Аннотация

Цель данной работы - реализация алгоритма нейронной сети, повторяющей работу мозга круглого червя нематоды C. elegans.

В данной работе рассмотрен круглый червь Caenorhabditis elegans в виду малого количества клеток нервной системы. Для проведения исследований изначально был сконструирован робот на базе робототехнического конструктора lego mindstorms ev3, после успешных испытаний было решено усложнить задачу и перенести нейросетевое управление на робота, построенного на основе микроконтроллера Arduino, для чего была собрана двухмоторная тележка с несколькими датчиками (датчик касания, датчик расстояния, гироскопический датчик, инфракрасный датчик – альтернатива сенсора еды). Позднее была разработана конструкция змееподобного робота, составные механические части которого построены в САПР Компас-3D и распечатаны на 3D-принтере. Управление реализовано с помощью микроконтроллера Arduino, который позволил управлять перемещением робота благодаря использованию 10 сервоприводов. Для управления всеми тремя роботами изначально написаны базовые линейные алгоритмы движения в помещении (по стенкам), позже принятие решений и обучение робота реализовано с помощью спайковой модели нейронной сети Ижикевича состоящей из 302 нейронов.

Ключевые слова: робот, нейронная сеть, Arduino, датчики, имитация мозга

Оглавление

Введение

Актуальность исследования.

Идея переноса нервной системы этой нематоды в робота не нова и появлялась уже неоднократно. Например, популярные реализации коннектома в Raspberry Pi GoPiGo или Lego EV3 роботах. Обычно подобную задачу решают с помощью сумматора с подобранными весами тормозящих связей, которые позволяют менять траекторию при обнаружении препятствия.

Объект исследования: возможность имитации мозга живого червя путем построения алгоритма на основе искусственной нейронной сети.

Предмет исследования: искусственный интеллект.

Цель исследования: Реализация алгоритма нейронной сети, повторяющей работу мозга круглого червя нематоды C. elegans.

Задачи исследования:

1. Анализ алгоритмов построения искусственного интеллекта;
2. Изучение и подбор нейросетевых алгоритмов;
3. Подбор языка программирования;
4. Написание кода;
5. Тестирование программы и исправление ошибок;
6. Тестирование работоспособности алгоритма на простейших роботах;
7. Подведение итогов (выводы).

Методы исследования: Поиск и анализ информации, сравнение информации, проведение эксперимента

Новизна исследования: Построена трехмерная модель и распечатана на 3D-принтере, использован микроконтроллер Arduino.

Теоретическая значимость исследования состоит в детальном изучении анатомии физиологии червя нематоды C. elegans**.**

Практическая значимость исследования: Создание роботизированной модели червя C. elegans.

1. Теоретическая часть

1.1. Червь C. elegans

Caenorhabditis elegans — свободноживущая нематода (круглый червь) длиной около 1 мм. Взрослая особь (самка) состоит из 959 клеток (самец — из 1031 клетки)

1.2. Искусственный нейрон

Основой всей модели должен быть искусственный нейрон, а точнее искусственная нейронная сеть. Для удобства и простоты понимания было решено использовать спайковую модель нейрона Ижикевича, которая позволяет не только визуализировать нейронную активность, но и обладает рядом интересных свойств, как синаптические токи, мембранные потенциалы и моделирует нейронные явления недалекие к настоящим. К реализации нейрона Ижикевича было добавлено еще дифференциальное обучение Хебба для укрепления связей между нейронами (случай, когда за 10-15 раз повторений движения червя начинают становиться систематизированными).

Коннектом нематоды — это простой список нейронов и связей между ними с уже заданными весами и типом соединения. В общем виде нас интересует только возбуждающие и тормозящие типы связей. Для поиска тормозящих нейронов потребовалось изучать как сами нейроны, так и нейротрансмитеры.

1.3. Искусственная сеть и разбор поведения червя

Исходя из ожидаемого поведения были найдены группы нейронов, стимуляция которых приведет к требуемой активности. Например, активизация сенсорных нейронов, отвечающих за обнаружения еды приводит к движению: ADFL, ADFR, ASGL, ASGR, ASIL, ASIR, ASJL, ASJR,

AWCL, AWCR, AWAL, AWAR. Обычно говорят, что движение вперед, но в реальности направление движения может внезапно меняться.

При стимуляции сенсорных нейронов отвечающих за обнаружение препятствия приводит к смене направления движения: ASHL, ASHR, FLPL, FLPR, OLQDL, OLQDR, OLQVL, OLQVR, IL1VL, IL1VR, IL1L, IL1R, IL1DL, IL1DR. И действительно, при стимуляции рецепторов поиска еды заметен и ритм и специфический рисунок активизации на мускульных нейронах. Но и возбуждение носовых рецепторов, которое должно приводить к смене направления движения, так же активизирует мускульные нейроны с характерным рисунком.

1. d: - - 1 - - - - - - - 1 1 1 - - - 1 - - - - - - -
2. v: -
3. v: - - 1 -
4. d: 1 - 1 - - - - - - - 1 1 1 - - - 1 - - - - - - -

Рис.1 D и v означает спинные и брюшные мышца соответственно. Патерн активизации мускульных нейронов используя спайковую модель нейрона.

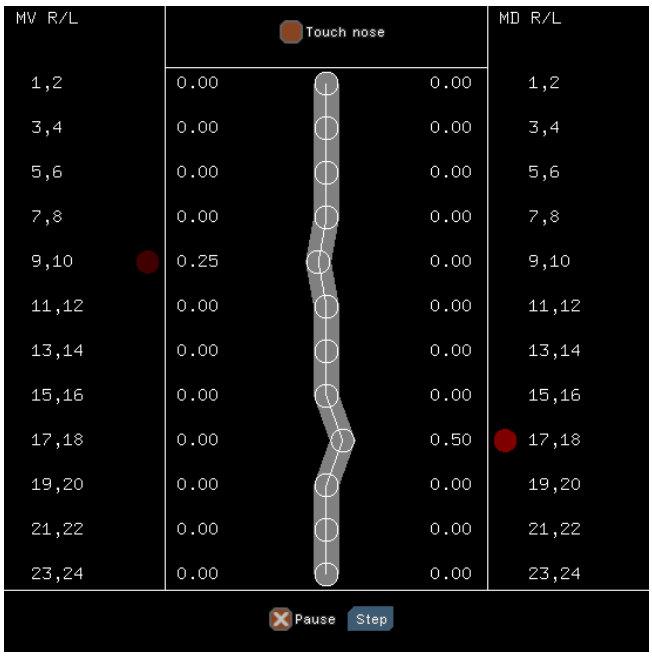

Рис 2. Разница между активностью брюшных и спинных мускульных нейронов. В динамике образует волноподобное движение.

Мускульные нейроны при движении вперед тоже активны, как и при движении назад. Но как определить в каком направлении идет движение или хотя бы момент смены направления движения? Движение вперед вызывается сигналом из AVB и PVC нейронов в B нейроны, а движение назад из AVA, AVD и AVE в A нейроны. Движения вперед и назад являются разного рода активностью и вызываются разными областями нервной системы. Хотя замечено, что эти области взаимодействуют с друг другом. И нейроны, отвечающие за движение, вперед играют какую-то роль при движении назад, т.е. активны при движении. Ниже показывается, что B нейроны активны при движении вперед, когда A нейроны активны при движении назад.

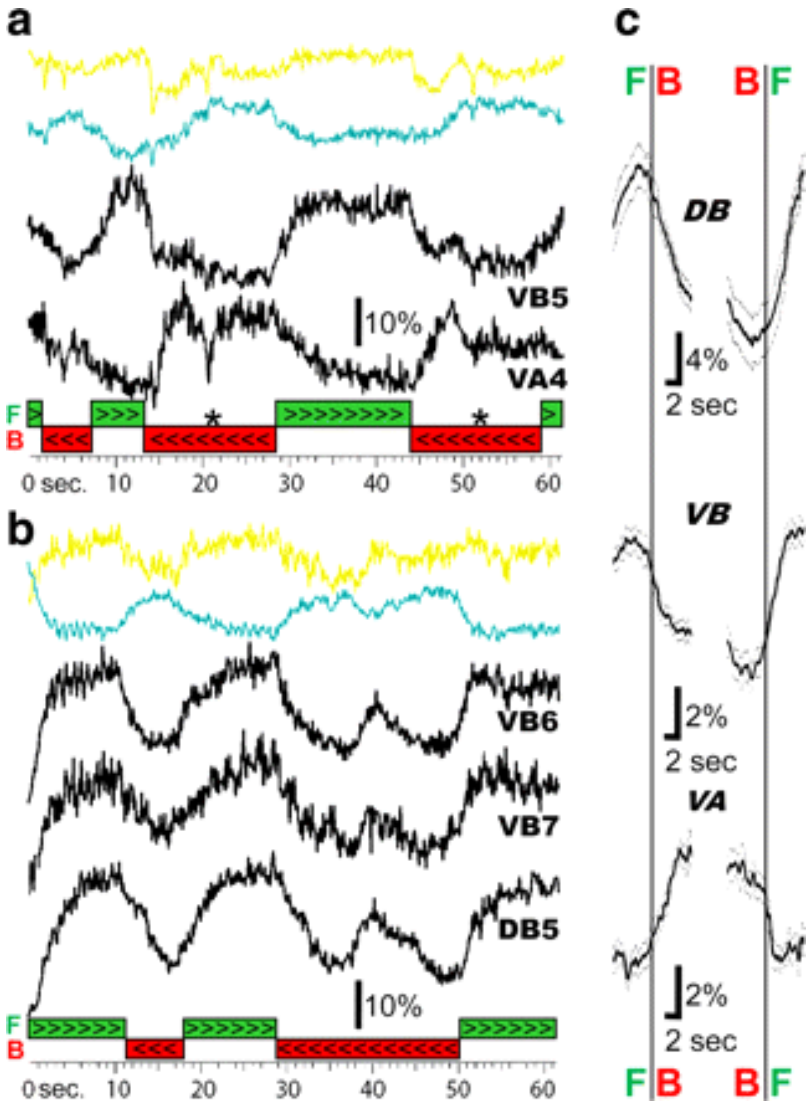

Рис 3. Смена активности VA и VB нейронов и их зависимости.

Моторные нейроны VB и VA не должны быть одновременно активны при движении. Это нам дало только то, что надо смотреть активность VA нейронов.

После анализа активности при стимуляции пищевых нейронов (food sensors) и нейронов носа (nose touch sensors), выяснилось, что в последнем случае повышается частота спайков.

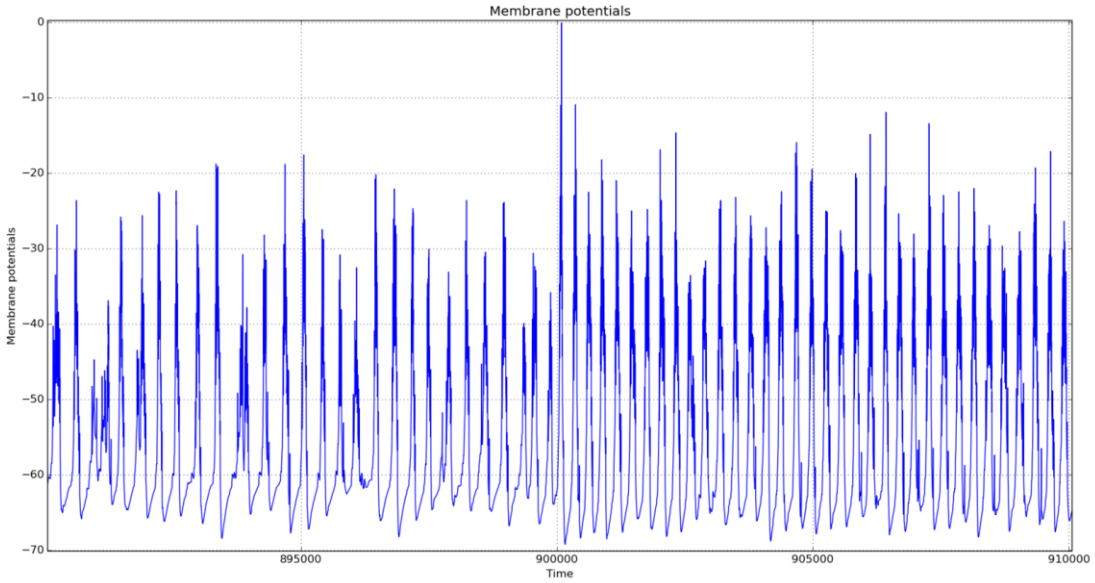

Рис 4. Средняя активность VA нейронов. На отметке 900000 видно, как частота импульсов увеличилась. Это момент начала стимуляции носовых нейронов.

Это позволило найти способ, когда менять направление движения. В общем виде это расчет средней частоты колебания при движении и при препятствии. Если в момент времени средняя частота ближе к движению, значит происходит движение, если ближе к препятствию — соответственно надо менять направление.

2. Практическая часть

Всю практическую часть можно разделить на несколько этапов:

1. Создание и сборка робота на конструкторе lego Mindstorms ev3;
2. Тестирование готовой программы, написанной на языке java;
3. Создание примерной модели робота на бумаге (чертеж);
4. Перенос модели в программу Комппас 3D;
5. Распечатка деталей и сборка макета;
6. Продумывание основных частей программы;
7. Разработка программы под Arduino.

Изначально был сконструирован робот на основе конструктора lego Mindstorms ev3. Программа, написанная на java, позволила выставить количество и типы нейронов. Полностью готовая программа загружалась на флеш-носитель, а результат срабатывания программы загружался напрямую в контроллер.

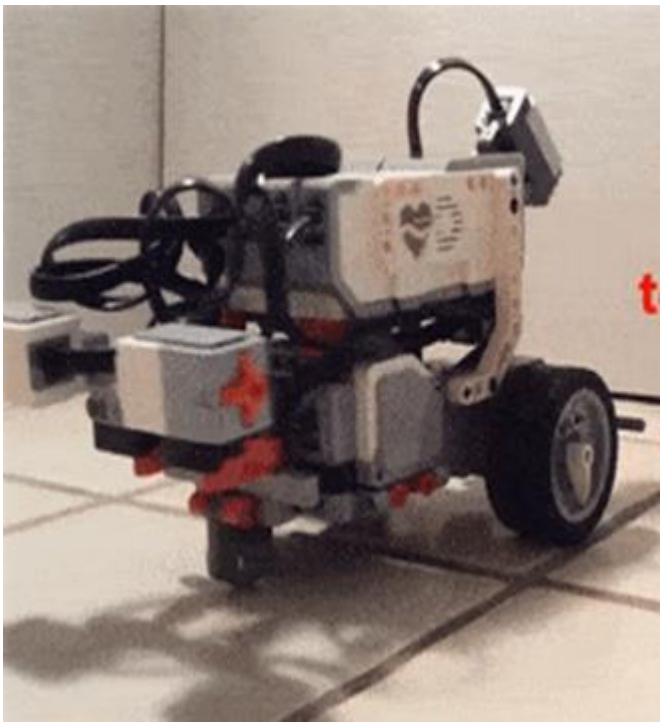

Рис 5. Робот для тестирования программы

Основываясь на реальном поведении c. elegans были рассмотрены следующие варианты активности робота:

- Движение вперед;
- Смена направления движения в случае обнаружения препятствия;

- Повороты налево/направо;
- Обнаружение питания.

Позднее после успешного тестирования робота собранного на основе конструктора lego Mindstorms ev3 было решено реализовать подобное устройство на микроконтроллере Arduino, для которой изначально была разработана 3D-модель (рис 6.).

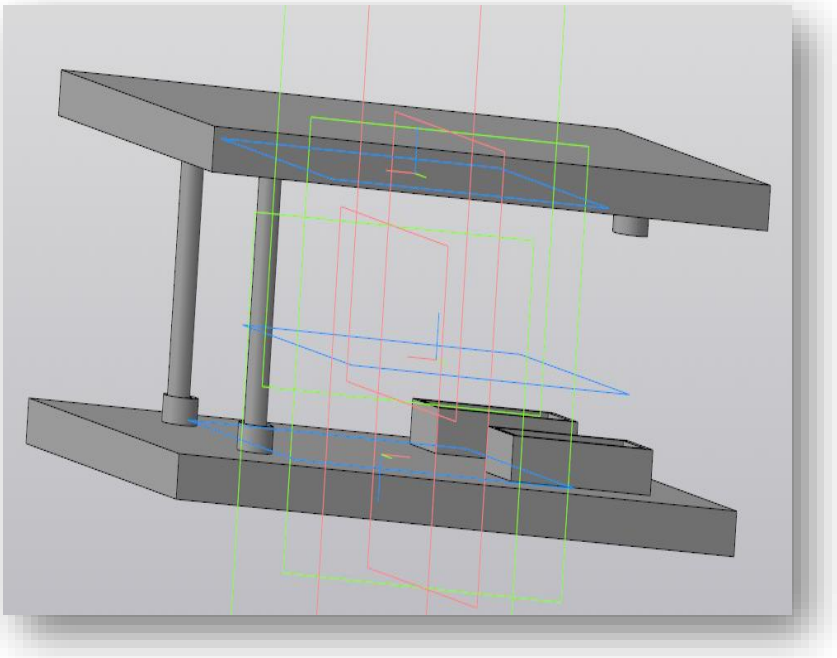

Рис. 6 Фрагмент крепежа роботизированного червя

Пробная версия для более плавного и беспроблемного перехода на другой конструктор была реализована также в виде двухмоторной тележки (рис 7.), финальная часть предполагает практически полную копию кинематики червя и расположения сенсоров (рис 8.).

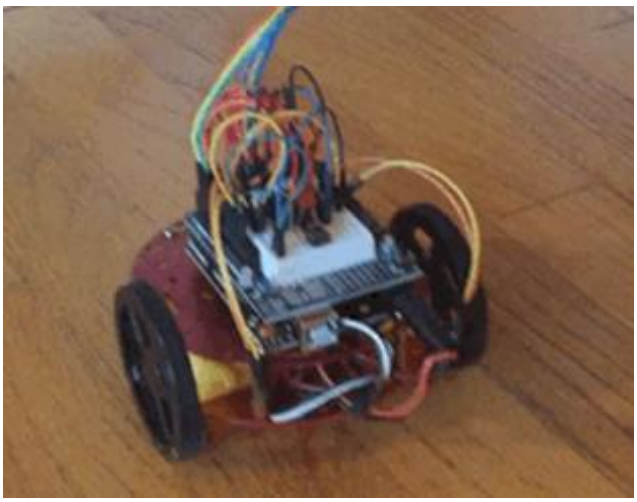

Рис. 7 двухмоторный робот-тележка для отработки алгоритма поведения

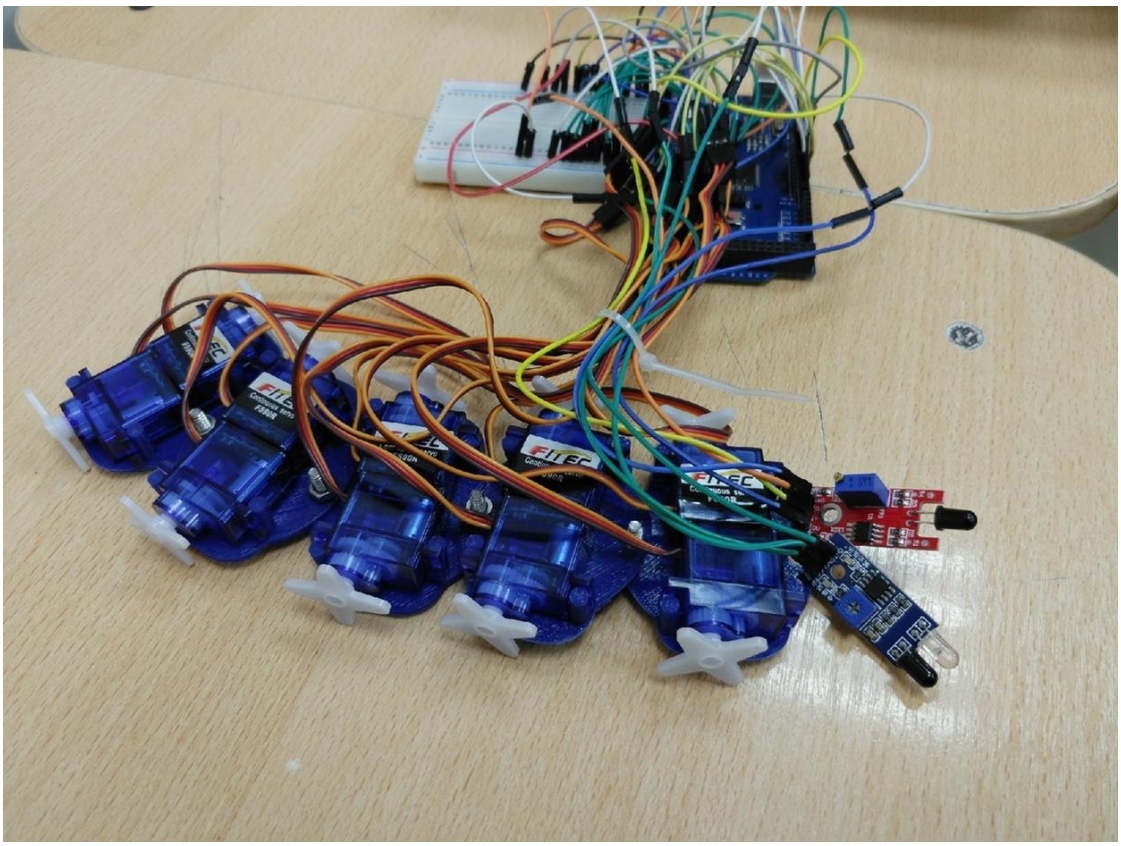

Рис. 8 Робот-нематода

Заключение

В данной работе были проанализированы алгоритмы построения искусственного интеллекта, изучены и подобраны нейросетевые алгоритмы, для написания программы использованы несколько языков программирования Python, Java, C++. При тестировании программы выявлены ошибки, несвойственные настоящему роботу.

В итоге алгоритм нейронной сети, повторяющей работу мозга круглого червя нематоды C. Elegans реализован на роботе, собранном из конструктора Lego mindstorms Ev3, позже была разработана 3D-модель для печати и сборки робота червя на основе микроконтроллера Arduino

Список использованных источников

1. Концепт переноса сознания червя в робота (https://habr.com/ru/post/365443/)
2. Цифровое бессмертие (https://www.youtube.com/watch?time_continue=55&v=A8XXYRvakao)
3. Робот на Arduino с нервной системой червя (http://robocraft.ru/blog/projects/3691.html)
4. Открытая библиотека (https://github.com/nategri/nematoduino)
5. Caenorhabditis elegans (https://en.wikipedia.org/wiki/Caenorhabditis_elegans)

Приложения

```
#ifndef pin_config_h
#define pin_config_h

// Motor pins
const uint8_t rightMotorPinA = 9;
const uint8_t rightMotorPinB = 10;

const uint8_t leftMotorPinA = 3;
const uint8_t leftMotorPinB = 11;

// Button pin (configured so that on is LOW)
const uint8_t buttonPin = 13;

// Transmit and receive pins for distance sensor
const uint8_t tPin = 2;
const uint8_t rPin = 4;

// Pin for status LED (turns on when nose touch neurons stimulated)
const uint8_t statusPin = 12;

#endif
```

Программа инициализации двигателей

```
// Word 0: Number of connected neurons (N_MAX)
// Word 1-(N_MAX): A address (offset from beginning) of each list of connections
// Word N_MAX+1: N_MAX plus the number of connections in the final neuron
// Word > (N_MAX+1): List of connections

// Connection words are formatted as follows:
// Bit 0-7: Lower part of neuron index
// Bit 8: High part of neuron index (9-bits reserved for index)
// Bit 9-15: Weight of connection

const uint16_t PROGMEM NeuralROM[] = {
0x012c, 0x012e, 0x0141, 0x0152, 0x016f, 0x0183, 0x018b, 0x0199, 0x01ab, 0x01bb,
0x0202, 0x0213, 0x0220, 0x0229, 0x0236, 0x023f, 0x0247, 0x0256, 0x0265, 0x026b,
0x02a4, 0x02aa, 0x02b5, 0x02be, 0x02ca, 0x02d1, 0x02da, 0x02e4, 0x02ec, 0x02f2,
0x0327, 0x0338, 0x0342, 0x034a, 0x034f, 0x0353, 0x0359, 0x0362, 0x036a, 0x0373,
0x0463, 0x0478, 0x0490, 0x04a1, 0x04b0, 0x04c3, 0x04d5, 0x04e5, 0x04f6, 0x0502,
0x0569, 0x0572, 0x057e, 0x0587, 0x058e, 0x0599, 0x05a1, 0x05a9, 0x05b4, 0x05c6,
0x0622, 0x0631, 0x063d, 0x064c, 0x0653, 0x065b, 0x066b, 0x067f, 0x0690, 0x06a4,
0x06e8, 0x06f0, 0x06f9, 0x0702, 0x070c, 0x0731, 0x073c, 0x0750, 0x075d, 0x076c,
0x07a8, 0x07ab, 0x07af, 0x07b4, 0x07bb, 0x07c8, 0x07d0, 0x07de, 0x07ef, 0x07f7,
0x0836, 0x0840, 0x084a, 0x0854, 0x0858, 0x085e, 0x0865, 0x0871, 0x087e, 0x0887,
0x08bb, 0x08cb, 0x08db, 0x08e3, 0x08ee, 0x08fb, 0x0907, 0x090d, 0x0912, 0x091a,
0x094e, 0x0955, 0x0959, 0x0966, 0x0968, 0x096a, 0x0972, 0x0994, 0x09ba, 0x09c0,
0x0a26, 0x0a2d, 0x0a3e, 0x0a58, 0x0a68, 0x0a6d, 0x0a72, 0x0a81, 0x0a93, 0x0aab,
0x0b02, 0x0b18, 0x0b28, 0x0b41, 0x0b55, 0x0b6e, 0x0b71, 0x0b74, 0x0b83, 0x0b9b,
0x0bf1, 0x0c02, 0x0c18, 0x0c23, 0x0c29, 0x0c2d, 0x0c3a, 0x0c44, 0x0c47, 0x0c61,
0x0c9b, 0x0ca1, 0x0ca5, 0x0ca8, 0x0cab, 0x0cb4, 0x0cc1, 0x0cc2, 0x0cc3, 0x0cc4,
0x0cdf, 0x0cec, 0x0cf7, 0x0d02, 0x0d14, 0x0d21, 0x0d31, 0x0d3e, 0x0d44, 0x0d4d,
```

Программа принятия решений

```
#include "defines.h"

const uint16_t PROGMEM LeftNeckMuscles[] = {
N_MDL05, N_MDL06, N_MDL07, N_MDL08, N_MVL05, N_MVL06, N_MVL07, N_MVL08
};

const uint16_t PROGMEM RightNeckMuscles[] = {
N_MDR05, N_MDR06, N_MDR07, N_MDR08, N_MVR05, N_MVR06, N_MVR07, N_MVR08
};

const uint16_t PROGMEM LeftBodyMuscles[] = {
N_MDL09, N_MDL10, N_MDL11, N_MDL12, N_MDL13, N_MDL14, N_MDL15,
N_MDL16, N_MDL17, N_MDL18, N_MDL19, N_MDL20, N_MDL21, N_MDL22, N_MDL23,
N_MVL09, N_MVL10, N_MVL11, N_MVL12, N_MVL13, N_MVL14, N_MVL15,
N_MVL16, N_MVL17, N_MVL18, N_MVL19, N_MVL20, N_MVL21, N_MVL22, N_MVL23
};

const uint16_t PROGMEM RightBodyMuscles[] = {
N_MDR09, N_MDR10, N_MDR11, N_MDR12, N_MDR13, N_MDR14, N_MDR15,
N_MDR16, N_MDR17, N_MDR18, N_MDR19, N_MDR20, N_MDR21, N_MDR22, N_MDR23,
N_MVR09, N_MVR10, N_MVR11, N_MVR12, N_MVR13, N_MVR14, N_MVR15,
N_MVR16, N_MVR17, N_MVR18, N_MVR19, N_MVR20, N_MVR21, N_MVR22, N_MVR23
};

/*
const uint16_t PROGMEM MotorNeuronsB[] = {
N_DB1, N_DB2, N_DB3, N_DB4, N_DB5, N_DB6, N_DB7, N_VB1, N_VB2, N_VB3, N_VB4,
N_VB5, N_VB6, N_VB7, N_VB8, N_VB9, N_VB10, N_VB11
};

const uint16_t PROGMEM MotorNeuronsA[] = {
N_DA1, N_DA2, N_DA3, N_DA4, N_DA5, N_DA6, N_DA7, N_DA8, N_DA9,
N_VA1, N_VA2, N_VA3, N_VA4, N_VA5, N_VA6, N_VA7, N_VA8, N_VA9, N_VA10, N_VA11,
N_VA12
};
*/

// Significant motor neurons (e.g. ones that are good indicators for locomotion
const uint16_t PROGMEM SigMotorNeuronsB[] = {
N_VB2, N_VB3, N_VB4, N_VB5, N_VB6
```

Программа обработки данных с датчика препятствия

```
#include "pin_config.h"

// Initialize button
// And loop until pressed

void ButtonInit() {
  pinMode(buttonPin, INPUT_PULLUP);
}

bool ButtonPress() {
  if(digitalRead(buttonPin) == LOW) {
    return true;
  }
  else {
    return false;
  }
}
```

Программа обработки данных с датчика касания

```
#include "pin_config.h"

void SensorInit() {
  pinMode(tPin, OUTPUT);
  pinMode(rPin, INPUT);
}

long SensorDistance() {

  // Send sound pulse
  digitalWrite(tPin, LOW);
  delayMicroseconds(2);
  digitalWrite(tPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(tPin, LOW);

  // Read off length
  long pulseWidth = pulseIn(rPin, HIGH);

  // Calculate distance
  long distance = (pulseWidth/29.1) / 2;

  return distance;
}
```

Программа обработки данных с датчика звука

```
#include "pin_config.h"

const uint8_t spdBoost = 0;
const uint8_t turnTime = 0;

void MotorsInit() {
  pinMode(leftMotorPinA, OUTPUT);
  pinMode(leftMotorPinB, OUTPUT);
  pinMode(rightMotorPinA, OUTPUT);
  pinMode(rightMotorPinB, OUTPUT);
}

void rightMotorForward(uint8_t spd) {
  analogWrite(rightMotorPinA, spd);
  digitalWrite(rightMotorPinB, LOW);
}

void rightMotorBackward(uint8_t spd) {
  digitalWrite(rightMotorPinA, LOW);
  analogWrite(rightMotorPinB, spd);
}

void rightMotorOff() {
  digitalWrite(rightMotorPinA, LOW);
  digitalWrite(rightMotorPinB, LOW);
}

void leftMotorForward(uint8_t spd) {
  analogWrite(leftMotorPinA, spd);
  digitalWrite(leftMotorPinB, LOW);
}

void leftMotorBackward(uint8_t spd) {
  digitalWrite(leftMotorPinA, LOW);
  analogWrite(leftMotorPinB, spd);
}

void leftMotorOff() {
  digitalWrite(leftMotorPinA, LOW);
  digitalWrite(leftMotorPinB, LOW);
}

void RunMotors(int16_t leftSpd, int16_t rightSpd) {
  uint8_t leftSpdMotor;
  uint8_t rightSpdMotor;

  // Left forward
  if(leftSpd >= 0) {
    leftSpd += spdBoost;
    if(leftSpd > 255) {
      leftSpdMotor = 255;
    }
    else {
      leftSpdMotor = leftSpd;
    }
    leftMotorForward(leftSpdMotor);
  }
  // Left backward
  else if(leftSpd < 0) {
    leftSpd -= spdBoost;
```

Код для передвижения робота