plot.ly/javascript/
-


# 6. Приложение

Този алгоритъм може да се приложи на различни превозни средства в зависимост от начина на движението им. Текущия алгоритъм бе приложен върху превозното средство танк - “Станка”, което има два мотора (две вериги).