ИНТЕЛЛЕКТУАЛЬНЫЙ МЕГАПОЛИС

ЗАДАЧНИК

ИТ-класс
в московской школе

НАПРАВЛЕНИЕ РОБОТОТЕХНИКА

ПРАКТИЧЕСКИЙ ЭТАП

МОСКВА
2025

ИНТЕЛЛЕКТУАЛЬНЫЙ МЕГАПОЛИС

ЗАДАЧНИК РАЗРАБОТАН:

Архипов Максим Викторович, к.т.н., доцент кафедры «Автоматика и управление», Федеральное государственное автономное образовательное учреждение высшего образования "Московский политехнический университет" (Московский Политех)

Малкаров Артем Юрьевич, преподаватель кафедры «Автоматика и управление», Федеральное государственное автономное образовательное учреждение высшего образования "Московский политехнический университет" (Московский Политех)

МОСКВА
2025

Оглавление

Решения заданий ИМ, IT класс, робототехника, практика, кейс 1, вариантов 14

Решения заданий ИМ, IT класс, робототехника, практика, кейс 1

Кейс 1. Вариант 1.

Счетчик нажатий кнопки.

Необходимо написать программу подсчета нажатий на кнопку (без ложных срабатываний, называемых дребезгом контактов). Для решения задачи потребуется программно устранить дребезг контактов кнопки и осуществлять вывод данных о количестве нажатий в окно монитора последовательного порта. *Дополнительно* сумму количества нажатий подсчитывается с момента запуска программы и выводится при каждом новом нажатии в окно монитора порта. Для обнуления счетчика нажатий требуется перезапуск программы.

Справочно. Дребезг контактов - явление, происходящее в электромеханических коммутационных устройствах и аппаратах (кнопке), длящееся некоторое время после замыкания электрических контактов. После замыкания происходят многократные неконтролируемые замыкания и размыкания контактов кнопки за счет упругости материалов и деталей контактной системы — некоторое время контакты отскакивают друг от друга при соударениях, размыкая и замыкая электрическую цепь. При неконтролируемом замыкании кнопки при однократном нажатии возможно появление серии ложных сигналов (срабатываний), которые устраняются программным способом (введением опроса с отсрочкой по времени).

Оборудование:

- 1 кнопка
- Соединительные провода
- Резистор 9-11 кОм.
- Arduino UNO и кабель питания USB
- Макетная плата

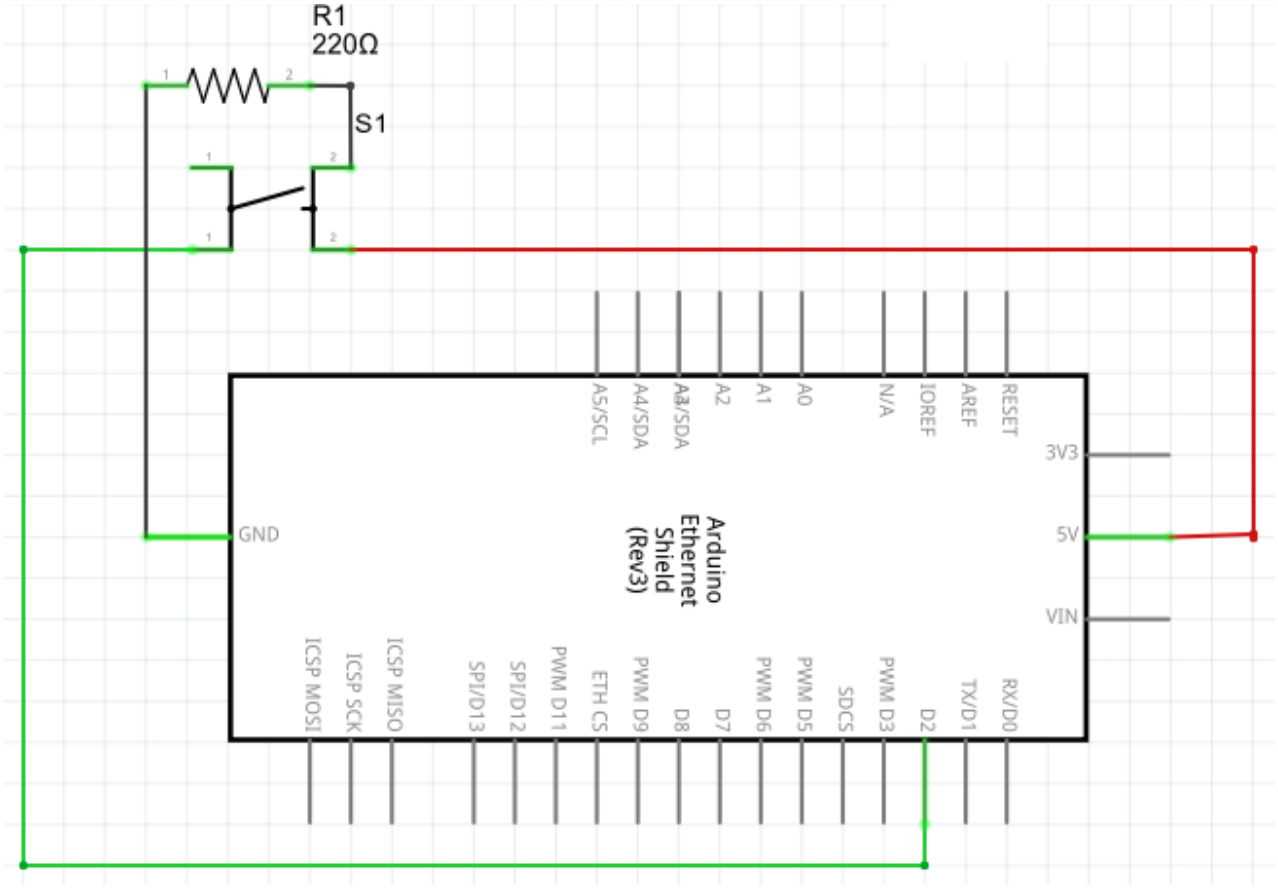
Решение.

Методические рекомендации к задаче и ответу.

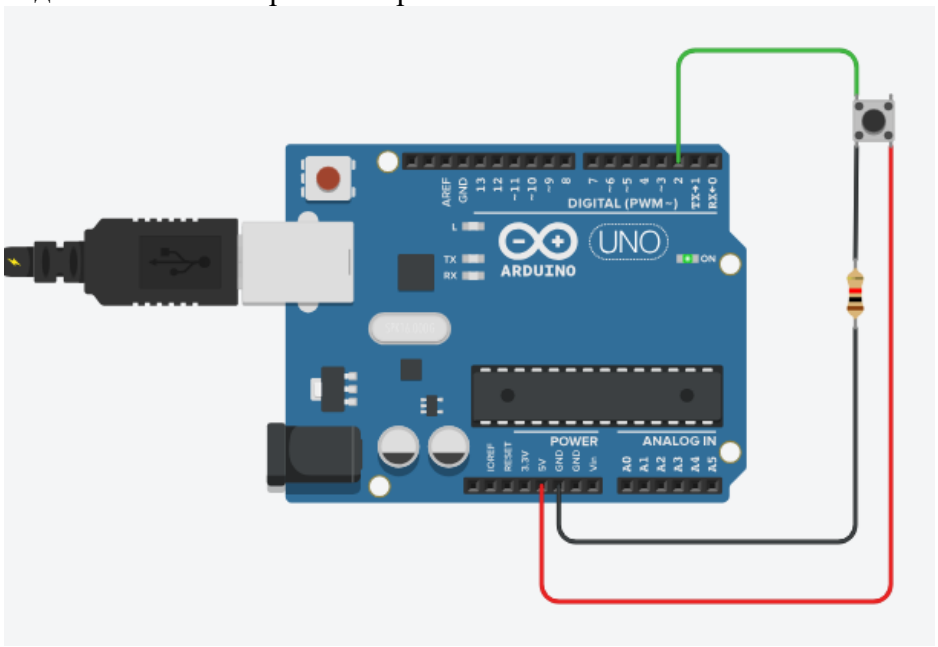
Условие. При проверке решения задачи проверяются знания метода временных интервалах проверки срабатывания кнопочного выключателя. Оценивается умение сравнивать сигналы в нажатом состоянии и отпущенной кнопки через некоторое время.

Ответ. Ответ должен содержать программный код, в котором фигурирует временной интервал отсрочки проверки срабатывания кнопочного выключателя.

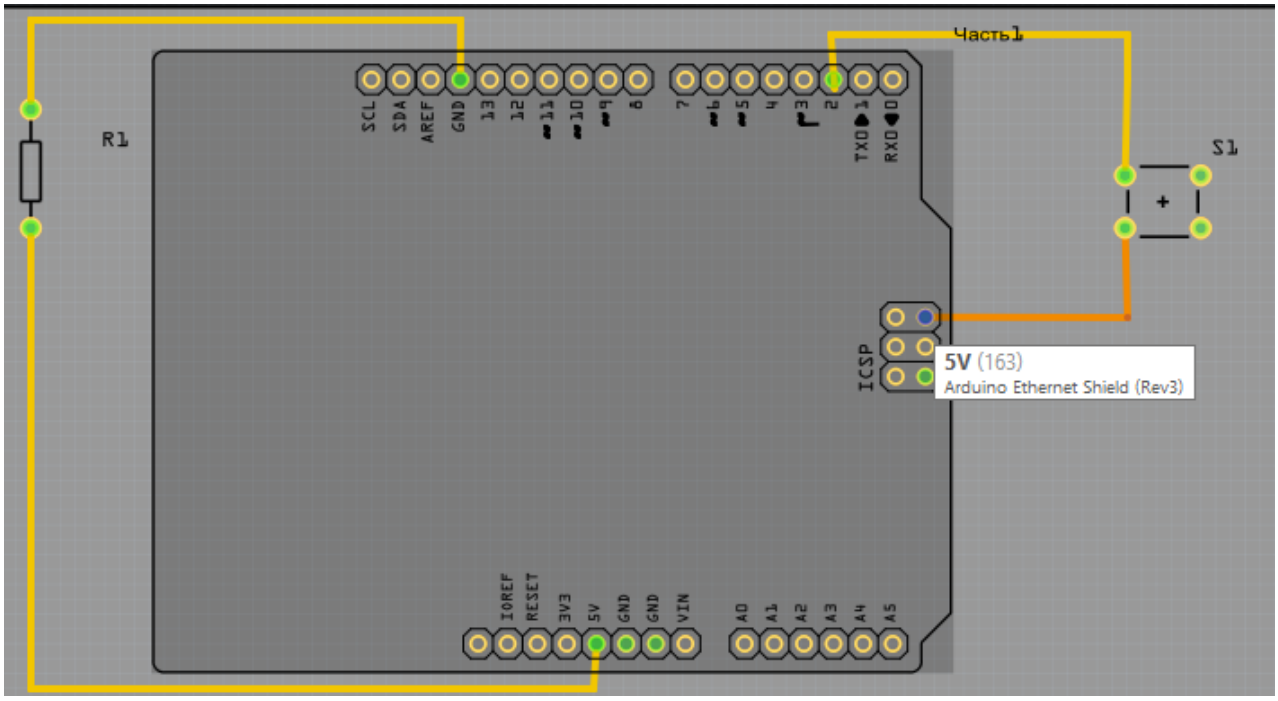
Задание № 1. Электрическая схема



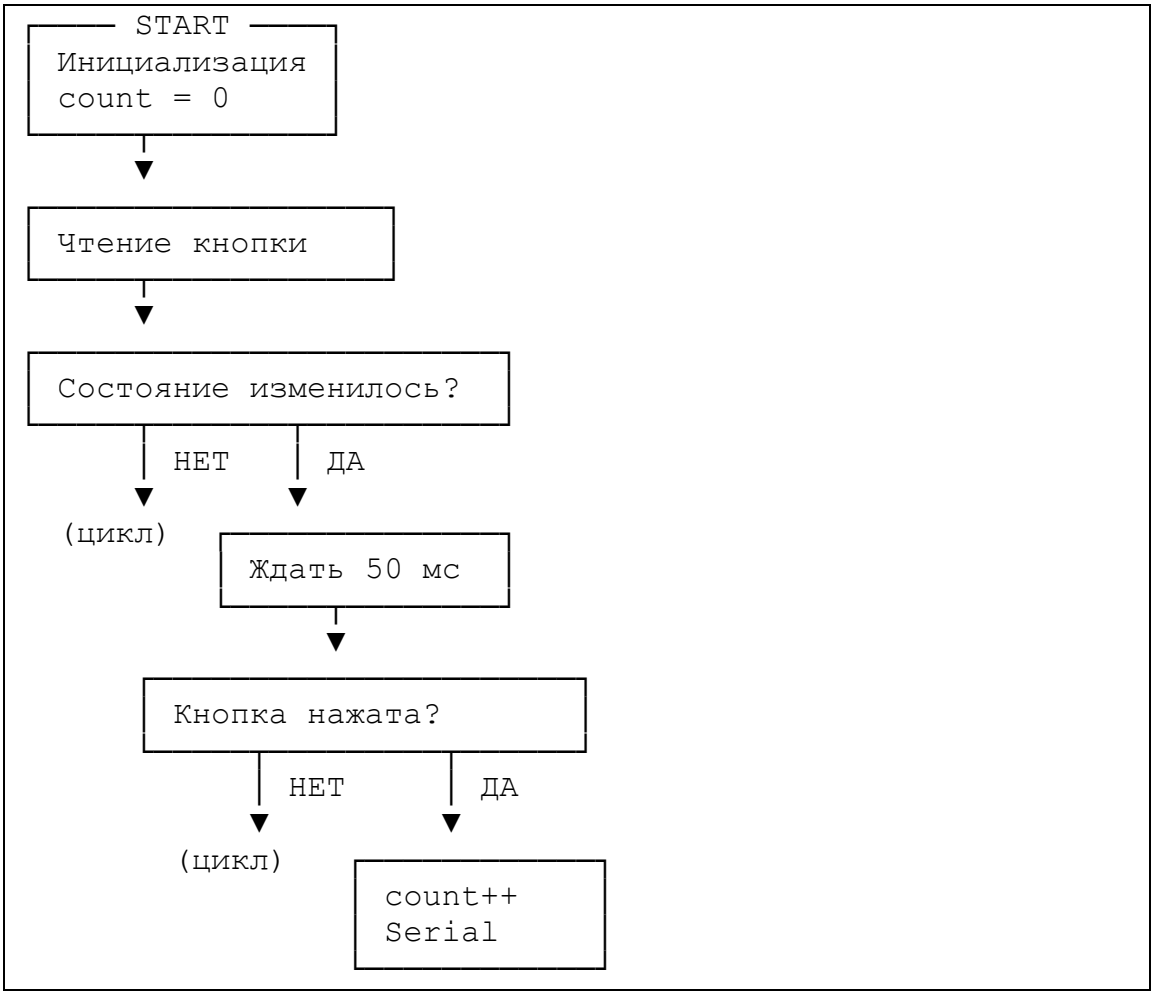
Задание № 2.1. Сборка электрической схемы



Задание № 2.2. Проектирование электрической схемы



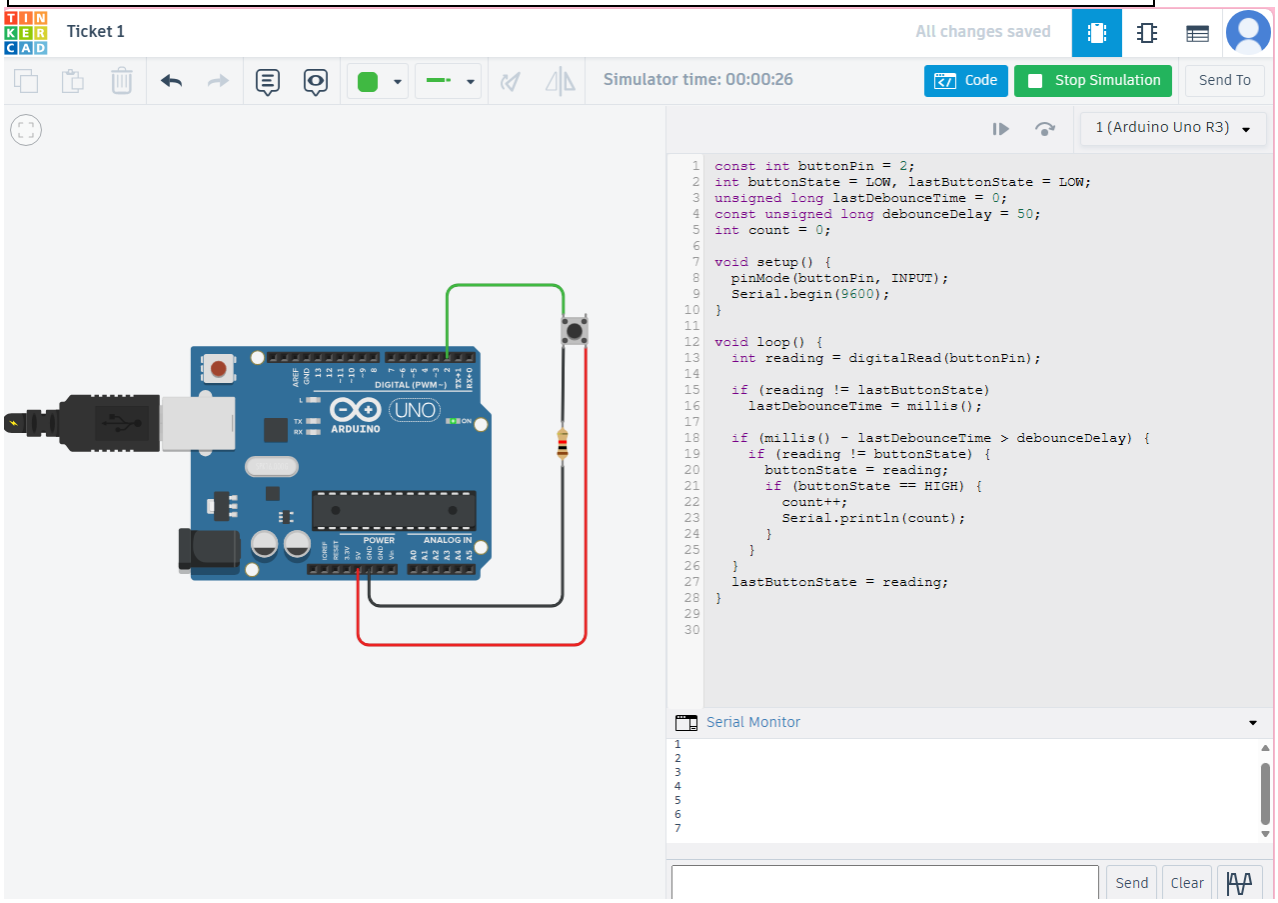
Задание № 3. Блок-схема алгоритма



Задание № 4.1. Управляющая программа и 4.2. Монитор последовательного порта

```
const int buttonPin = 2;
int buttonState = LOW, lastButtonState = LOW;
unsigned long lastDebounceTime = 0;
const unsigned long debounceDelay = 50;
int count = 0;
```

```
void setup() {
  pinMode(buttonPin, INPUT);
  Serial.begin(9600);
}

void loop() {
  int reading = digitalRead(buttonPin);

  if (reading != lastButtonState)
    lastDebounceTime = millis();

  if (millis() - lastDebounceTime > debounceDelay) {
    if (reading != buttonState) {
      buttonState = reading;
      if (buttonState == HIGH) {
        count++;
        Serial.println(count);
      }
    }
  }
  lastButtonState = reading;
}
```



Кейс 1. Вариант 2.

Управление освещением через пульт.

Необходимо написать программу управления освещением, с пульта дистанционного управления, четырьмя светодиодами по отдельности, при помощи клавиш (цифры 1-4). Под управлением подразумевается то, что светодиоды должны включаться и выключаться по сигналам возникающим при нажатии на определенные клавиши на пульте. При первом нажатии и отпускании одной из клавиш – светодиод включается, при повторном нажатии и

отпускании той же клавиши, тот же светодиод выключается. *Дополнительно* в окно монитора порта выводить информация о текущей нажатой клавише.

Оборудование:

- ИК приёмник
- ИК пульт
- Использование библиотеки IRremote.hpp
- Светодиоды разноцветные (4 штуки)
- Резисторы 200-300 Ом (4 штуки)
- Соединительные провода
- Arduino UNO и кабель питания USB
- Макетная плата

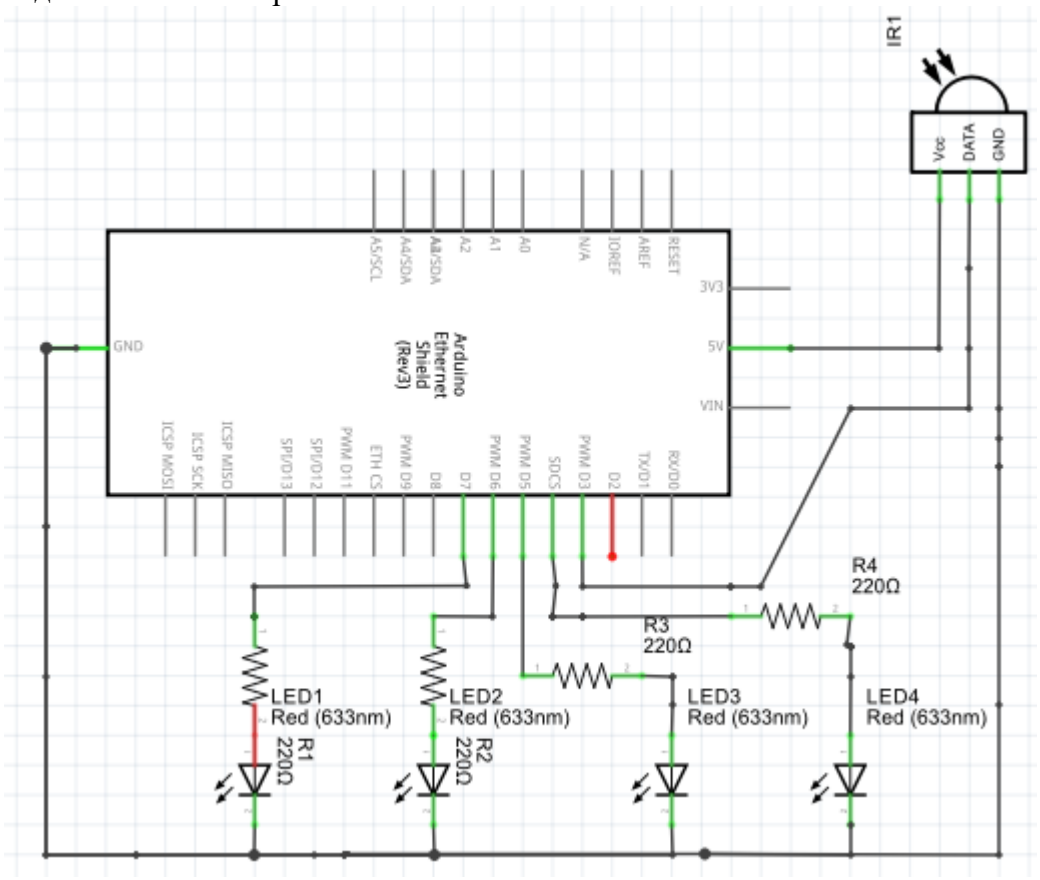
Решение.

Методические рекомендации к задаче и ответу.

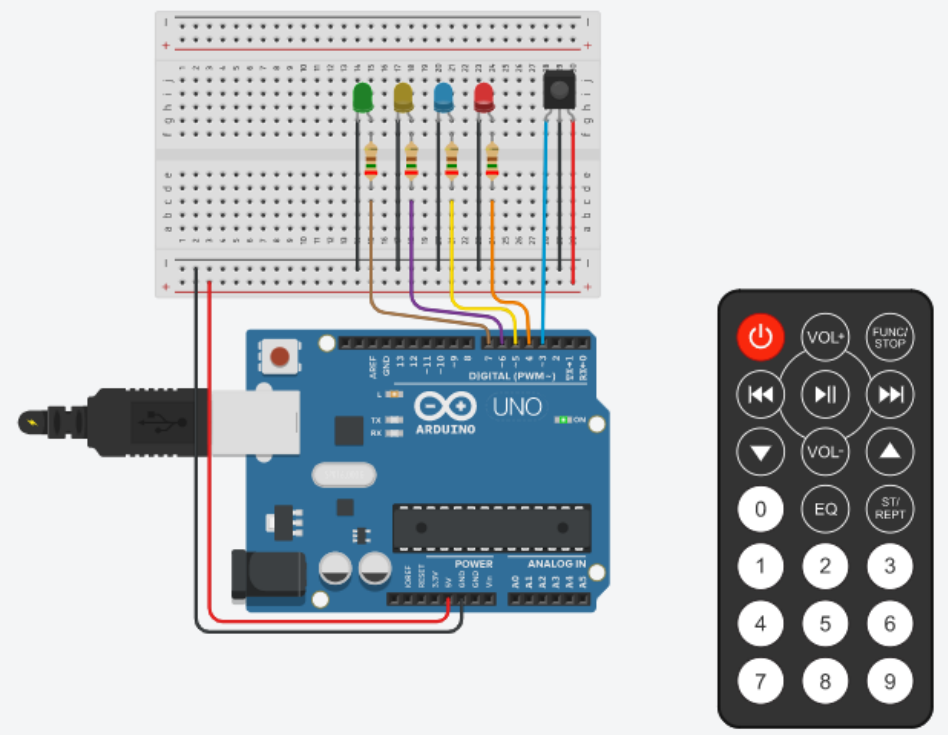
Условие. При проверке решения задачи проверяются знания метода введения дополнительных переменных и буферного хранения данных. Оценивается умение менять содержимое переменных местами.

Ответ. Ответ должен содержать переменные хранящие текущею и предыдущую информацию.

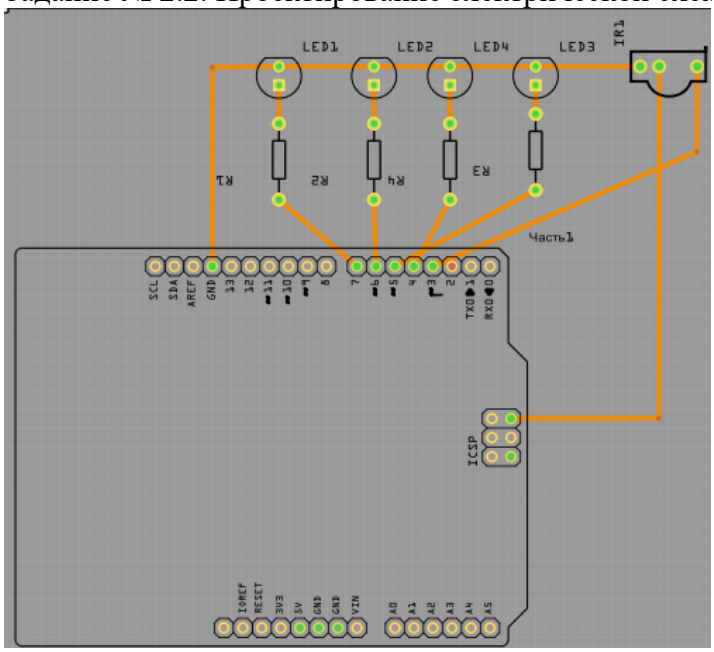
Задание № 1. Электрическая схема



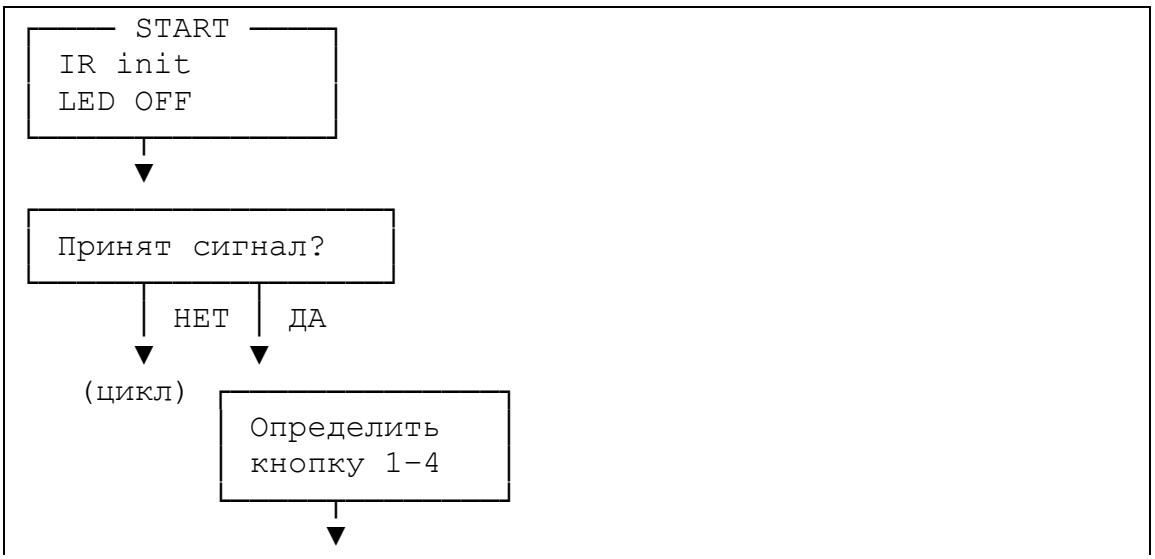
Задание № 2.1. Сборка электрической схемы

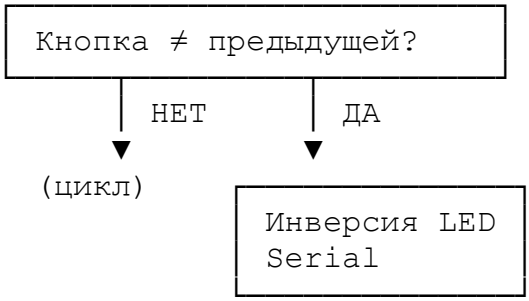


Задание № 2.2. Проектирование электрической схемы



Задание № 3. Блок-схема алгоритма

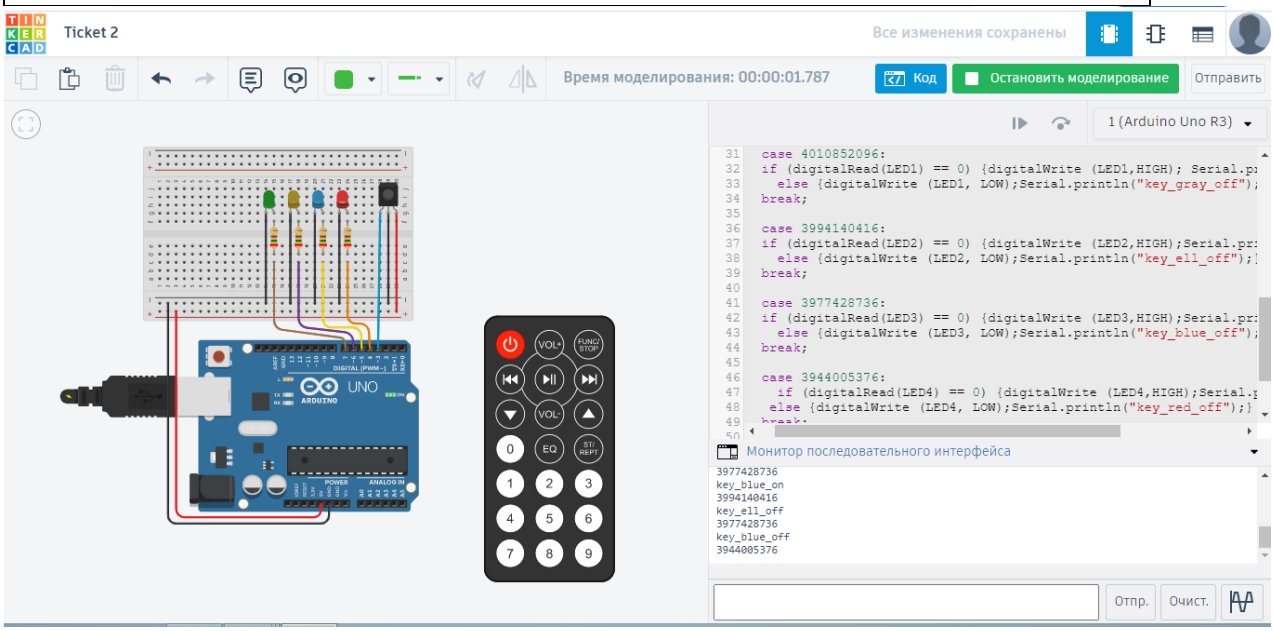




Задание № 4.1. Управляющая программа и 4.2. Монитор последовательного порта

```
#include "IRremote.hpp"
#define IR_RECEIVE_PIN 3
#define LED1 7
#define LED2 6
#define LED3 5
#define LED4 4

/* int regist;
IRrecv irrecv(3); // указываем вывод, к которому подключен
приемник
*/

void setup() {
  pinMode(LED1, OUTPUT);
  pinMode(LED2, OUTPUT);
  pinMode(LED3, OUTPUT);
  pinMode(LED4, OUTPUT);
  digitalWrite(LED1, LOW);
  digitalWrite(LED2, LOW);
  digitalWrite(LED3, LOW);
  digitalWrite(LED4, LOW);

  Serial.begin(9600); // выставляем скорость COM порта
  IrReceiver.begin(IR_RECEIVE_PIN, ENABLE_LED_FEEDBACK);
// запускаем прием
}

void loop() {
  if ( IrReceiver.decode( )) { // если данные пришли

Serial.println(IrReceiver.decodedIRData.decodedRawData);
// печатаем данные

    switch(IrReceiver.decodedIRData.decodedRawData){
    case 4010852096:
    if (digitalRead(LED1) == 0) {digitalWrite (LED1,HIGH);
Serial.println("key_gray_on");}
      else {digitalWrite (LED1,
LOW);Serial.println("key_gray_off");}
    break;

    case 3994140416:
    if (digitalRead(LED2) == 0) {digitalWrite
(LED2,HIGH);Serial.println("key_ell_on");}
```

```
      else {digitalWrite (LED2,
LOW);Serial.println("key_ell_off");}
    break;

    case 3977428736:
    if (digitalRead(LED3) == 0) {digitalWrite
(LED3,HIGH);Serial.println("key_blue_on");}
      else {digitalWrite (LED3,
LOW);Serial.println("key_blue_off");}
    break;

    case 3944005376:
      if (digitalRead(LED4) == 0) {digitalWrite
(LED4,HIGH);Serial.println("key_red_on");}
     else {digitalWrite (LED4,
LOW);Serial.println("key_red_off");}
    break;
    }
    IrReceiver.resume(); // принимаем следующую команду
    }
}
```



Кейс 1. Вариант 3.

Таймер приготовления блюд.

Для таймера по приготовлению блюд, содержащего 3 светодиода подключенных к цифровым выходам Arduino, необходимо написать программу включения светодиодов в зависимости от времени приготовления двух видов блюд: блюдо А, блюдо В. При полной готовности блюд загорается зеленый светодиод, по времени полная готовность блюд наступает: блюдо А - 10 сек., блюдо В - 15 сек. При промежуточной готовности начинает мигать желтый сигнал (мигание 1 раз в 1 секунду, время свечения 0,5 сек). По времени промежуточная готовность наступает: блюдо А - с 4 по 5 секунду, блюдо В - с 6 по 7 секунду. До наступления промежуточной готовности горит красный сигнал. В программе необходимо указать переменную для наименования приготавливаемого блюда. После запуска программы начинается отсчет таймера и активируется работа красного светодиода. *Дополнительно* реализуйте ввод названия блюда методом отправки букв А или В через окно монитора порта.

Оборудование:

- Резистор 200-330 Ом 3 шт.
- Arduino UNO и кабель питания USB
- Макетная плата
- Светодиоды: красный (1 шт.), зеленый (1 шт.), желтый (1 шт.).
- Соединительные провода
- Ссылка на схему в TinkerCad

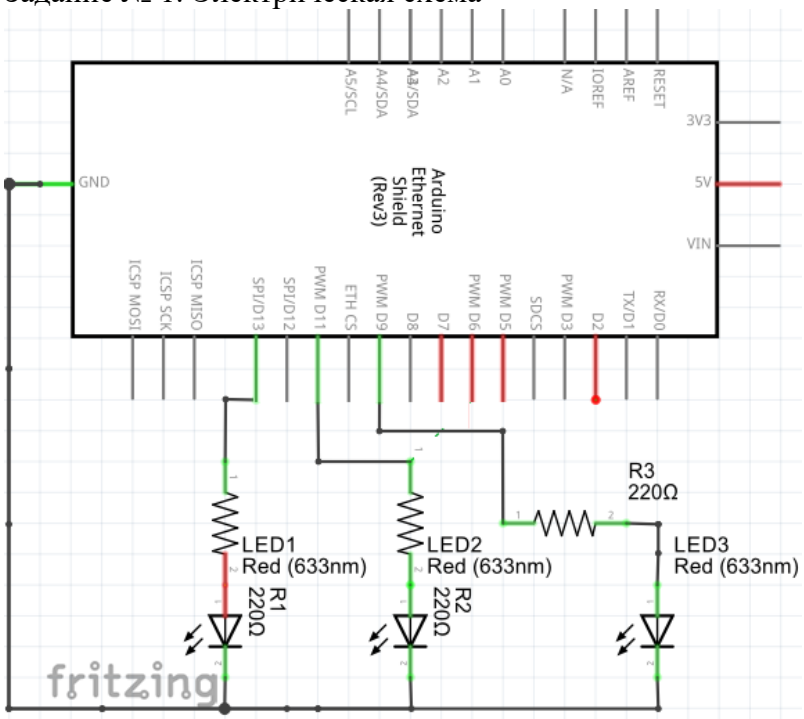
Решение.

Методические рекомендации к задаче и ответу.

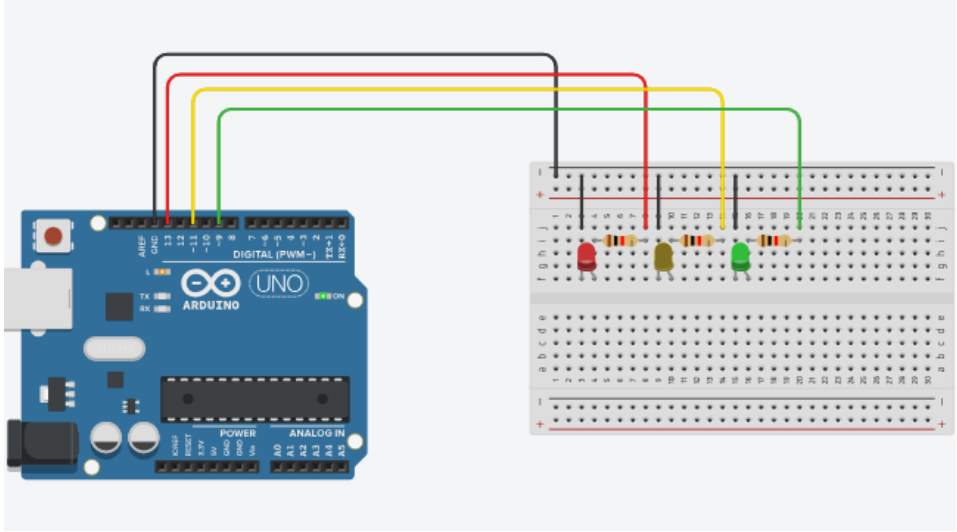
Условие. При проверке решения задачи проверяются знания метода управления задержками. Оценивается умение планировать процедурную работу программы.

Ответ. Ответ должен содержать программный код, в котором фигурируют правильные временные задержки.

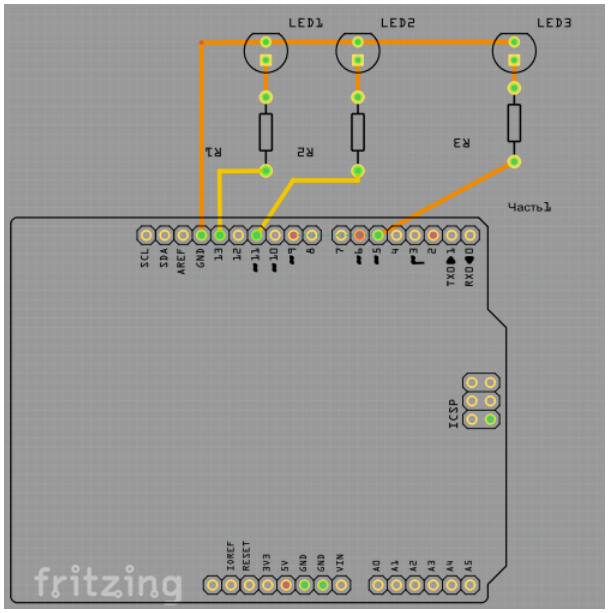
Задание № 1. Электрическая схема



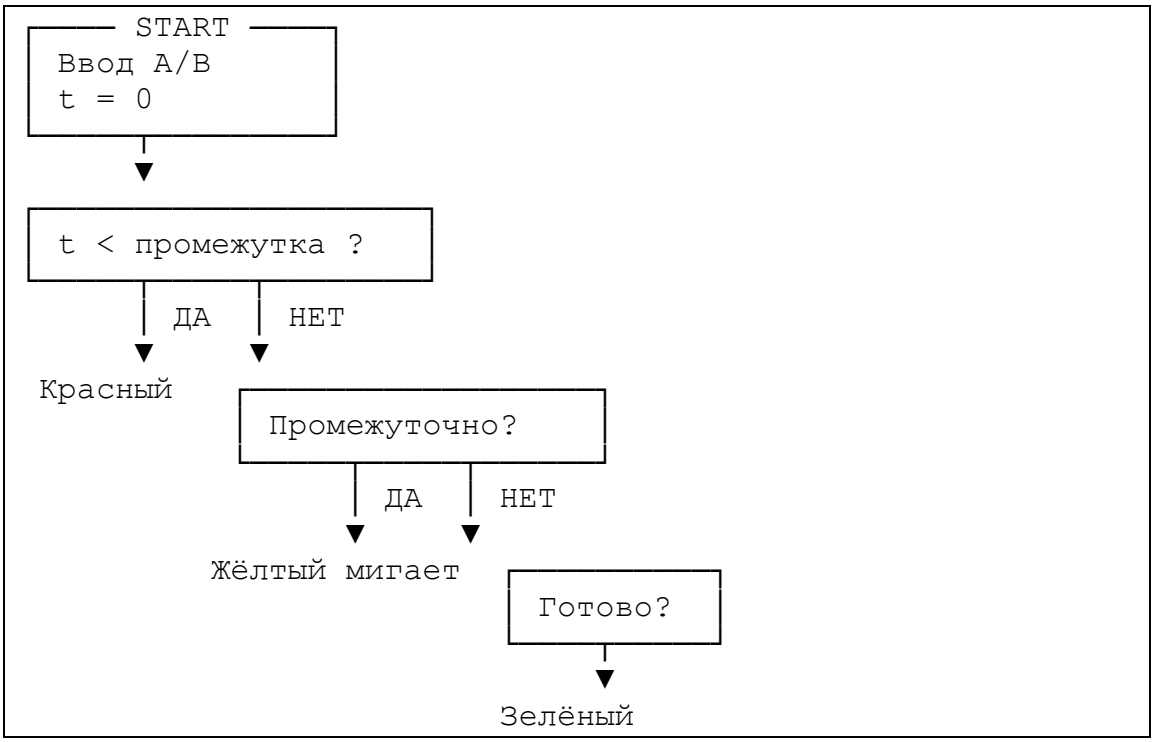
Задание № 2.1. Сборка электрической схемы



Задание № 2.2. Проектирование электрической схемы



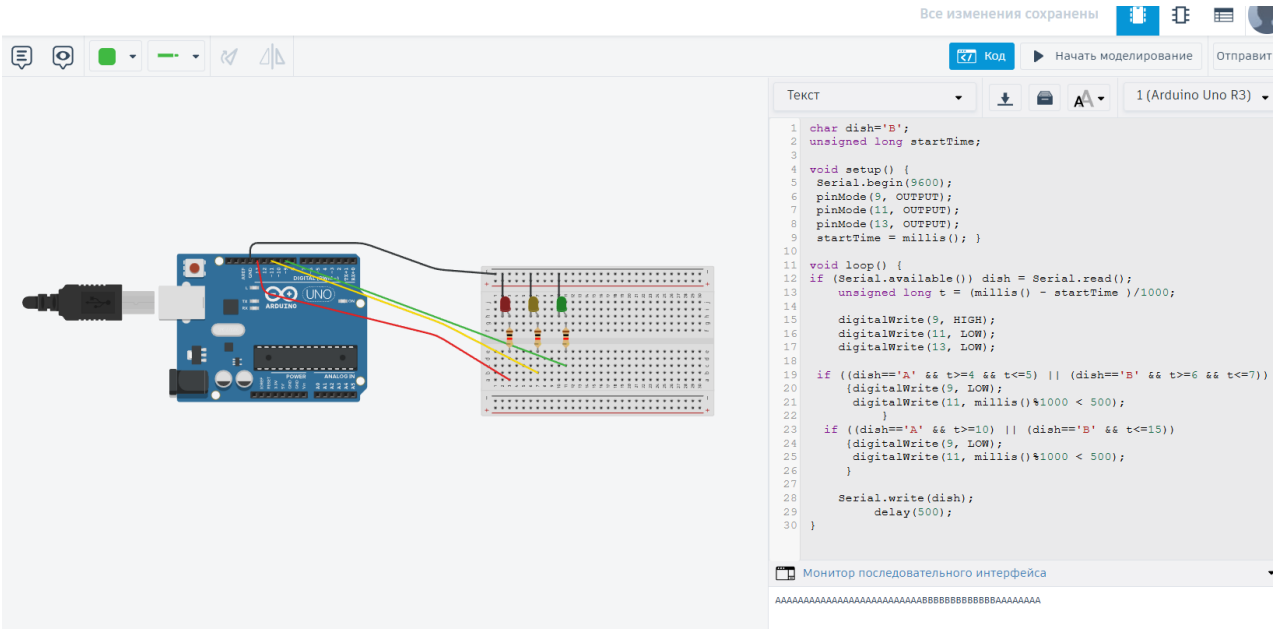
Задание № 3. Блок-схема алгоритма



Задание № 4.1. Управляющая программа и 4.2. Монитор последовательного порта

```
char dish = 'A';
unsigned long startTime;

void setup() {
  Serial.begin(9600);
  pinMode(3,OUTPUT);
  pinMode(4,OUTPUT);
  pinMode(5,OUTPUT);
  startTime = millis();
}

void loop() {
  if (Serial.available()) dish = Serial.read();
  unsigned long t = (millis() - startTime) / 1000;
```

```
  digitalWrite(3, HIGH);
  digitalWrite(4, LOW);
  digitalWrite(5, LOW);

  if ((dish=='A' && t>=4 && t<=5) || (dish=='B' && t>=6 &&
t<=7)) {
    digitalWrite(3, LOW);
    digitalWrite(4, millis()%1000 < 500);
  }

  if ((dish=='A' && t>=10) || (dish=='B' && t>=15)) {
    digitalWrite(3, LOW);
    digitalWrite(4, LOW);
    digitalWrite(5, HIGH);
  }
}
```



Кейс 1. Вариант 4.

Разноцветный ночник.

Напишите программу управления свечением ночника, в зависимости от значений освещенности, считываемой фоторезистором. Калибровку фоторезистора проведите используя команду analogRead(). При ярком освещении RGB-светодиод не горит, при затенении - горит голубым цветом. *Дополнительно* реализуйте вывод информации о включении-выключении освещенности светодиода через окно монитора порта.

Справочно. Для отображения бледно-голубым цветом необходимо задать цвета: R: 144, G: 195, B: 212.

Оборудование:

- Резистор 200-330 Ом 4 шт.
- Arduino UNO и кабель UBS
- Макетная плата
- Светодиоды: RGB (1 шт.).
- Соединительные провода
- Фоторезистор

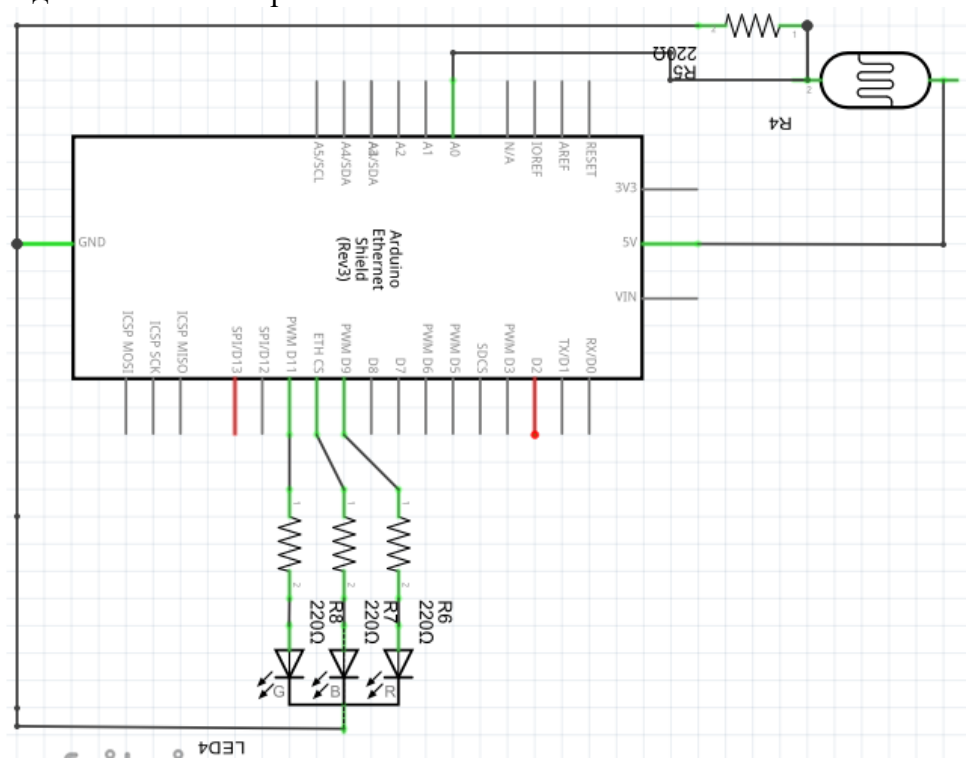
Решение.

Методические рекомендации к задаче и ответу.

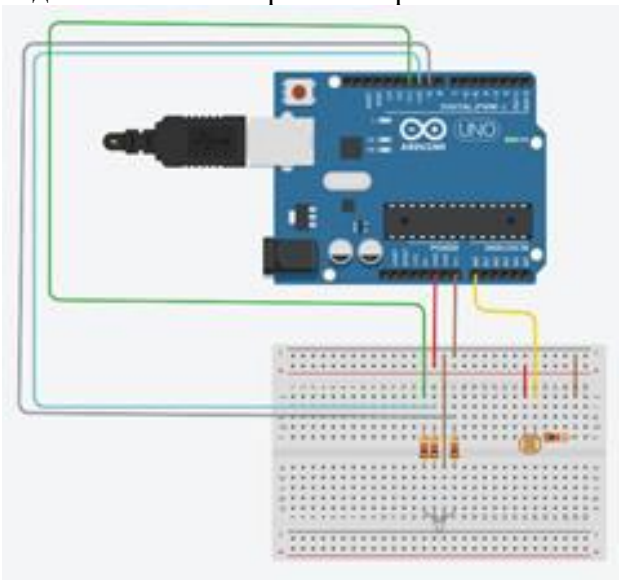
Условие. При проверке решения задачи проверяются знания метода опроса аналогового сигнала. Оценивается умение обрабатывать аналоговый сигнал.

Ответ. Ответ должен содержать программный код, в котором имеется считывание аналогового сигнала.

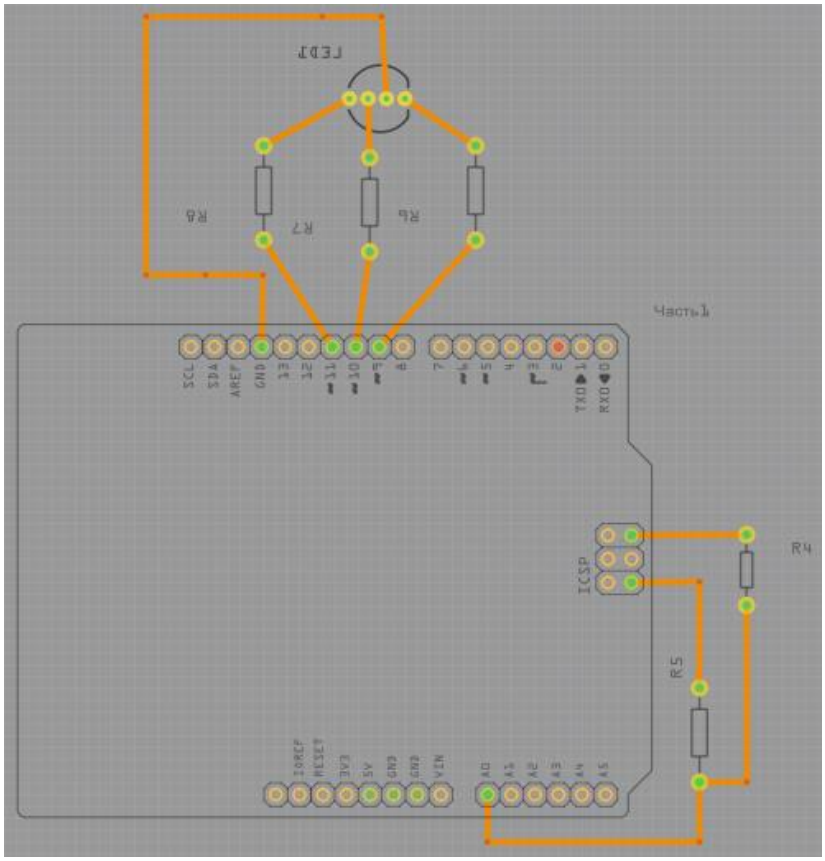
Задание № 1. Электрическая схема



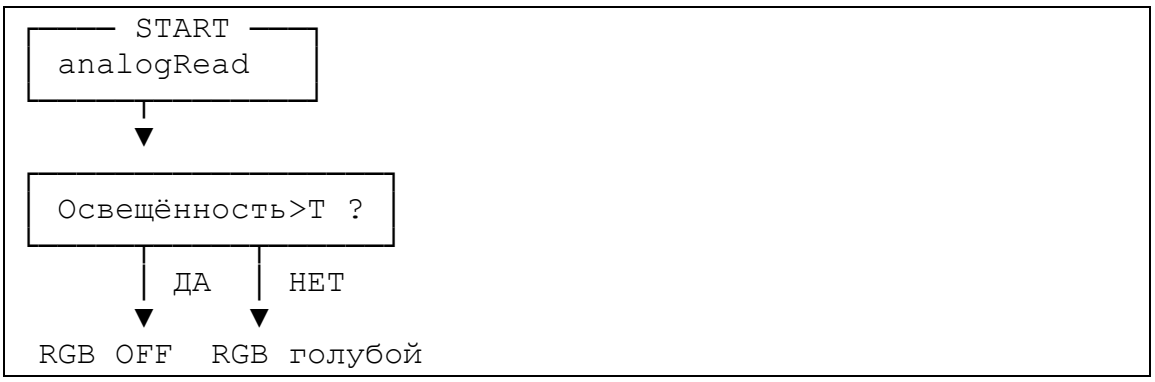
Задание № 2.1. Сборка электрической схемы



Задание № 2.2. Проектирование электрической схемы



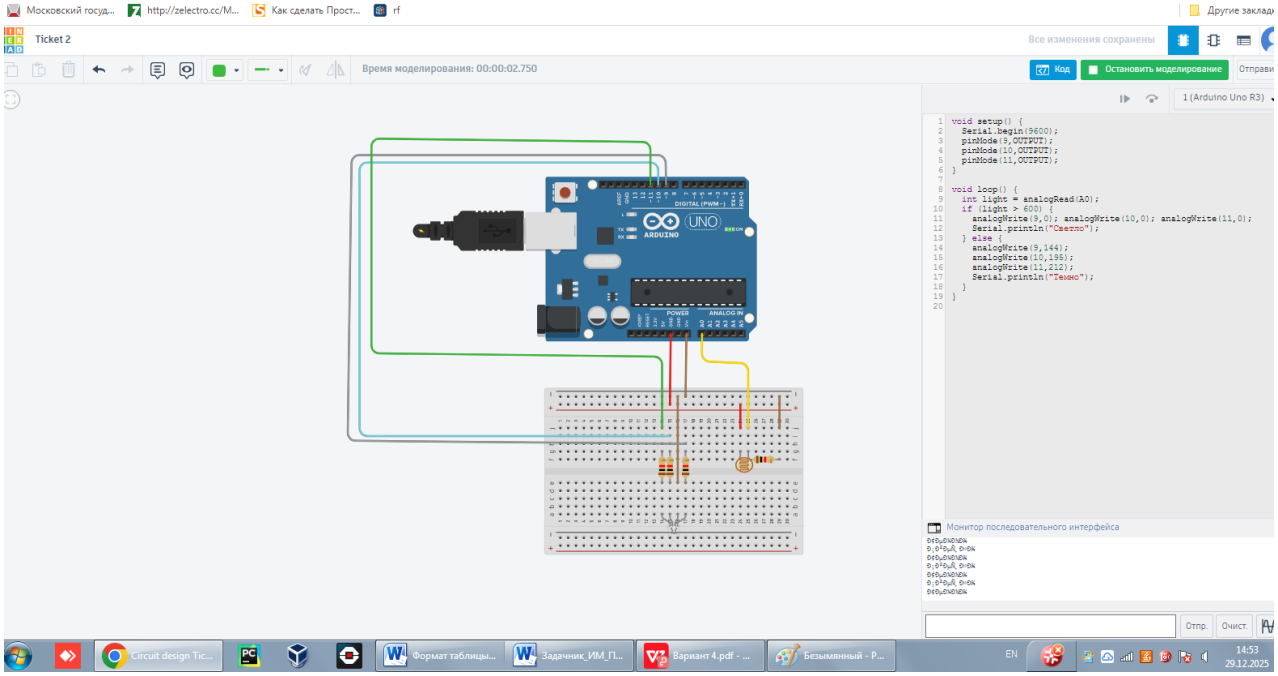
Задание № 3. Блок-схема алгоритма



Задание № 4.1. Управляющая программа и 4.2. Монитор последовательного порта

```
void setup() {
  Serial.begin(9600);
  pinMode(9,OUTPUT);
  pinMode(10,OUTPUT);
  pinMode(11,OUTPUT);
}

void loop() {
  int light = analogRead(A0);
  if (light > 600) {
    analogWrite(9,0); analogWrite(10,0);
analogWrite(11,0);
    Serial.println("Светло");
  } else {
    analogWrite(9,144);
    analogWrite(10,195);
    analogWrite(11,212);
```

```
    Serial.println("Темно");
  }
}
```



Кейс 1. Вариант 5.

Вентилятор - элемент настольной теплицы.

Написать программу для включения электродвигателя вентилятора теплицы, в зависимости от превышения заданной температуры (например, 22 градуса, или на полградуса выше температуры в помещении, где проводится измерение). Подключение электродвигателя к Arduino через транзистор NPN типа с использованием диода выполняется участниками по выбору для получения повышенных баллов. Датчик температуры - TMP36. В программе значения полученные через analogRead(A0) должны пересчитываться в градусы Цельсия. *Дополнительно* реализуйте вывод пересчитанной температуры в градусы Цельсия один раз в 2 секунды через окно монитора порта.

Оборудование:

- Резистор 200-330 Ом 1 шт.
- Arduino UNO и кабель питания USB
- Макетная плата
- Диод 1 шт.
- Соединительные провода и провода с зажимами «крокодил»
- Транзистор NPN рассчитанный на 5 вольт
- Датчик температуры TMP36
- Электродвигатель (двигатель постоянного тока)

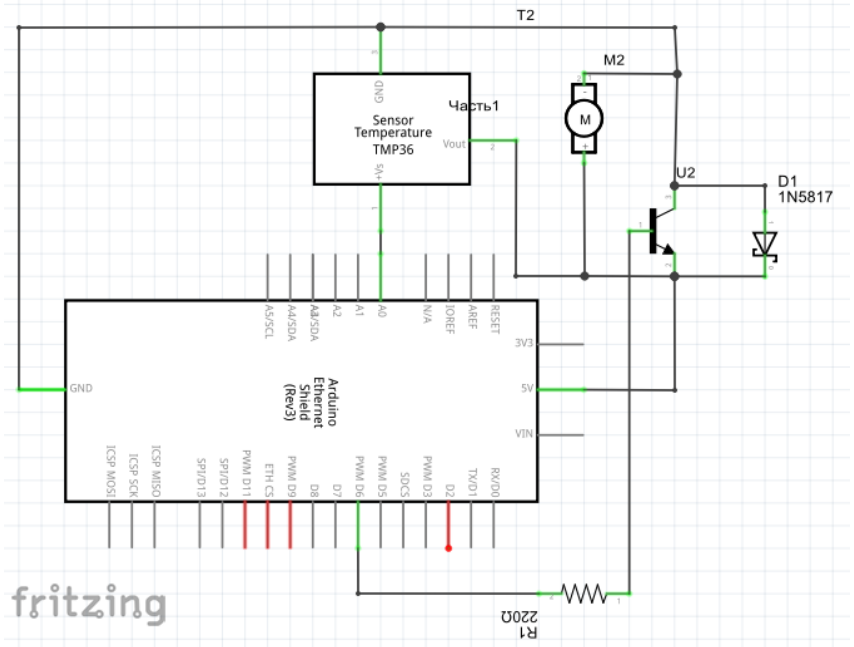
Решение.

Методические рекомендации к задаче и ответу.

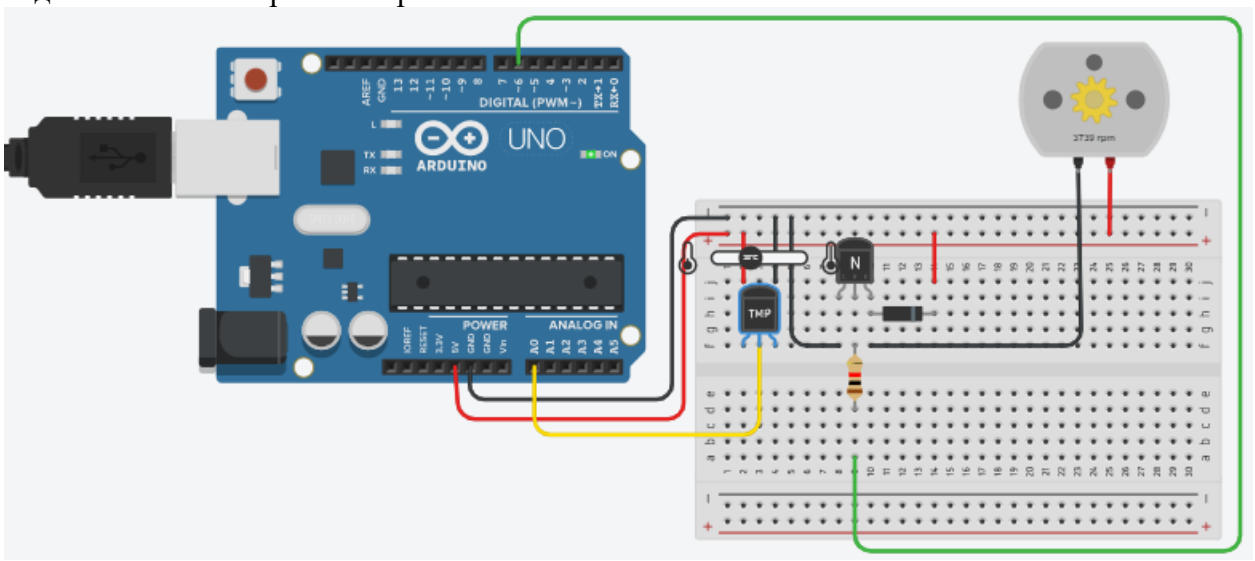
Условие. При проверке решения задачи проверяются знания метода управления электродвигателем в режиме проверки превышения порогового значения. Оценивается умение запрограммировать включение двигателя постоянного тока в режиме проверки порогового значения контролируемого сигнала.

Ответ. Ответ должен содержать программный код, в котором фигурирует наличие алгоритма включения двигателя постоянного тока при преодолении порогового значения контролируемого сигнала.

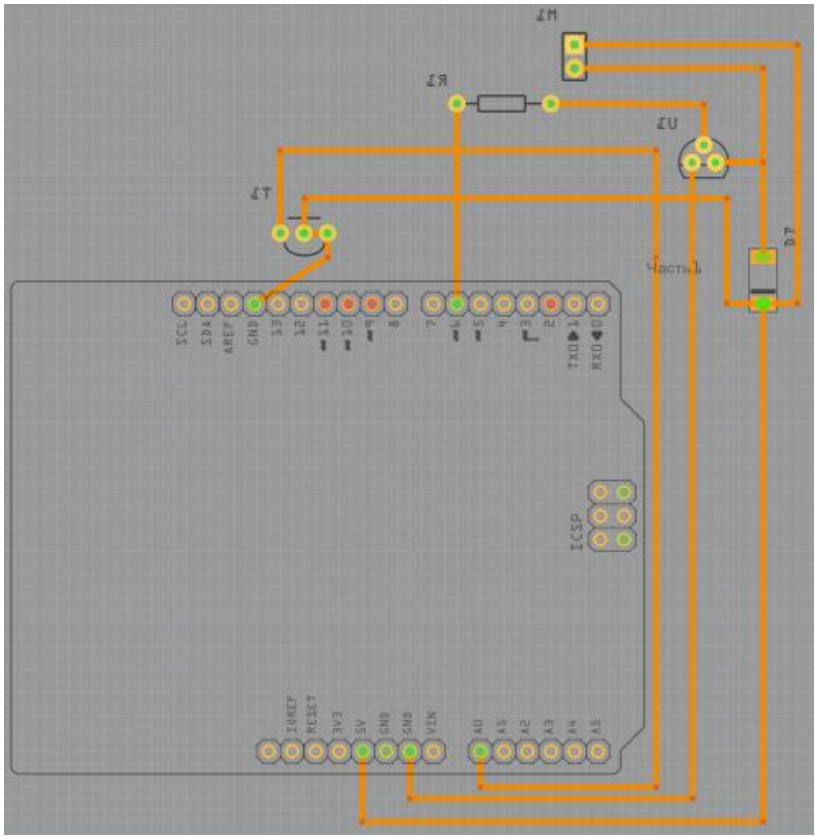
Задание № 1. Электрическая схема



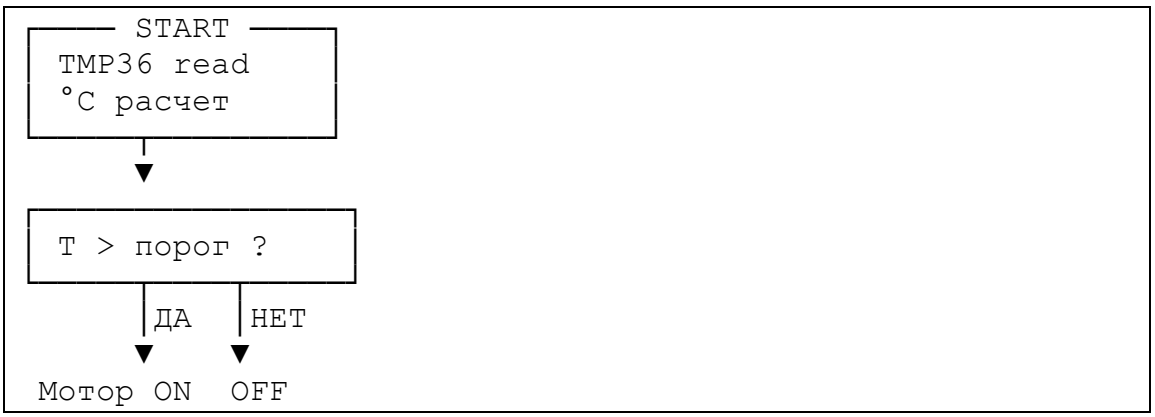
Задание № 2.1. Сборка электрической схемы



Задание № 2.2. Проектирование электрической схемы

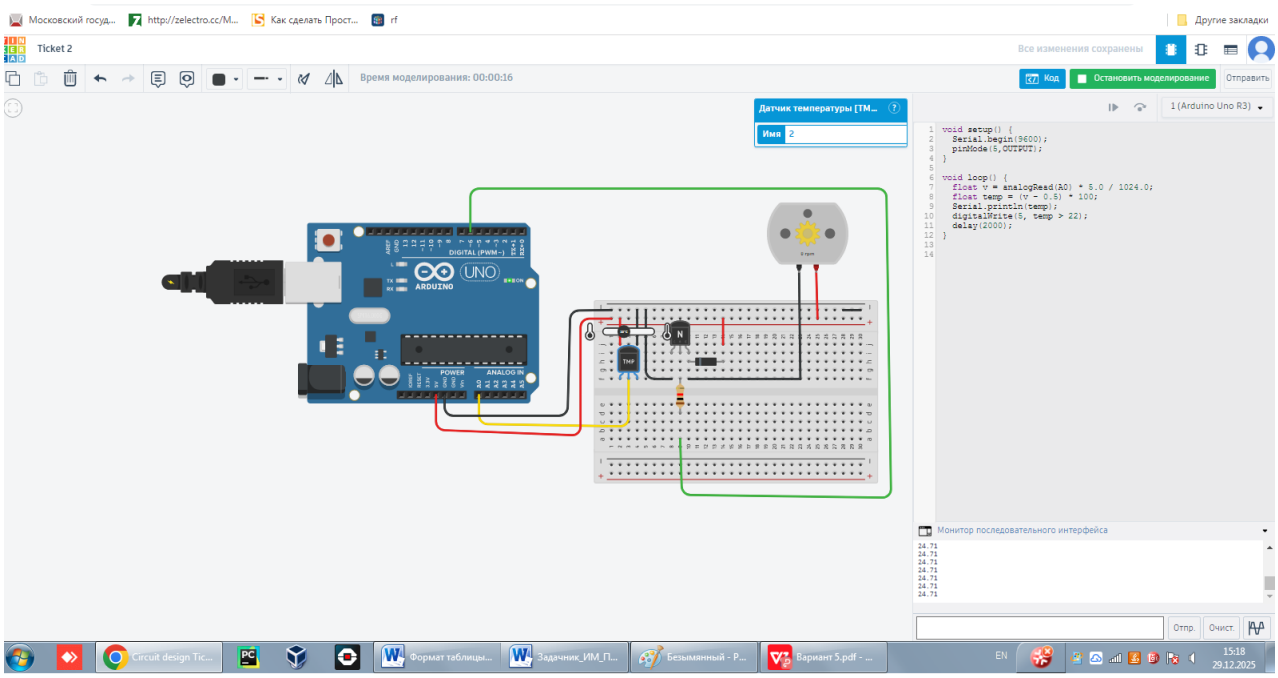


Задание № 3. Блок-схема алгоритма



Задание № 4.1. Управляющая программа и 4.2. Монитор последовательного порта

```
void setup() {
  Serial.begin(9600);
  pinMode(5,OUTPUT);
}

void loop() {
  float v = analogRead(A0) * 5.0 / 1024.0;
  float temp = (v - 0.5) * 100;
  Serial.println(temp);
  digitalWrite(5, temp > 22);
  delay(2000);
}
```



Кейс 1. Вариант 6.

Форточка - элемент настольной теплицы.

Написать программу для включения сервомотора форточки, в зависимости от превышения заданной температуры (например, 22 градуса, или на полградуса выше температуры в помещении, где проводится измерение). Серомотор подключен к Arduino используя один из выходов ШИМ. Применение в схеме мультиметра и считывание с него показаний выполняется участниками по выбору для получения повышенных баллов. Датчик температуры - TMP36. В программе значения полученные через analogRead(A0) должны пересчитываться в градусы Цельсия. *Дополнительно* реализуйте вывод пересчитанной температуры в градусы Цельсия один раз в 5 секунды через окно монитора порта.

Оборудование:

- Arduino UNO и кабель питания USB
- Макетная плата
- Соединительные провода и провода с зажимами «крокодил»
- Сервомотор
- Датчик температуры TMP36
- Мультиметр (вольтметр)

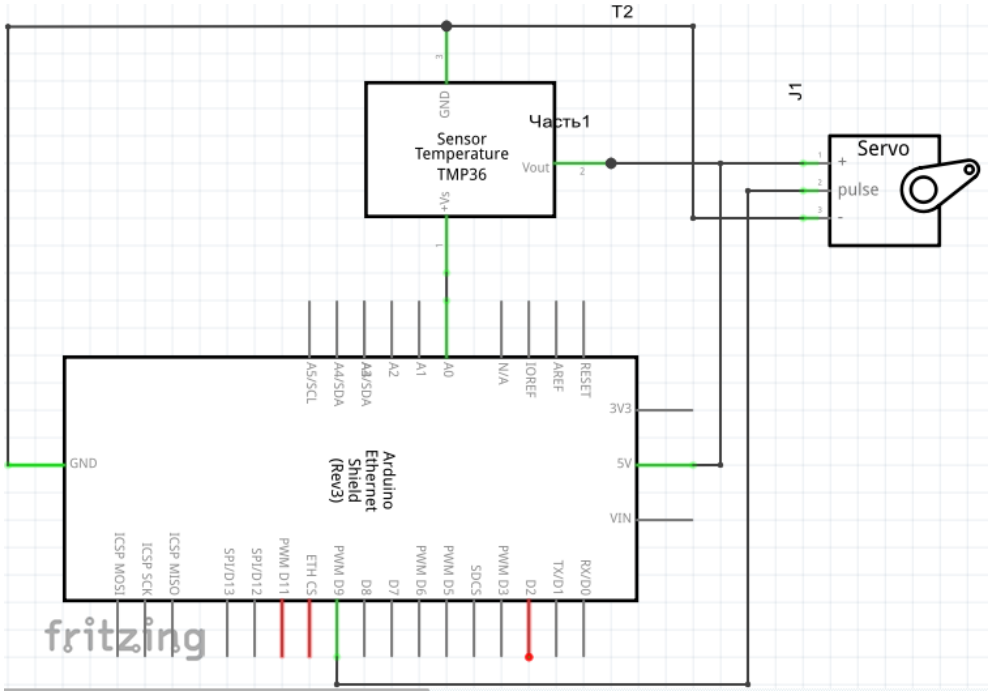
Решение.

Методические рекомендации к задаче и ответу.

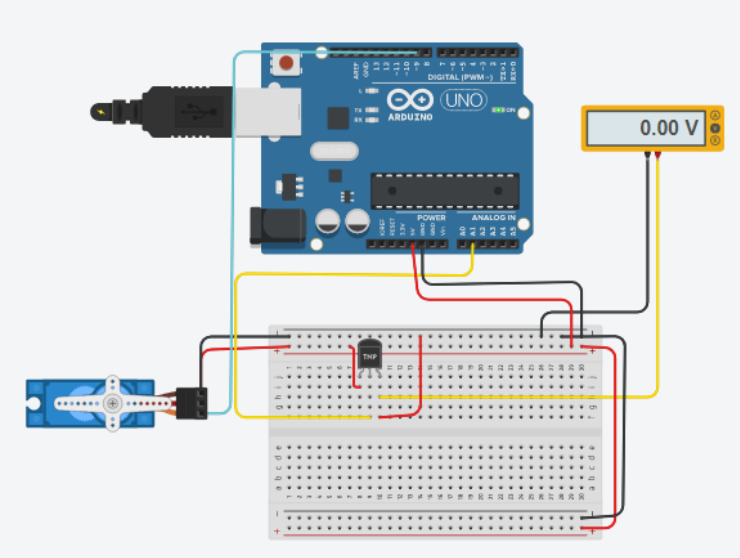
Условие. При проверке решения задачи проверяются знания метода управления сервомотором в режиме проверки превышения порогового значения. Оценивается умение запрограммировать включение сервомотора в режиме проверки порогового значения контролируемого сигнала.

Ответ. Ответ должен содержать программный код, в котором фигурирует наличие алгоритма включения сервомотора при преодолении порогового значения контролируемого сигнала.

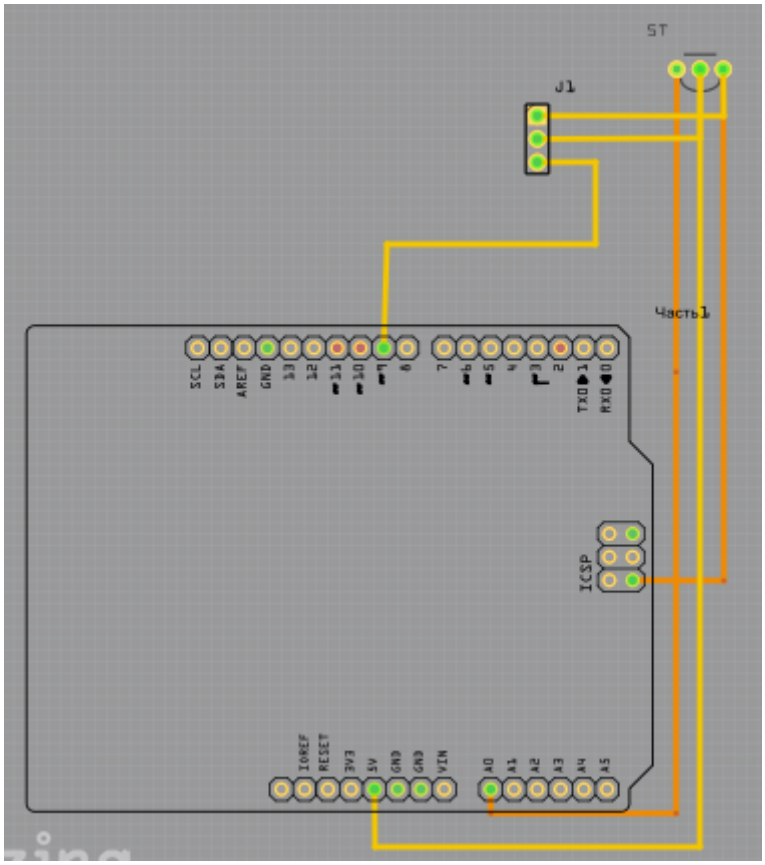
Задание № 1. Электрическая схема



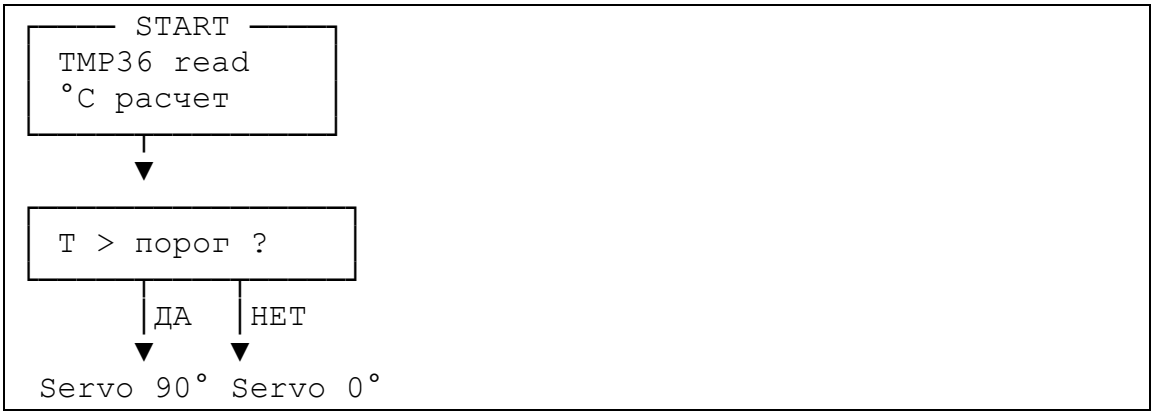
Задание № 2.1. Сборка электрической схемы



Задание № 2.2. Проектирование электрической схемы

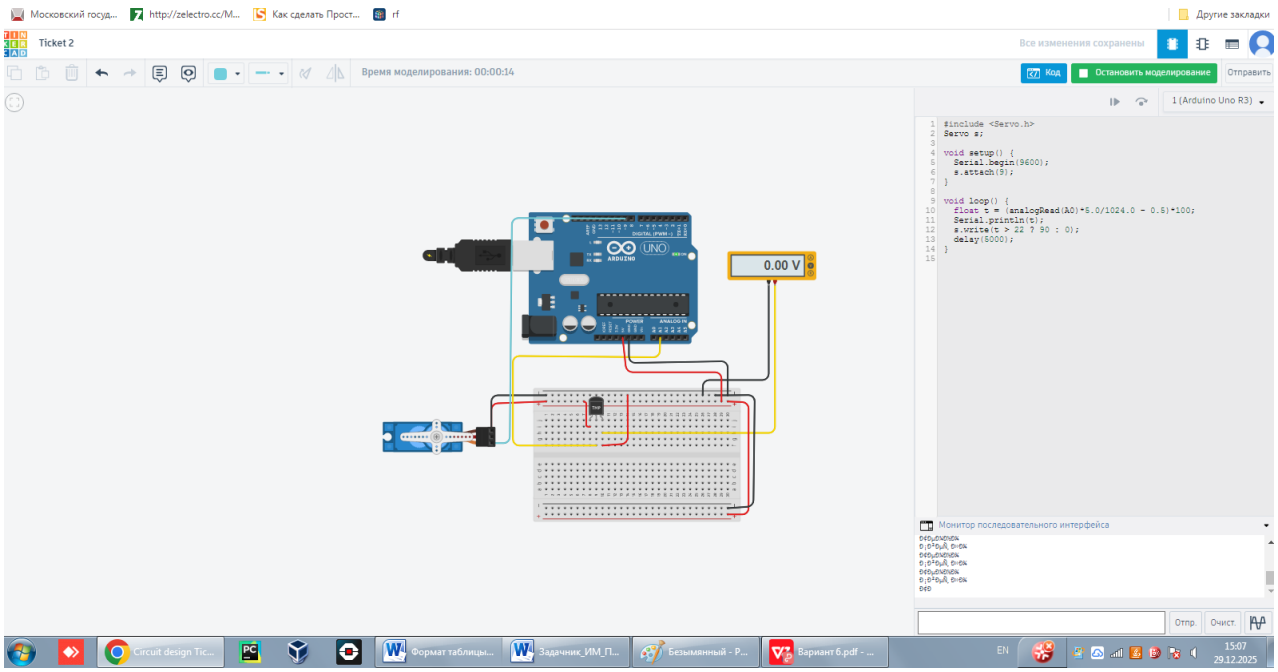


Задание № 3. Блок-схема алгоритма



Задание № 4.1. Управляющая программа и 4.2. Монитор последовательного порта

```
#include <Servo.h>
Servo s;

void setup() {
  Serial.begin(9600);
  s.attach(9);
}

void loop() {
  float t = (analogRead(A0)*5.0/1024.0 - 0.5)*100;
  Serial.println(t);
  s.write(t > 22 ? 90 : 0);
  delay(5000);
}
```



Кейс 1. Вариант 7.

Мотор-колесо мобильного робота.

Написать программу для управления вращением мотор-колеса мобильного робота. Включение мотор-колеса производить на заданное время 5 сек. По завершению отсчета ~~оба~~ мотор-колесо останавливается. Необходимо написать программу управляющую включением мотор-колеса в одну и другую стороны (по и против часовой стрелке), а так же полной остановкой. Для управления вращением двигателя в разные стороны используется схема Н-моста - L293B. *Дополнительно* реализуйте вывод информации о режиме вращения через окно монитора порта. В окно монитора порта 1 раз в 1 секунду-должна выводиться информация о режиме работы двигателя.

Оборудование:

- Драйвер двигателя L293B
- Arduino UNO и кабель питания USB
- Макетная плата
- Соединительные провода и провода с зажимами «крокодил»
- Мотор-редуктор

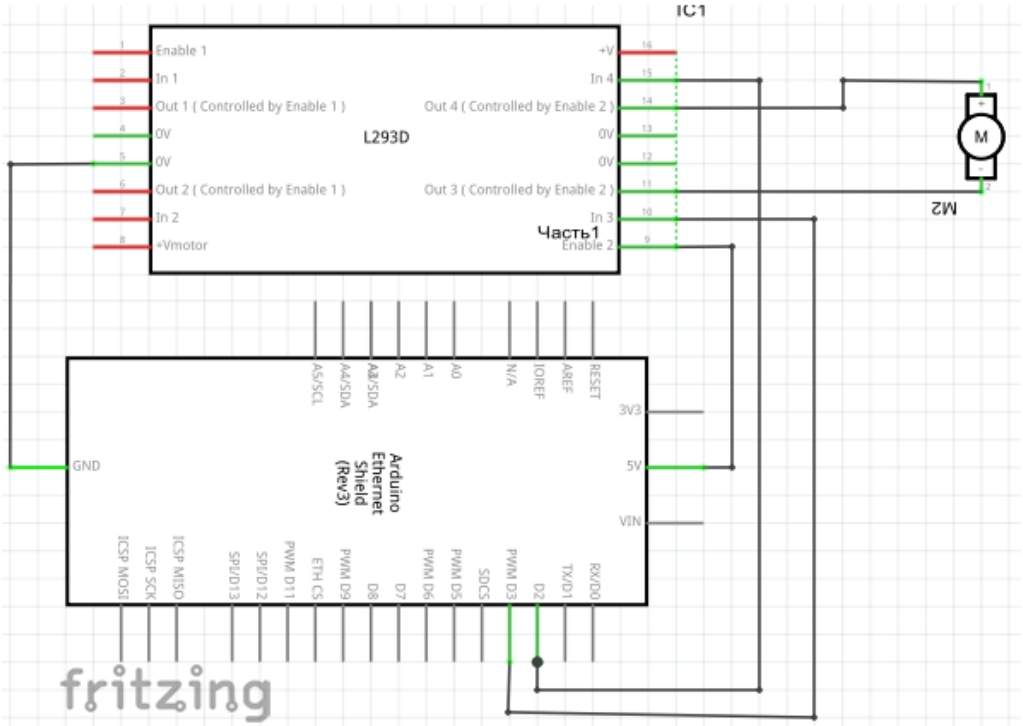
Решение.

Методические рекомендации к задаче и ответу.

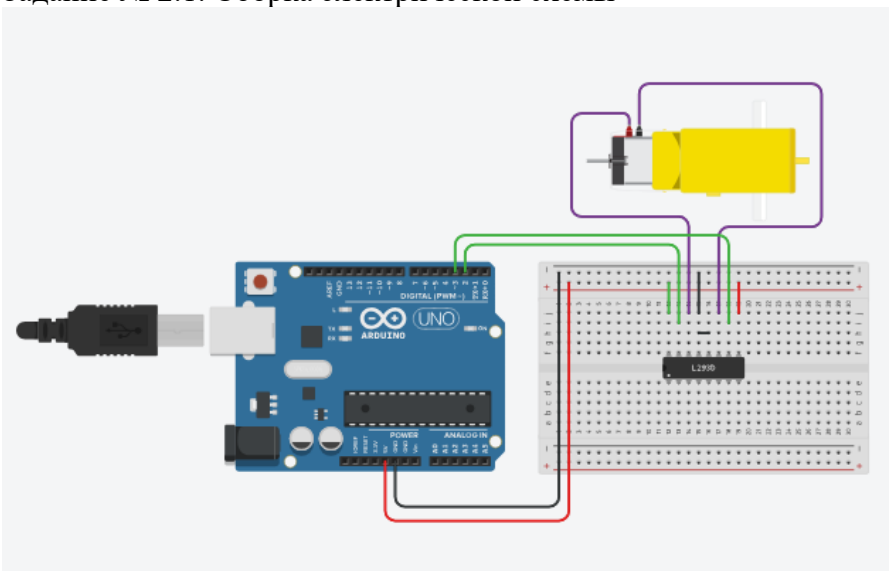
Условие. При проверке решения задачи проверяются знания метода запуска мотор-колеса в составе схема Н-моста. Оценивается умение программировать режимы работы мотор-колеса.

Ответ. Ответ должен содержать программный код, в котором фигурирует элементы программного управления мотор-колесом.

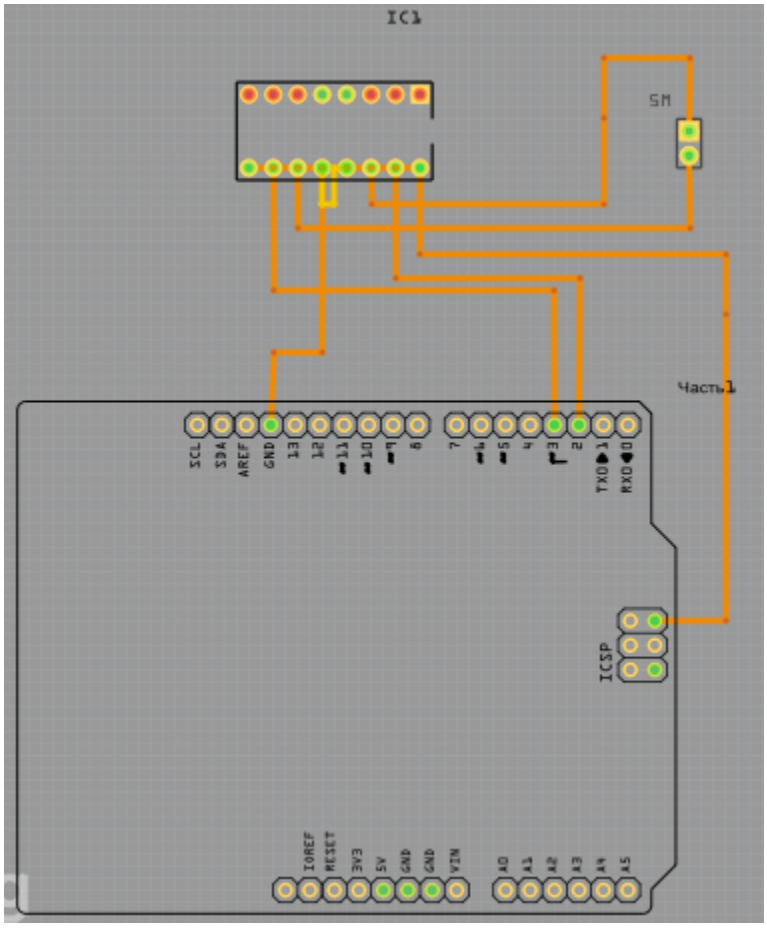
Задание № 1. Электрическая схема



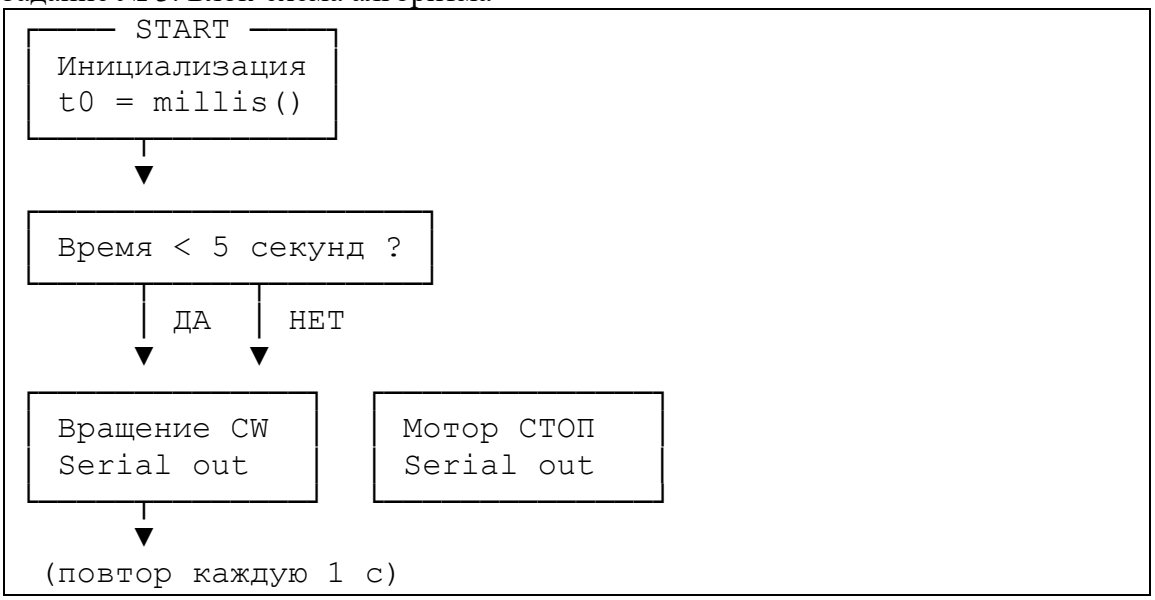
Задание № 2.1. Сборка электрической схемы



Задание № 2.2. Проектирование электрической схемы



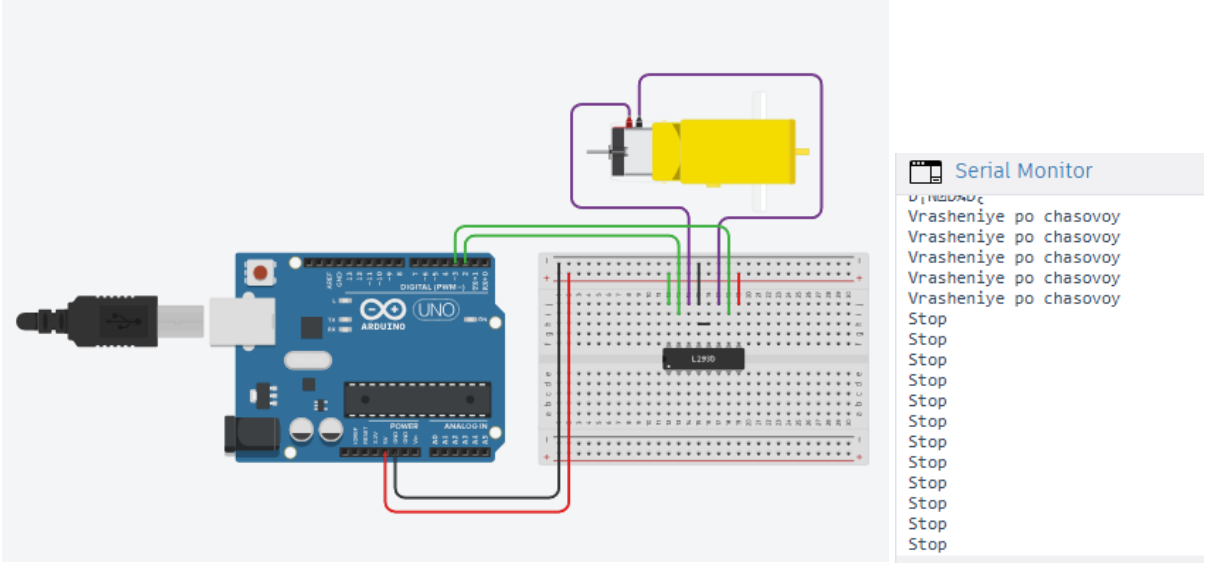
Задание № 3. Блок-схема алгоритма



Задание № 4.1. Управляющая программа и 4.2. Монитор последовательного порта

```
const int IN1 = 3;
const int IN2 = 4;
unsigned long startTime;

void setup() {
  Serial.begin(9600);
  pinMode(IN1, OUTPUT);
  pinMode(IN2, OUTPUT);
  startTime = millis();
}
```

```
void loop() {
  if (millis() - startTime < 5000) {
    digitalWrite(IN1, HIGH);
    digitalWrite(IN2, LOW);
    Serial.println("Вращение по часовой");
  } else {
    digitalWrite(IN1, LOW);
    digitalWrite(IN2, LOW);
    Serial.println("Стоп");
  }
  delay(1000);
}
```



Кейс 1. Вариант 8.

Электронное пианино.

Разработать программу преобразования нажатия шести кнопок в частоту звучания пьезодинамика. Каждая нажатая клавиша должна сопоставляться с определенной частотой. *Дополнительно* реализуйте вывод нажатых кнопок в окно монитора порта. *Справочно*. Используйте ноты первой октавы гаммы До Мажор и соответствующие им приблизительные частоты: C(До) 264; D(Ре) 294; E(Ми) 330; F(Фа) 394; G(Соль) 392; A(Ля) 440. Числа — это значения, которые программно передаются на динамик и приблизительно соответствуют тональности нот.

Оборудование:

- Пьезодинамик
- Резистор 200-330 Ом 1 шт.
- Arduino UNO и кабель питания USB
- Макетная плата
- Соединительные провода
- Кнопки 6 шт.

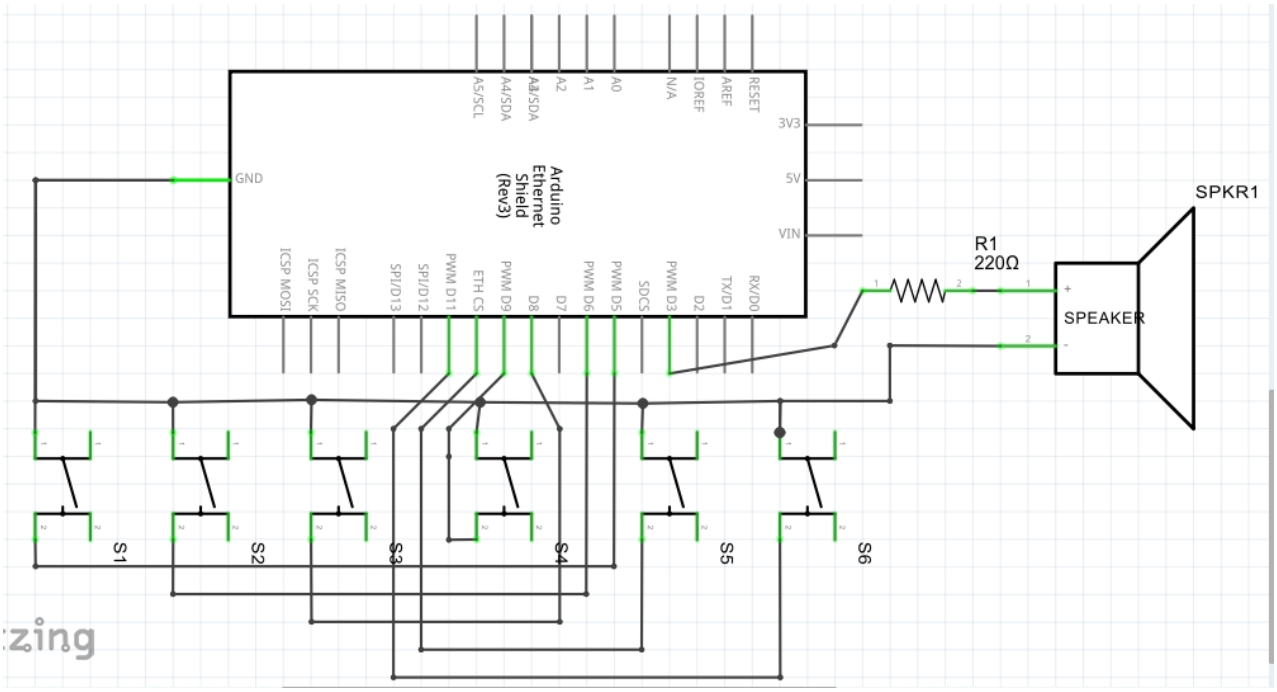
Решение.

Методические рекомендации к задаче и ответу.

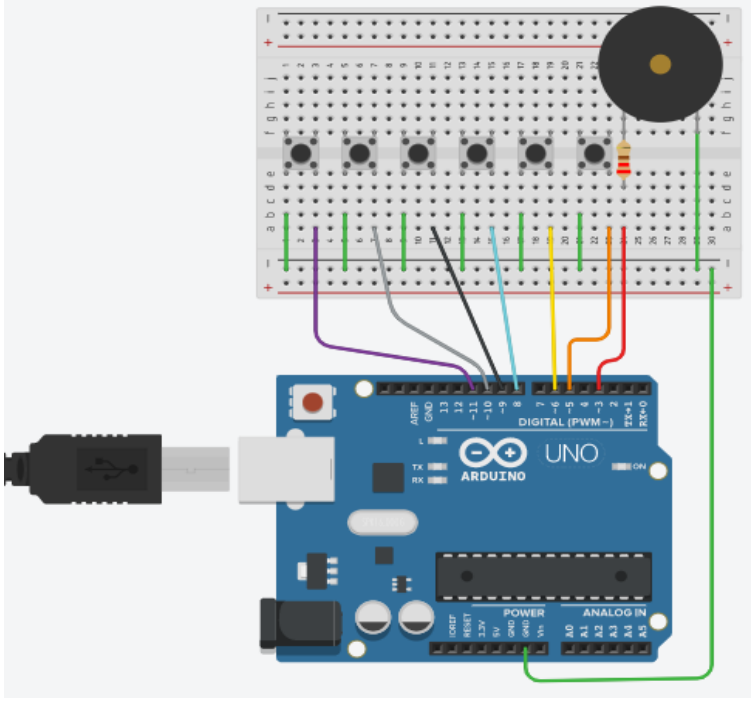
Условие. При проверке решения задачи проверяются знания метода выбора из множества вариантов. Оценивается умение программно реализовывать выбор из множества вариантов.

Ответ. Ответ должен содержать программный код в котором фигурирует наличие применения оператора выбора из вариантов.

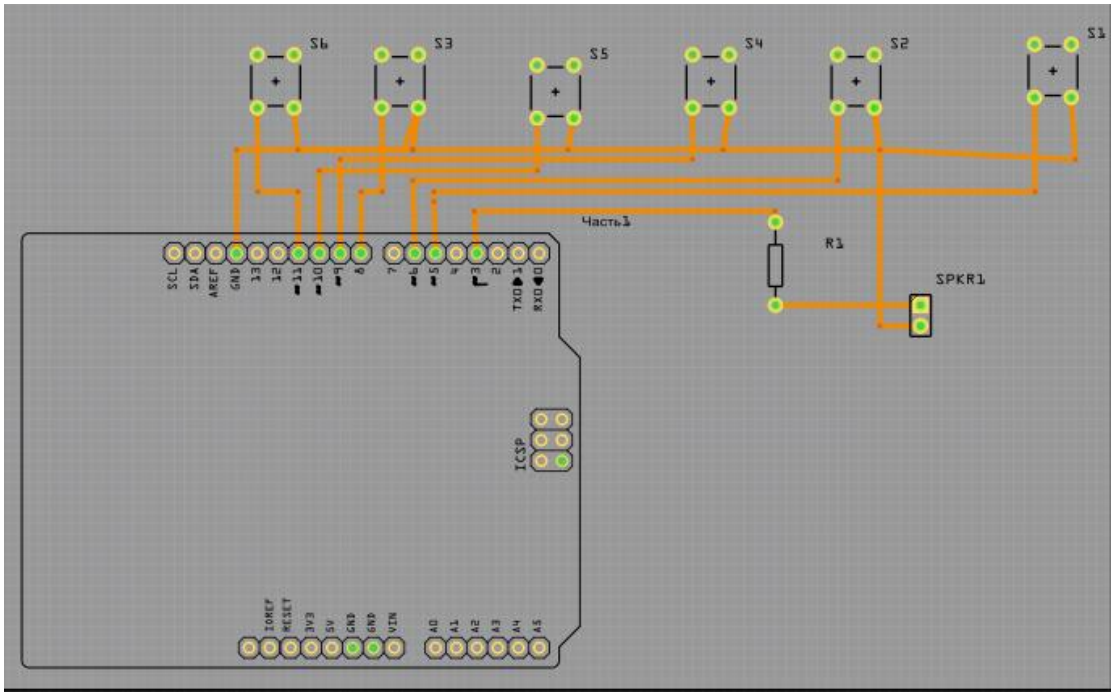
Задание № 1. Электрическая схема



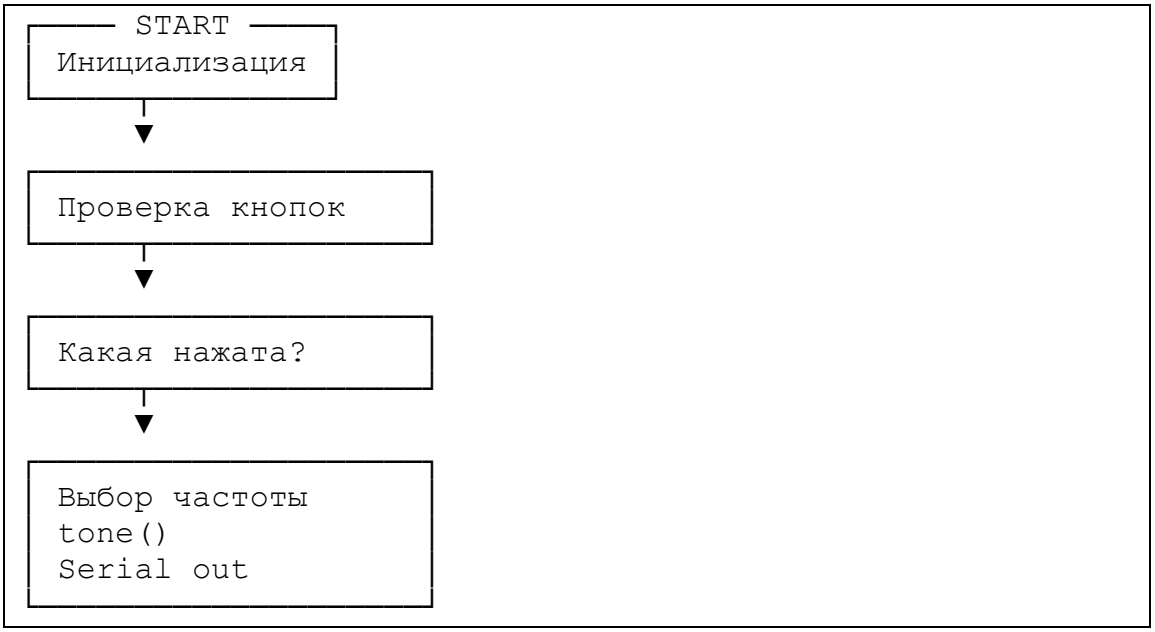
Задание № 2.1. Сборка электрической схемы



Задание № 2.2. Проектирование электрической схемы



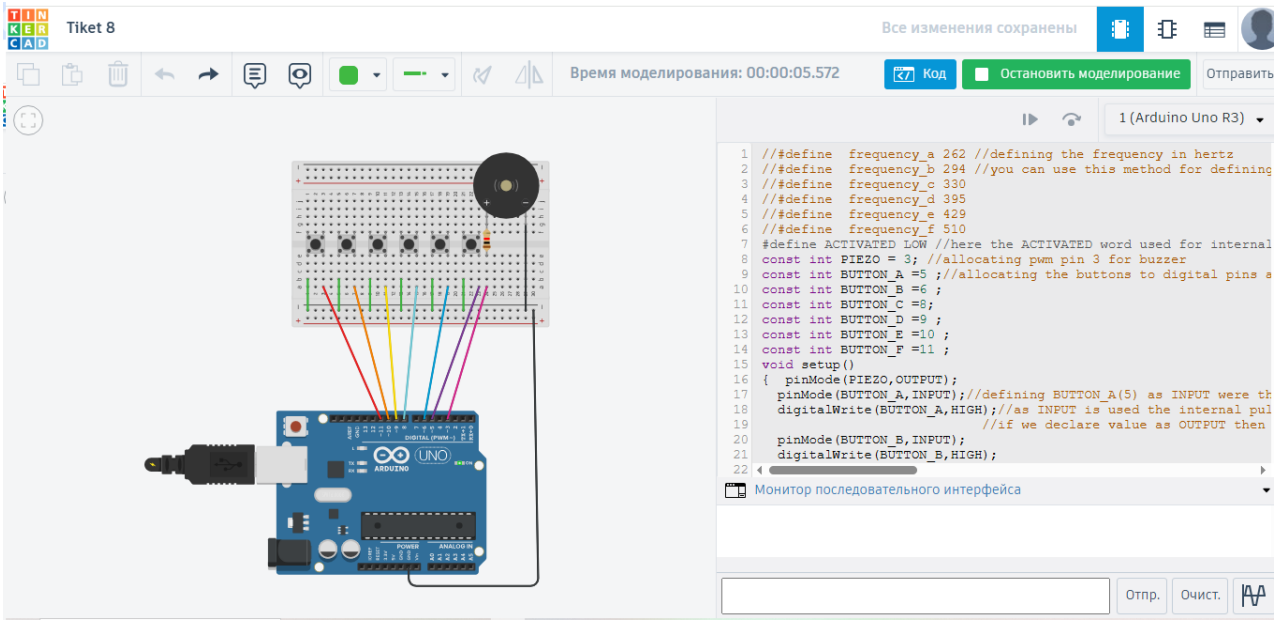
Задание № 3. Блок-схема алгоритма



Задание № 4.1. Управляющая программа и 4.2. Монитор последовательного порта

```
//#define  frequency_a 262 //defining the frequency in
hertz
//#define  frequency_b 294 //you can use this method for
defining frequency (or) the method i used below directly
giving frequency in tone function
//#define  frequency_c 330
//#define  frequency_d 395
//#define  frequency_e 429
//#define  frequency_f 510
#define ACTIVATED LOW //here the ACTIVATED word used for
internal pull_up resistor when the button pressed it
compare with this and if it is LOW BUZZER BUZZERS
const int PIEZO = 3; //allocating pwm pin 3 for buzzer
const int BUTTON_A =5 ;//allocating the buttons to digital
pins accordingly
```

```
const int BUTTON_B =6 ;
const int BUTTON_C =8;
const int BUTTON_D =9 ;
const int BUTTON_E =10 ;
const int BUTTON_F =11 ;
void setup()
{  pinMode(PIEZO,OUTPUT);
  pinMode(BUTTON_A,INPUT);//defining BUTTON_A(5) as INPUT
were the value is read by ardunio
  digitalWrite(BUTTON_A,HIGH);//as INPUT is used the
internal pull-up resistor acts on.so When we use INPUT
then if we declare value as HIGH then pull-up resistor
fully acts and for LOW the resistance will decrease
                                    //if we declare value as
OUTPUT then it acts reversly HIGH means resistance
decreases
  pinMode(BUTTON_B,INPUT);
  digitalWrite(BUTTON_B,HIGH);
  pinMode(BUTTON_C,INPUT);
  digitalWrite(BUTTON_C,HIGH);
  pinMode(BUTTON_D,INPUT);
  digitalWrite(BUTTON_D,HIGH);
   pinMode(BUTTON_E,INPUT);
  digitalWrite(BUTTON_E,HIGH);
   pinMode(BUTTON_F,INPUT);
  digitalWrite(BUTTON_F,HIGH);
}
void loop()
{  while(digitalRead(BUTTON_A) == ACTIVATED)//so when the
button pressed the resistance decreases so if button value
is equal to ACTIVATED(LOW)then tone() will start
  {    tone(PIEZO,380,500);//here PIEZO MEANS PIN 3,380 is
frequency,500 is time to play sound in milliseconds
                              //if you use above method as
definig freq at starting then you should give the name at
place of 380 like frequency_a which above given
  }
  while(digitalRead(BUTTON_B) == ACTIVATED)
  {    tone(PIEZO,420,500);   }
  while(digitalRead(BUTTON_C) == ACTIVATED)
  {    tone(PIEZO,490,500);  }
  while(digitalRead(BUTTON_D) == ACTIVATED)
  {     tone(PIEZO,530,500);  }
  while(digitalRead(BUTTON_E) == ACTIVATED)
  {    tone(PIEZO,590,500);   }
  while(digitalRead(BUTTON_F) == ACTIVATED)
  {     tone(PIEZO,620,500);  }
  noTone(PIEZO);//it stops the tone() and loop continues
}
```



Кейс 1. Вариант 9.

Управляемая светодиодная лампа.

Необходимо запрограммировать включение светодиодов на схеме в следующих режимах: каждый светодиод включается по очереди по часовой стрелке, против часовой стрелке, все светодиоды выключены. В режиме работы по часовой стрелке, включение очередного светодиода происходит 1 раз в секунду. При включении очередного светодиода предыдущий выключается. Включение каждого из трех режимов задается потенциометром. При выкручивании рукоятки потенциометра в начало - светодиоды не горят. При выкручивании ручки потенциометра на половину от полного оборота запускается режим включения по часовой стрелке. При выкручивании ручки потенциометра на полный оборот запускается режим включения против часовой стрелке. *Дополнительно*. Реализуйте вывод в окно монитора порта режим работы лампы.

Оборудование:

- Четыре красных светодиода
- Два токоограничивающих резистора 150 Ом Arduino Uno
- Arduino UNO и кабель питания USB
- Макетная плата
- Потенциометр
- Соединительные провода

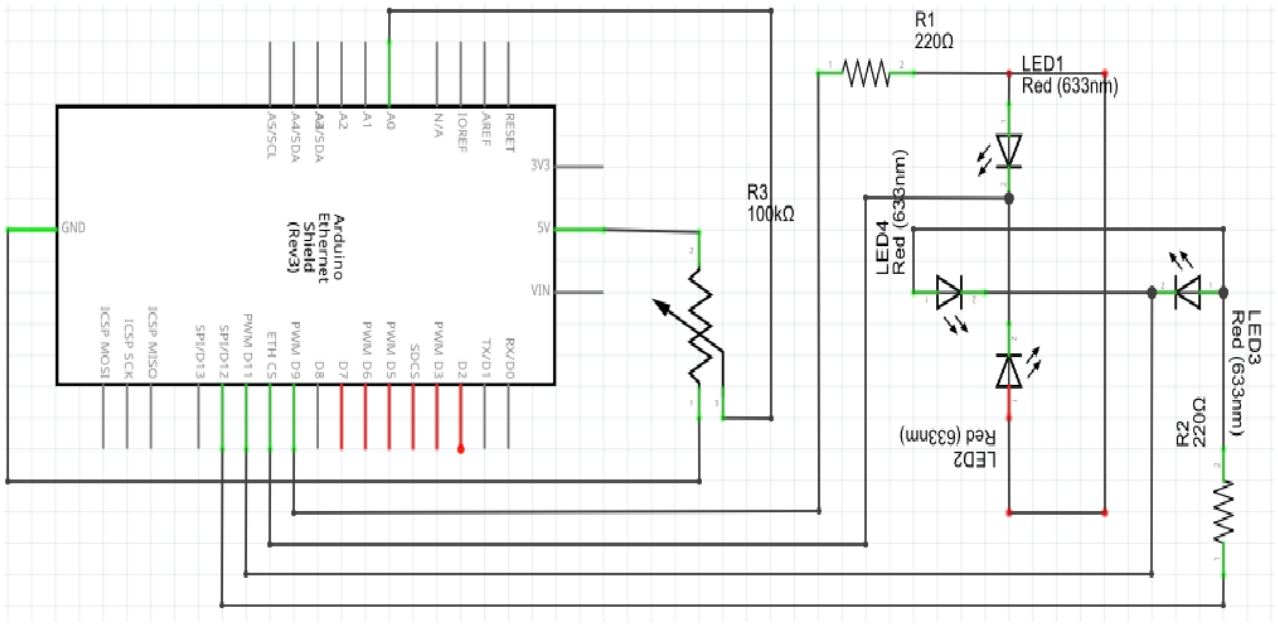
Решение.

Методические рекомендации к задаче и ответу.

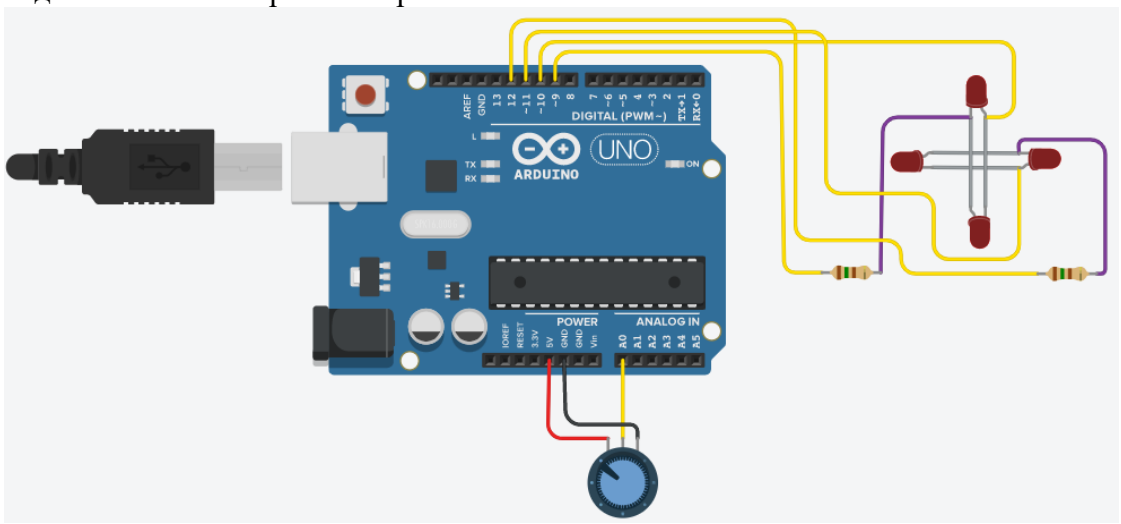
Условие. При проверке решения задачи проверяются знания принципов поочередной подачи сигналов. Оценивается умение алгоритмически представить работу светодиодной лампы.

Ответ. Ответ должен содержать программный код, в котором выполняется управление светодиодной лампой.

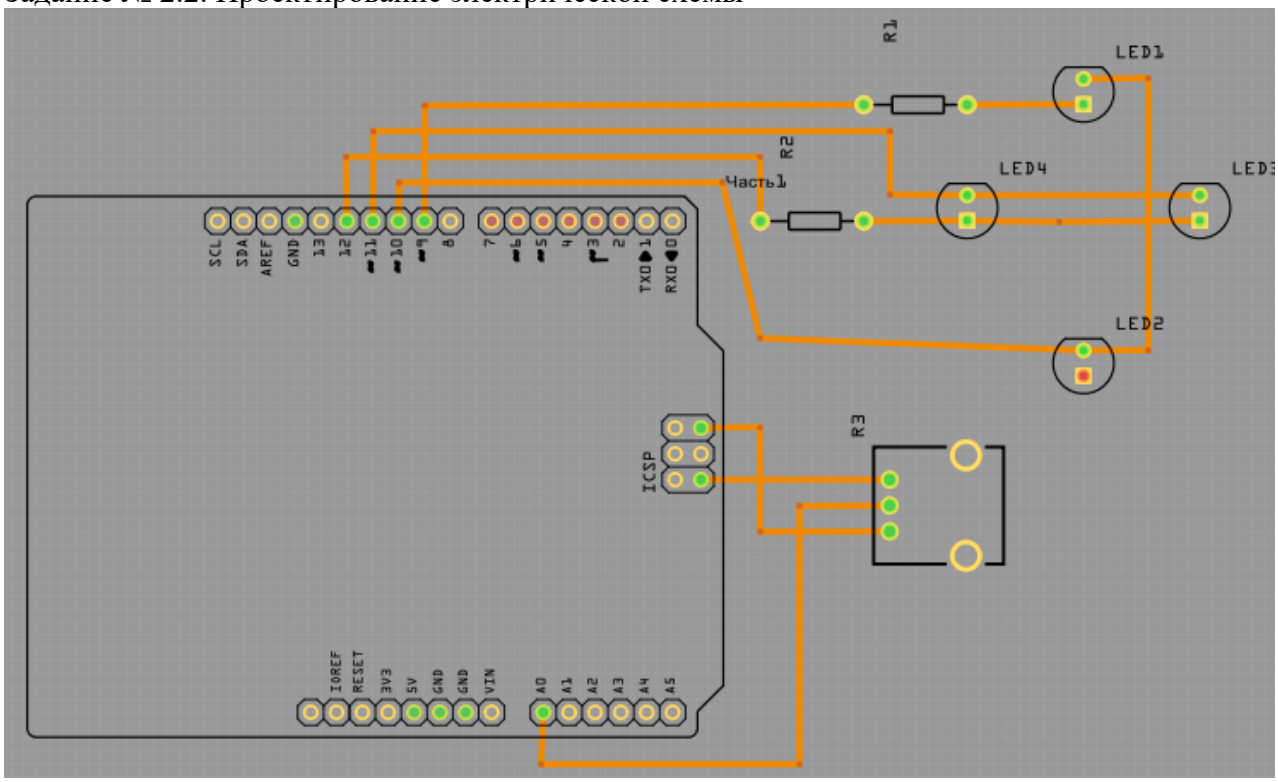
Задание № 1. Электрическая схема



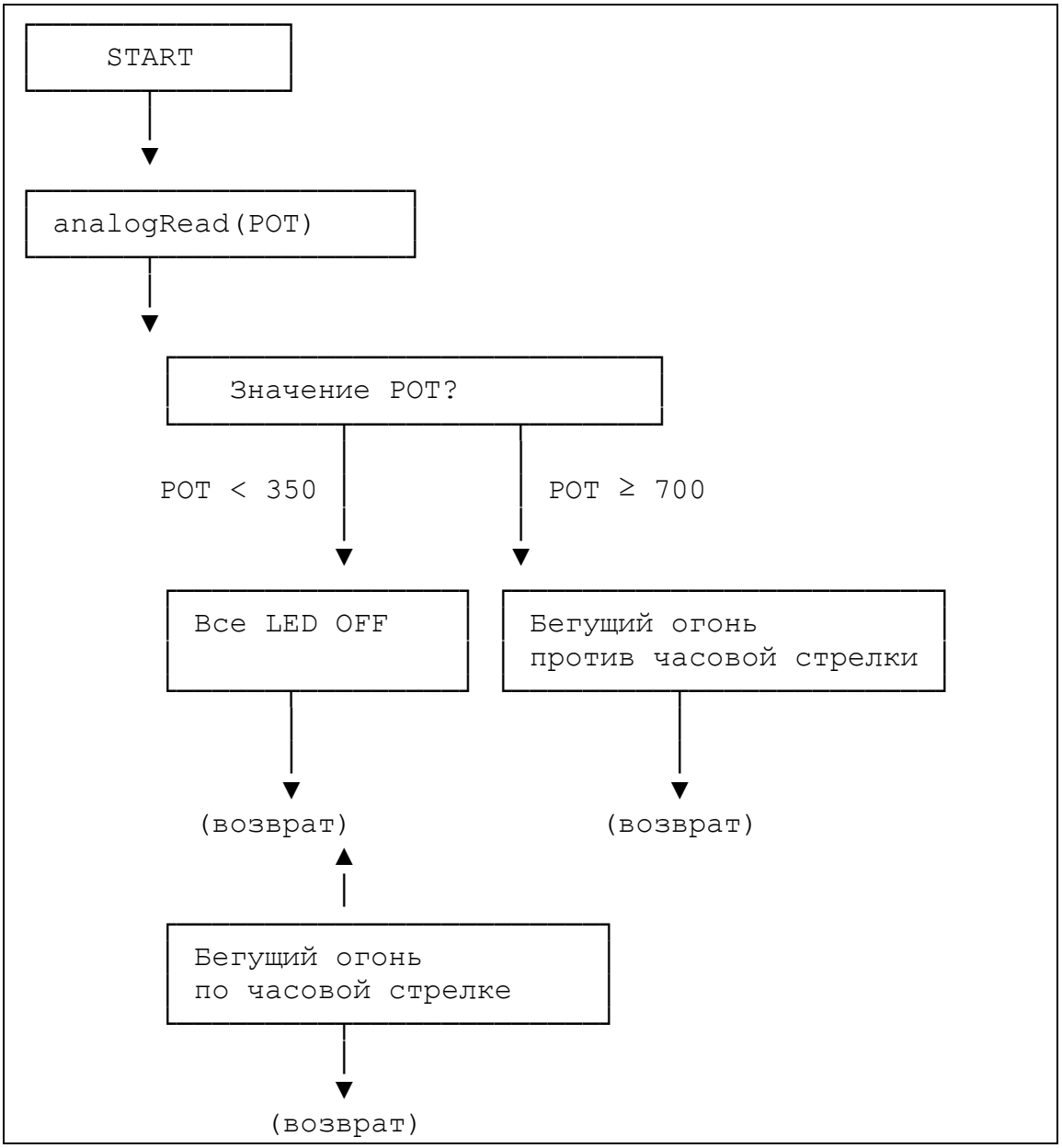
Задание № 2.1. Сборка электрической схемы



Задание № 2.2. Проектирование электрической схемы



Задание № 3. Блок-схема алгоритма

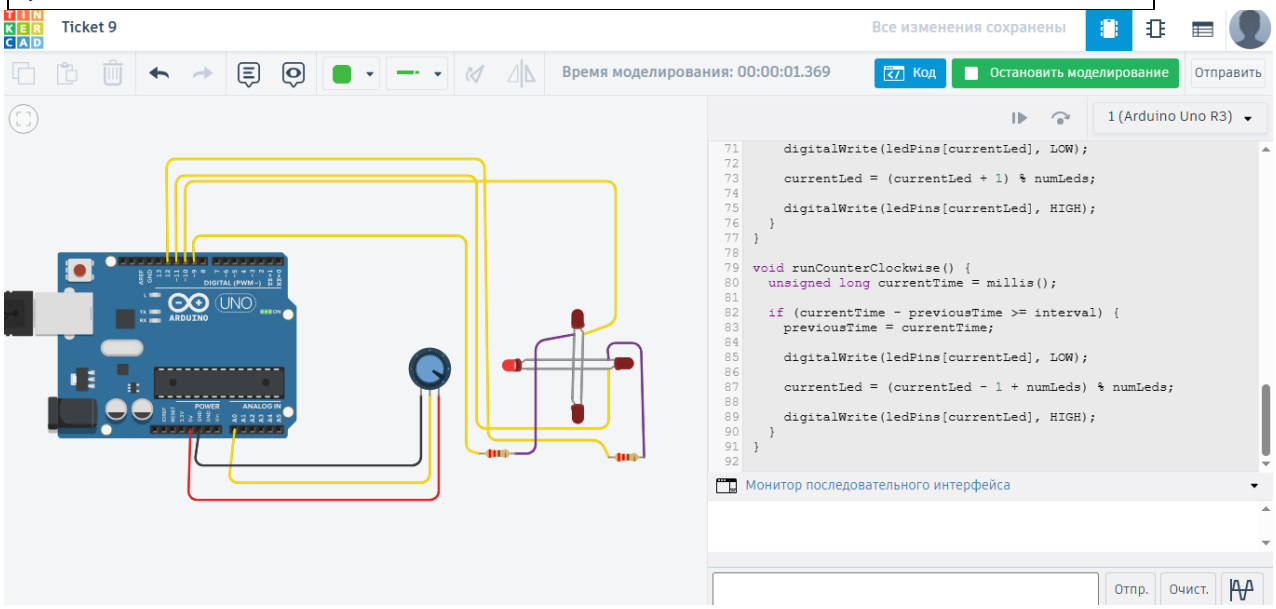


Задание № 4.1. Управляющая программа и 4.2. Монитор последовательного порта

```
// C++ code
//
const int potPin = A0;

const int ledPins[] = {9, 10, 11, 12};
const int numLeds = 4;

enum Mode {
  OFF,
  CLOCKWISE,
  COUNTER_CLOCKWISE
};

int currentLed = 0;
unsigned long previousTime = 0;
const long interval = 1000;
Mode currentMode = OFF;
```

```
void setup() {
  for (int i = 0; i < numLeds; i++) {
    pinMode(ledPins[i], OUTPUT);
    digitalWrite(ledPins[i], LOW);
  }

  pinMode(potPin, INPUT);


}

void loop() {
  int potValue = analogRead(potPin);

  if (potValue < 341) {
    currentMode = OFF;
  } else if (potValue < 682) {
    currentMode = CLOCKWISE;
  } else {
    currentMode = COUNTER_CLOCKWISE;
  }

  switch (currentMode) {
    case OFF:
      turnOffAllLeds();
      break;

    case CLOCKWISE:
      runClockwise();
      break;

    case COUNTER_CLOCKWISE:
      runCounterClockwise();
      break;
  }
}



void turnOffAllLeds() {
  for (int i = 0; i < numLeds; i++) {
    digitalWrite(ledPins[i], LOW);
  }
  currentLed = 0;
}

void runClockwise() {
  unsigned long currentTime = millis();

  if (currentTime - previousTime >= interval) {
    previousTime = currentTime;

    digitalWrite(ledPins[currentLed], LOW);

    currentLed = (currentLed + 1) % numLeds;
```

```
    digitalWrite(ledPins[currentLed], HIGH);
  }
}

void runCounterClockwise() {
  unsigned long currentTime = millis();

  if (currentTime - previousTime >= interval) {
    previousTime = currentTime;

    digitalWrite(ledPins[currentLed], LOW);

    currentLed = (currentLed - 1 + numLeds) % numLeds;

    digitalWrite(ledPins[currentLed], HIGH);
  }
}
```



Кейс 1. Вариант 10.

Кнопочное управление светодиодным светильником

Необходимо запрограммировать включение светодиодов на схеме в следующих режимах: каждый светодиод включается по очереди по часовой стрелке, против часовой стрелке, все светодиоды выключены. В режиме работы по часовой стрелке, включение очередного светодиода происходит 1 раз в секунду. При включении очередного светодиода предыдущий выключается. Включение каждого из трех режимов задается кнопочным выключателем. Изначально, после запуска программы, все светодиоды выключены. При однократном нажатии оператором, тестирующим программу, кнопочного выключателя, светодиоды должны начать включаться по часовой стрелке. При двукратном нажатии (два кратковременных нажатия в 1 сек с перерывом в 1 сек), светодиоды поочередно включаются против часовой стрелке. *Дополнительно*. Реализуйте вывод в окно монитора порта режим работы светильника.

Оборудование:

- Четыре красных светодиода
- Два токоограничивающих резистора 150 Ом Arduino Uno
- Резистор 10 кОм
- Arduino UNO и кабель питания USB
- Макетная плата
- Соединительные провода

- кнопочный выключатель

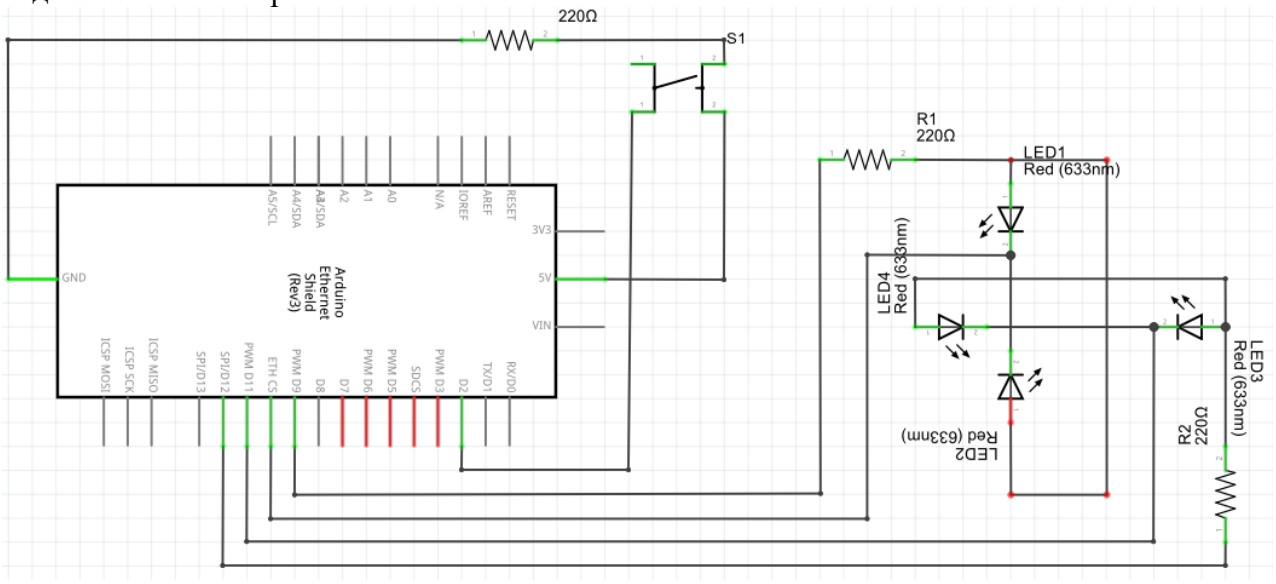
Решение.

Методические рекомендации к задаче и ответу.

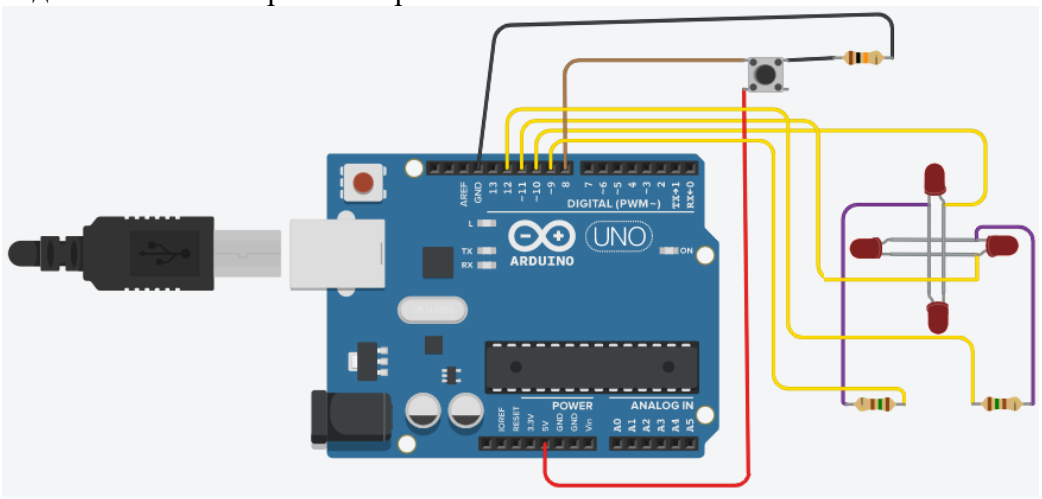
Условие. При проверке решения задачи проверяются знания режимов управления с помощью кнопочного выключателя. Оценивается умение алгоритмически представить работу светильника.

Ответ. Ответ должен содержать программный код, в котором выполняется управление светильником.

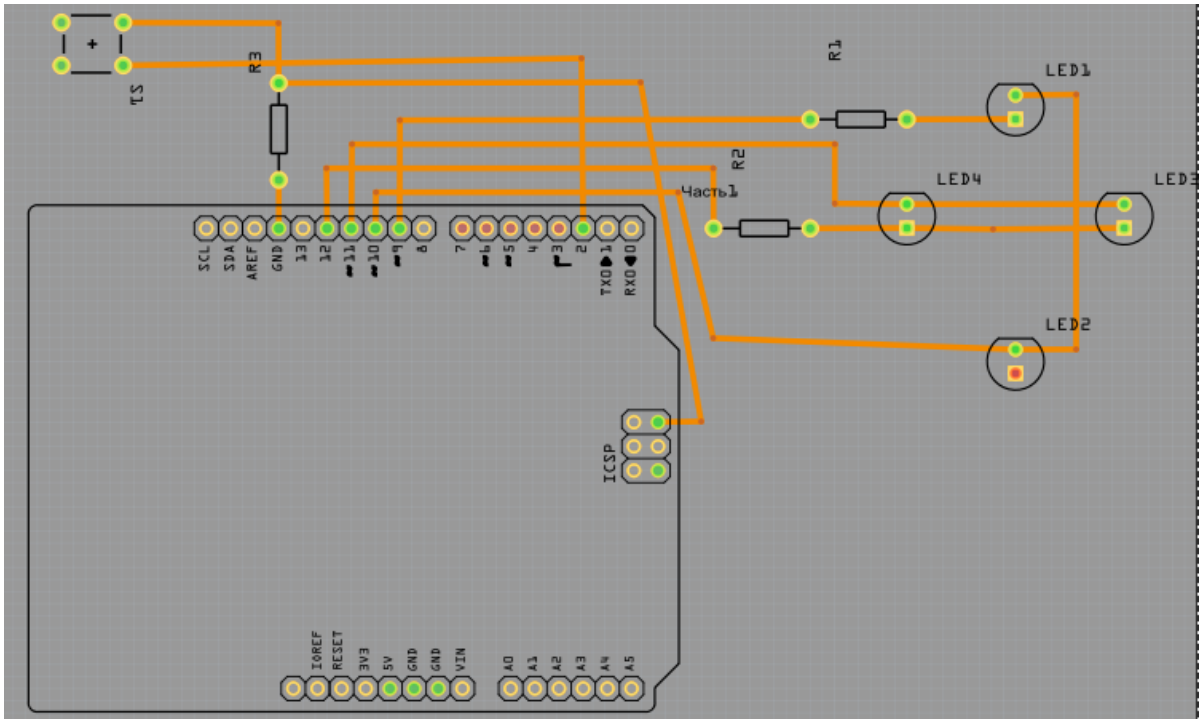
Задание № 1. Электрическая схема



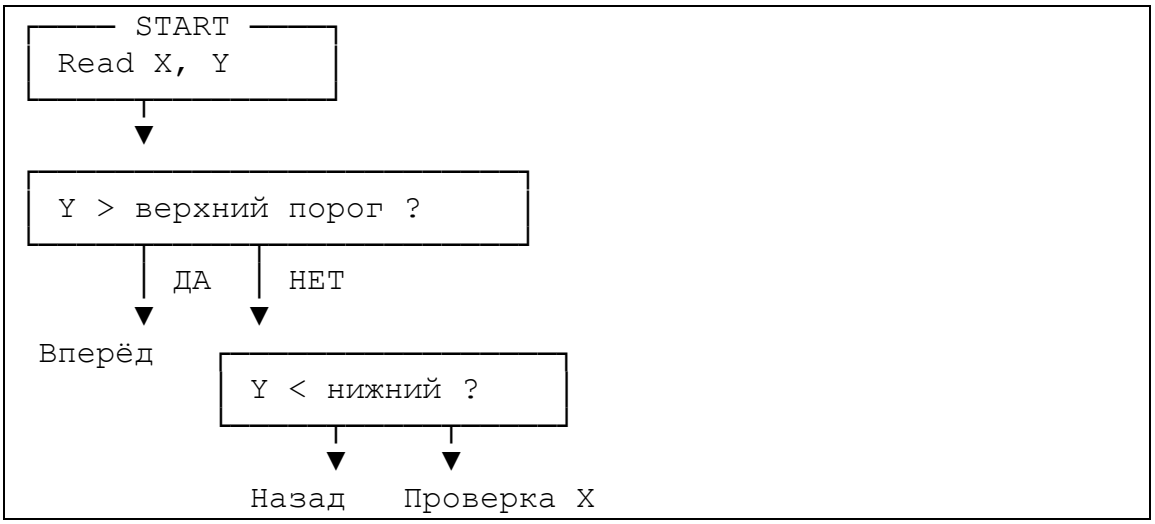
Задание № 2.1. Сборка электрической схемы



Задание № 2.2. Проектирование электрической схемы



Задание № 3. Блок-схема алгоритма

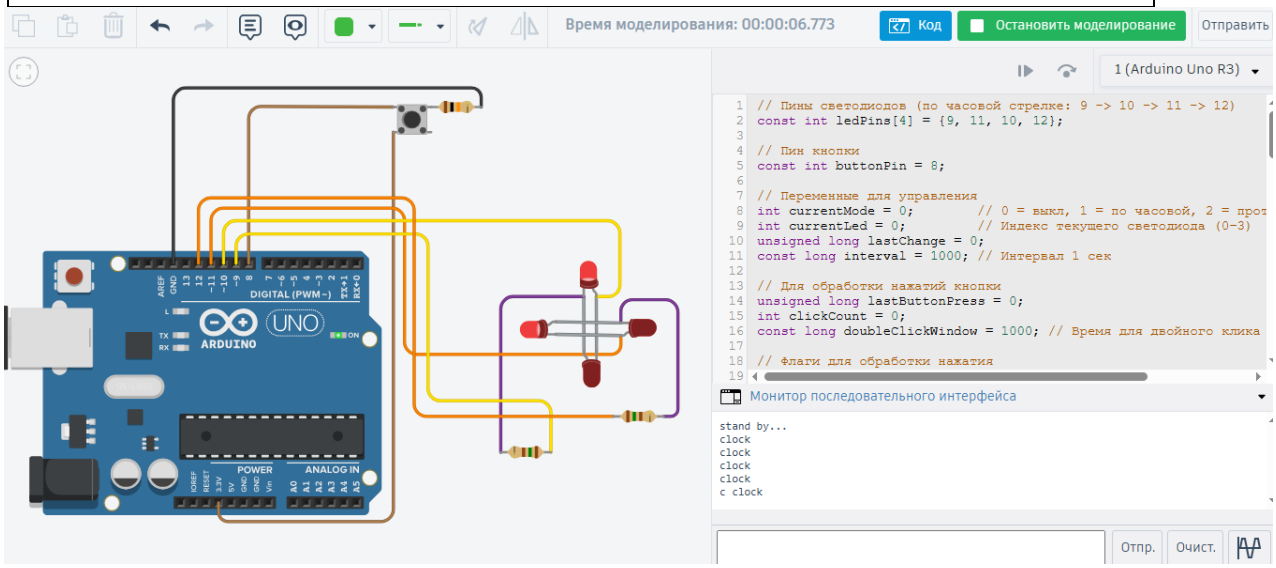


Задание № 4.1. Управляющая программа и 4.2. Монитор последовательного порта

```
// Пины светодиодов (по часовой стрелке: 9 -> 10 -> 11 ->
12)
const int ledPins[4] = {9, 11, 10, 12};

// Пин кнопки
const int buttonPin = 8;

// Переменные для управления
int currentMode = 0;        // 0 = выкл, 1 = по часовой, 2
= против часовой
int currentLed = 0;         // Индекс текущего светодиода
(0-3)
unsigned long lastChange = 0;
const long interval = 1000; // Интервал 1 сек

// Для обработки нажатий кнопки
unsigned long lastButtonPress = 0;
```

```
int clickCount = 0;
const long doubleClickWindow = 1000; // Время для двойного
клика (1 сек)

// Флаги для обработки нажатия
bool buttonPressed = false;
bool buttonWasPressed = false;

void setup() {
  // Инициализация светодиодов
  for (int i = 0; i < 4; i++) {
    pinMode(ledPins[i], OUTPUT);
    digitalWrite(ledPins[i], LOW);
  }

  // Инициализация кнопки с внутренней подтяжкой
  pinMode(buttonPin, INPUT_PULLUP);

  // Инициализация Serial
  Serial.begin(9600);
  Serial.println("stand by...");
}

void loop() {
  unsigned long currentTime = millis();

  // Обработка кнопки (без блокировки)
  bool isButtonPressed = digitalRead(buttonPin)== HIGH;

  if (isButtonPressed && !buttonPressed) {
    // Кнопка только что нажата (фронт)
    buttonPressed = true;
    unsigned long now = currentTime;

    if (now - lastButtonPress < doubleClickWindow) {
      clickCount++;
    } else {
      clickCount = 1;
    }
    lastButtonPress = now;

    // Обработка кликов
    if (clickCount == 1) {
      currentMode = 1;
      currentLed = 0;
      Serial.println("clock");
    } else if (clickCount == 2) {
      currentMode = 2;
      currentLed = 0;
      Serial.println("c clock");
      clickCount = 0; // Сброс счетчика после обработки
двойного клика
    }

  } else if (!isButtonPressed && buttonPressed) {
    // Кнопка только что отпущена
```

```
    buttonPressed = false;
    // Можно добавить здесь логику, если нужно
  }

  // Сброс счетчика кликов через 1.5 сек после последнего
нажатия
  if (clickCount > 0 && currentTime - lastButtonPress >
doubleClickWindow + 500) {
    clickCount = 0;
  }

  // Управление светодиодами
  if (currentMode != 0 && currentTime - lastChange >=
interval) {
    // Выключаем предыдущий
    digitalWrite(ledPins[currentLed], LOW);

    // Переходим к следующему
    if (currentMode == 1) {
      currentLed = (currentLed + 1) % 4;
    } else if (currentMode == 2) {
      currentLed = (currentLed - 1 + 4) % 4;
    }

    // Включаем новый
    digitalWrite(ledPins[currentLed], HIGH);
    lastChange = currentTime;
  }
}
```



Кейс 1. Вариант 11.

Джойстик робота на потенциометрах

Написать программу управления двумя мотор-редукторами робота колесного типа (колеса не показаны). Управление осуществляется с джойстика, реализованного на двух потенциометрах. При программировании предусмотреть следующие режимы управления вращением мотор-редукторов: вперед, назад, поворот вправо, поворот влево, остановка. *Дополнительно*. Реализовать вывод информации о текущем режиме управления в окно монитора порта. *Справочно*. При повороте ручки потенциометров необходимо предусмотреть, что полученные значения углов должны быть преобразованы в

соответствующие режимы вращения мотор-редукторов. Одним из вариантов решения является разбиение значений углов на нескольких диапазонов.

Оборудование:

- Мотор-редуктор 2 шт.
- Н-мост или диодный мост для управления двигателями 1 шт. (на микросхеме L293)
- Arduino UNO и кабель питания USB
- Макетная плата
- Потенциометр 2 шт.
- Соединительные провода и провода с зажимами «крокодил»

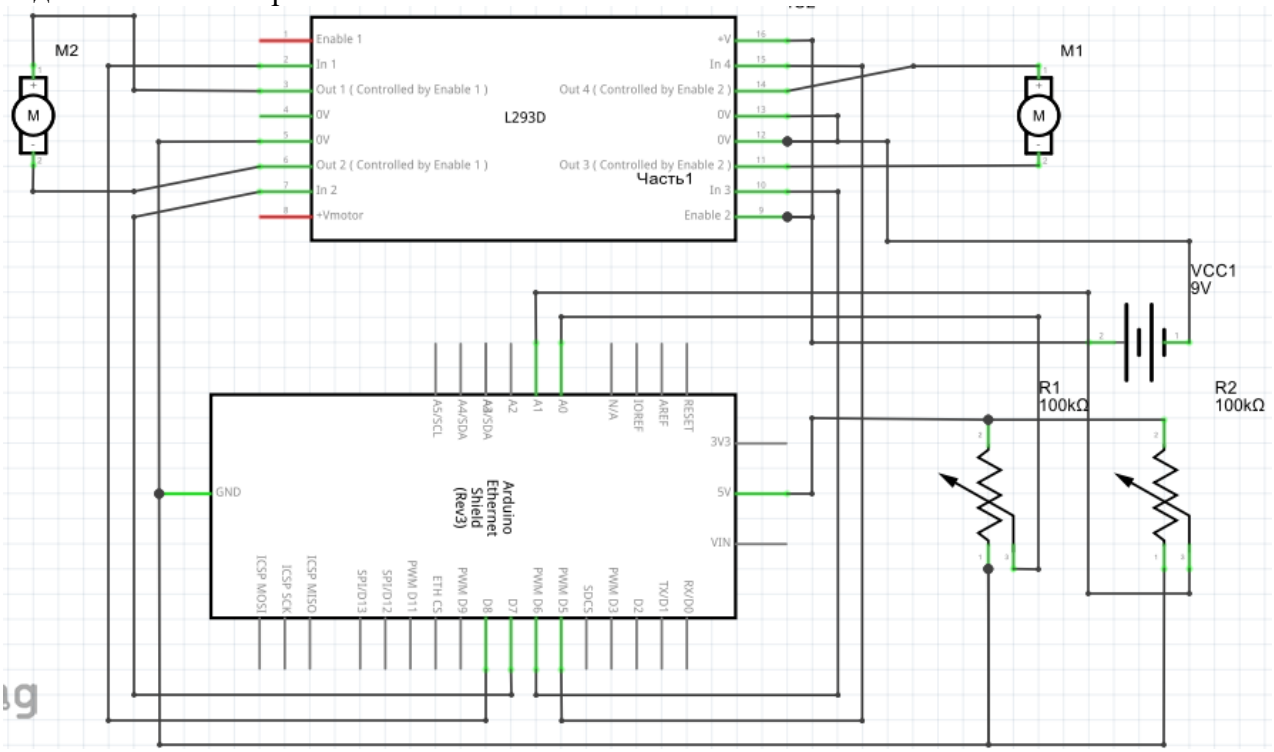
Решение.

Методические рекомендации к задаче и ответу.

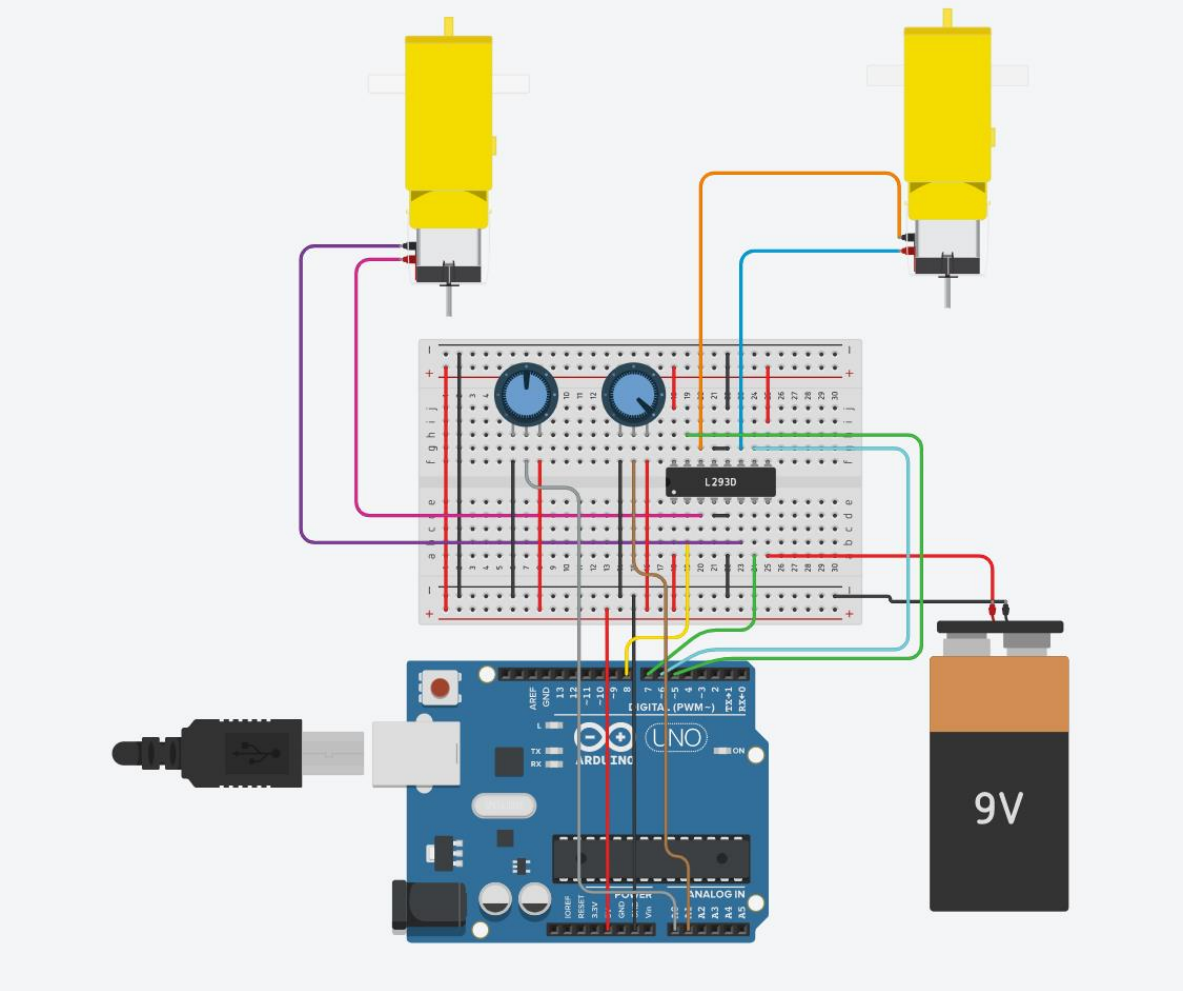
Условие. При проверке решения задачи проверяются знания режимов управления двигателями постоянного тока. Оценивается умение программирования режимов работы двигателями постоянного тока.

Ответ. Ответ должен содержать программный код, в котором фигурирует смена режимов работы двигателями постоянного тока.

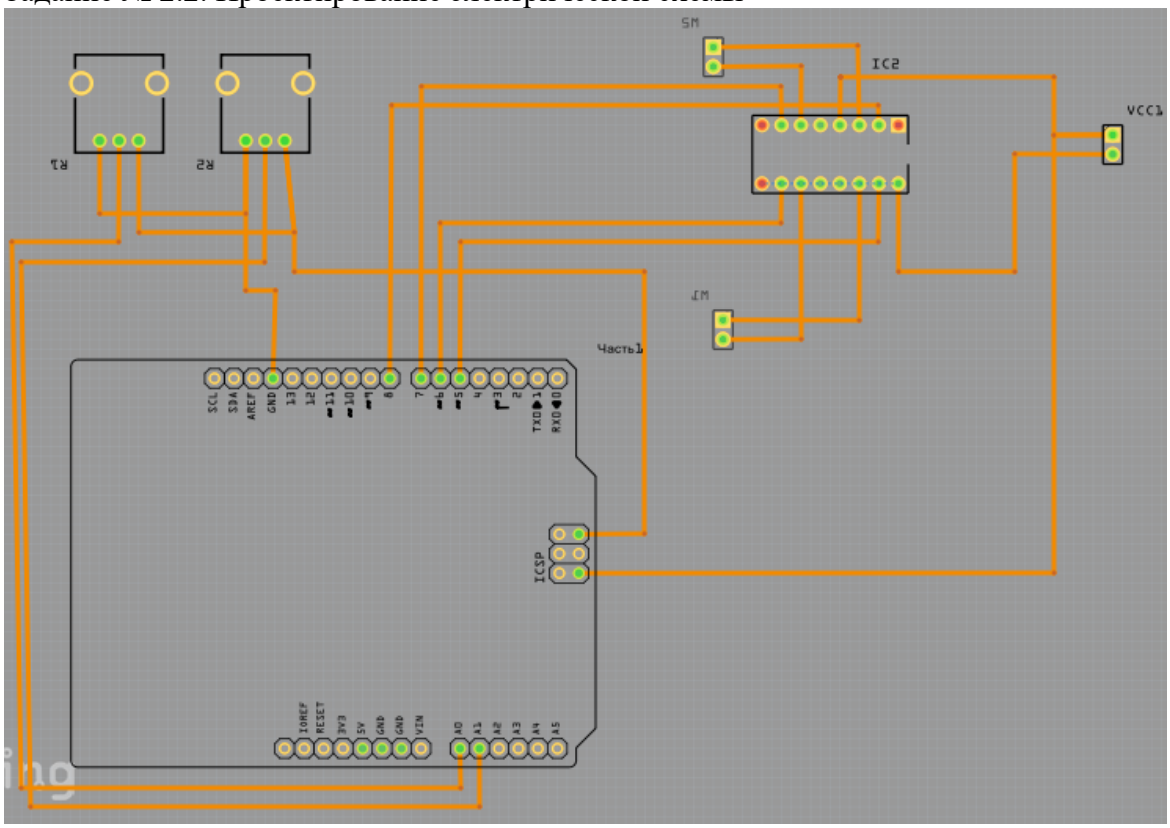
Задание № 1. Электрическая схема



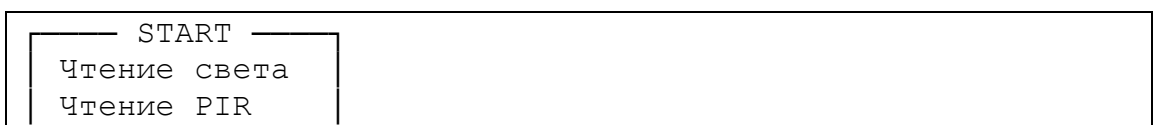
Задание № 2.1. Сборка электрической схемы

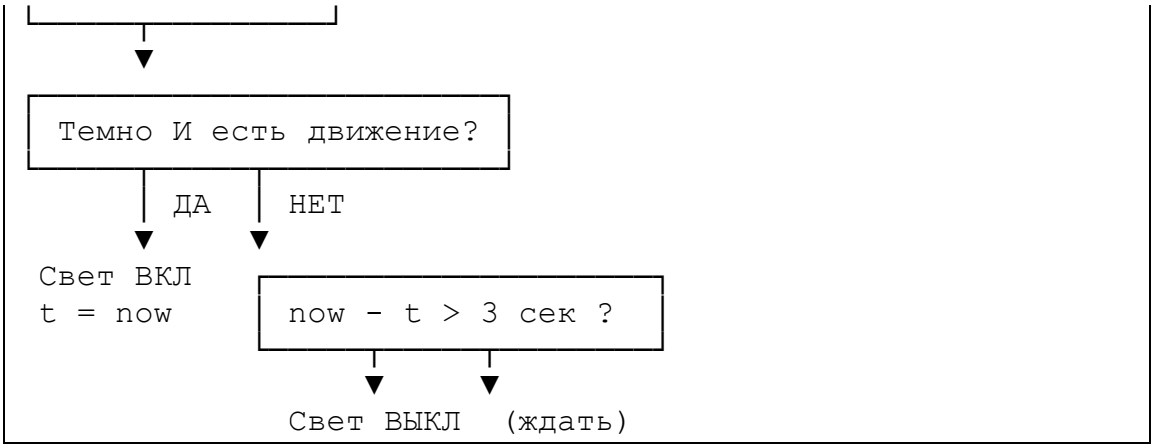


Задание № 2.2. Проектирование электрической схемы



Задание № 3. Блок-схема алгоритма

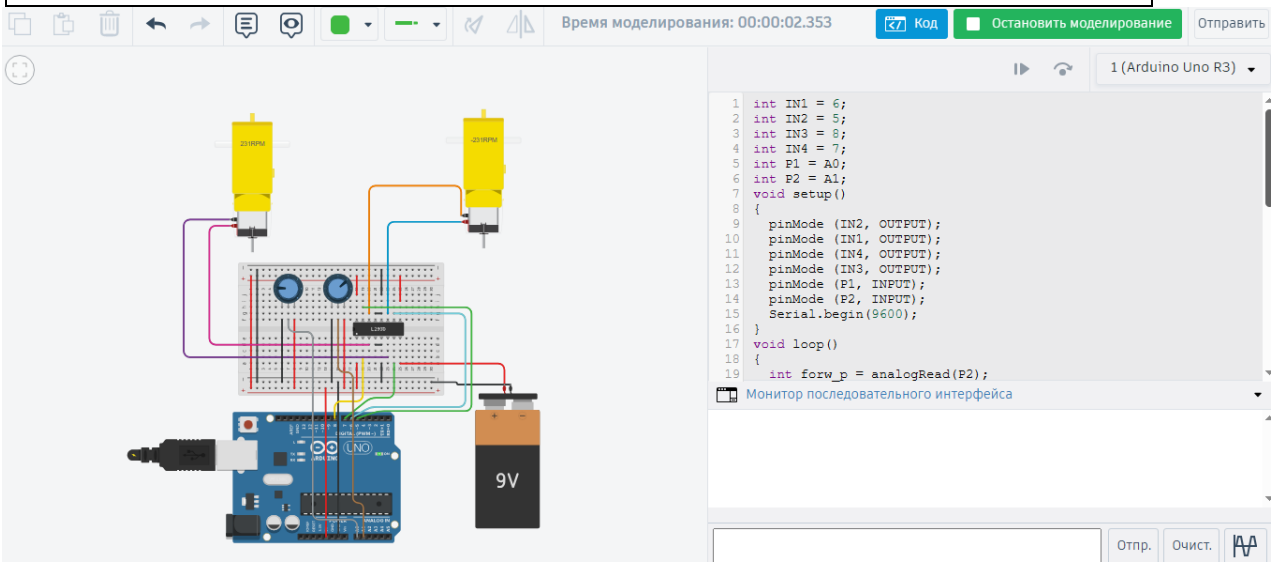




Задание № 4.1. Управляющая программа и 4.2. Монитор последовательного порта

```
int IN1 = 6;
int IN2 = 5;
int IN3 = 8;
int IN4 = 7;
int P1 = A0;
int P2 = A1;
void setup()
{
  pinMode (IN2, OUTPUT);
  pinMode (IN1, OUTPUT);
  pinMode (IN4, OUTPUT);
  pinMode (IN3, OUTPUT);
  pinMode (P1, INPUT);
  pinMode (P2, INPUT);
  Serial.begin(9600);
}
void loop()
{
  int forw_p = analogRead(P2);
  int rot_p = analogRead(P1);
  if (forw_p > 500 && rot_p > 500 && rot_p<524){
    r_forward();
    l_forward();
  }
  if(rot_p<500){
    l_forward();
    r_back();
  }
  if(rot_p>524){
    l_back();
    r_forward();
  }
}
void r_forward(){
  digitalWrite (IN2, LOW);
  digitalWrite (IN1, HIGH);
}
void l_forward(){
  digitalWrite (IN4, LOW);
  digitalWrite (IN3, HIGH);
}
```

```
void r_back(){
  digitalWrite (IN1, LOW);
  digitalWrite (IN2, HIGH);
}
void l_back(){
  digitalWrite (IN3, LOW);
  digitalWrite (IN4, HIGH);
}
```



Кейс 1. Вариант 12.

Умный светильник.

Необходимо написать программу управления светильником, который будет включаться только в темноте. Светильник включается и работает до тех пор, пока в зоне действия датчика находится человек (после ухода человека, светильник включен ещё 3 секунды и выключается). Внешние условия изменяются путем перемещения интерактивных точек на датчиках (черная и бирюзовая, как на иллюстрации к задаче ниже). *Дополнительно*. В окно монитора порта выводить время выключения светодиода. *Справочно*. Реализовывать задачу рекомендуется без использования команд задержек. Вместо задержек могут быть использованы команды millis().

Оборудование:

- Фоторезистор
- Светодиод
- Резистор 200-300 Ом
- Резистор 9-11 кОм
- Пироэлектрический инфракрасный датчик движения
- Соединительные провода
- Arduino UNO и кабель питания USB
- Макетная плата

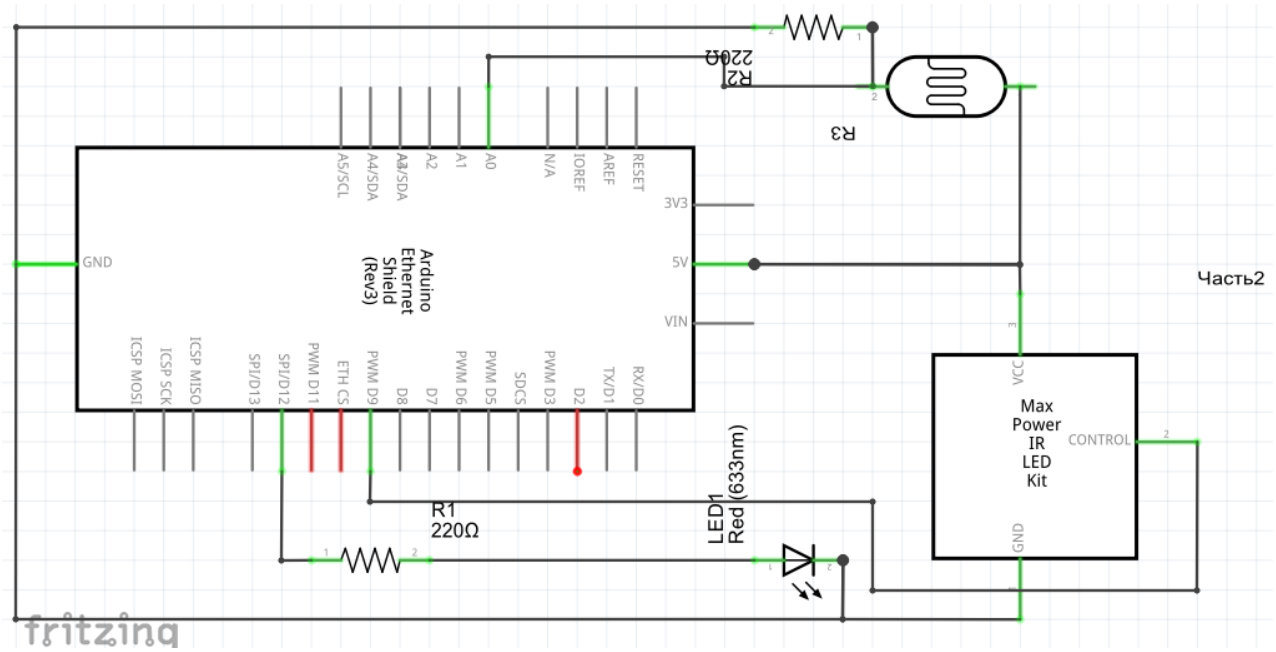
Решение.

Методические рекомендации к задаче и ответу.

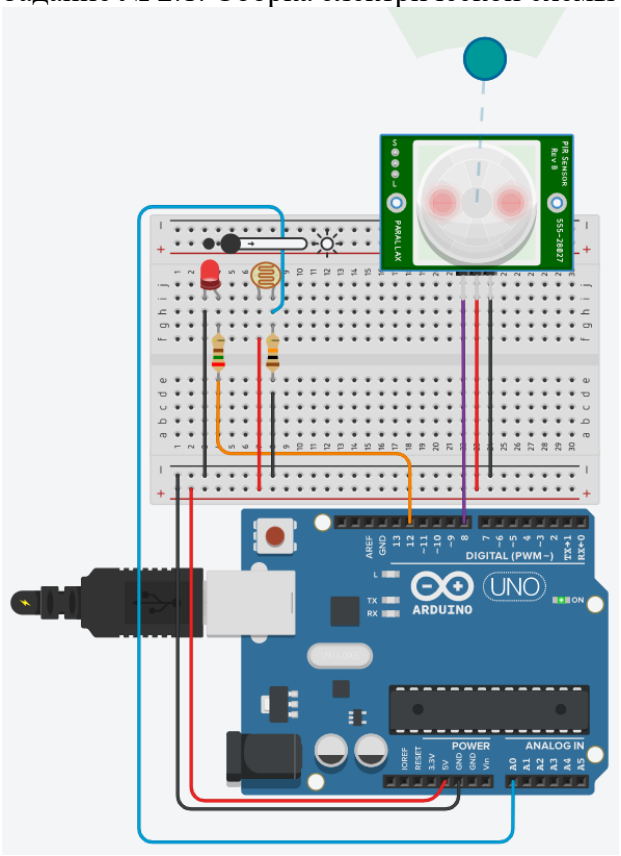
Условие. При проверке решения задачи проверяются знания основ системы управления и контроля доступом. Оценивается умение программирования определения наличия человека.

Ответ. Ответ должен содержать программный код, в котором фигурирует контроль наличия присутствия человека.

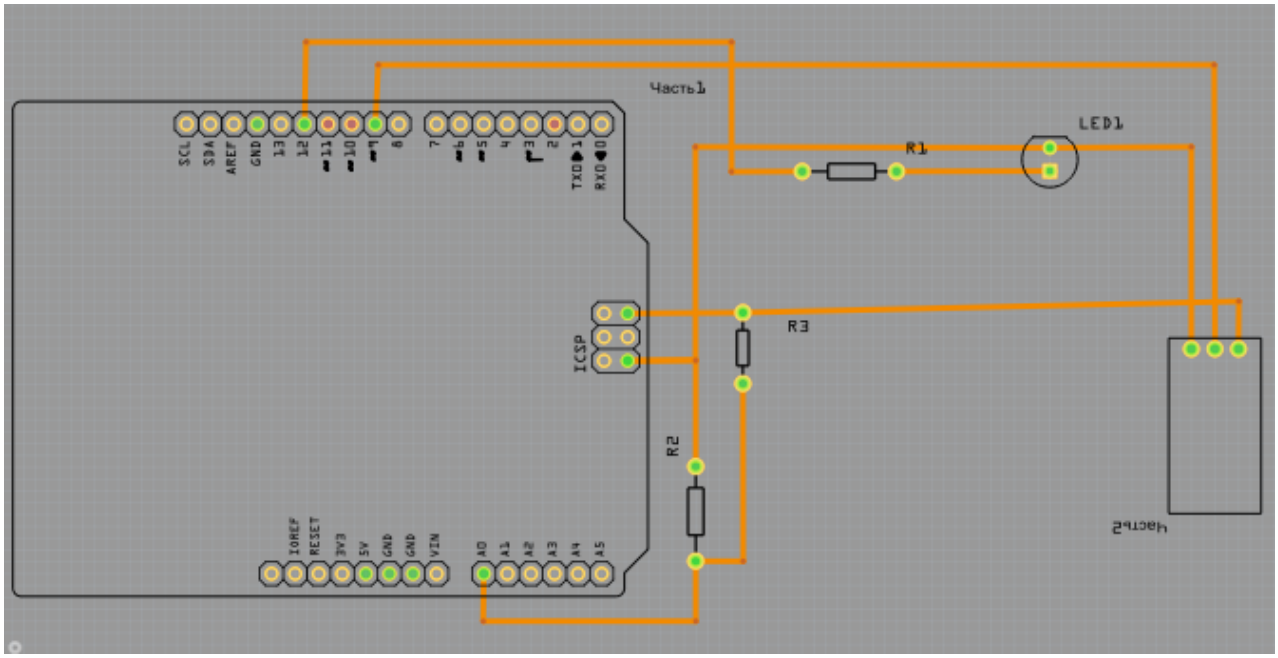
Задание № 1. Электрическая схема



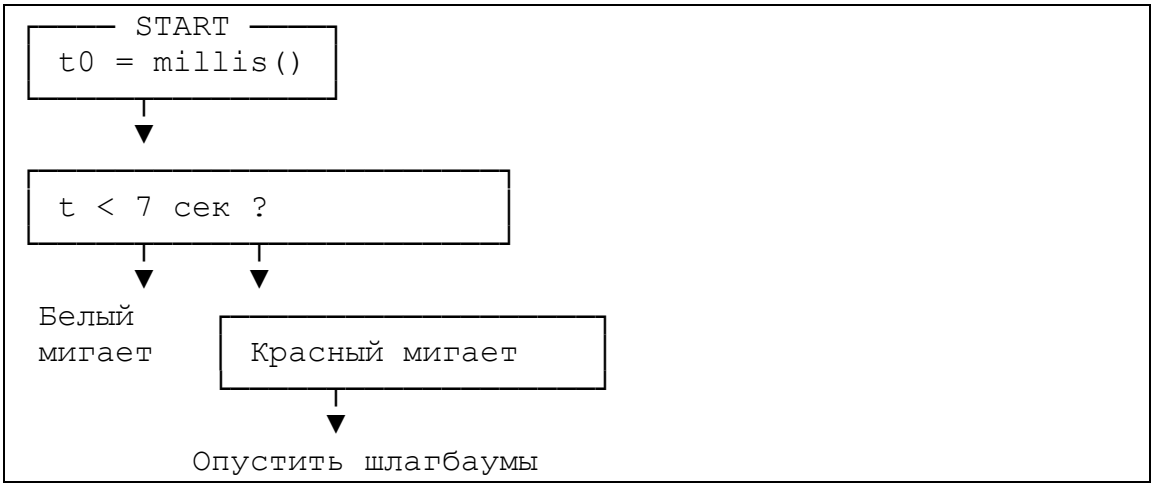
Задание № 2.1. Сборка электрической схемы



Задание № 2.2. Проектирование электрической схемы



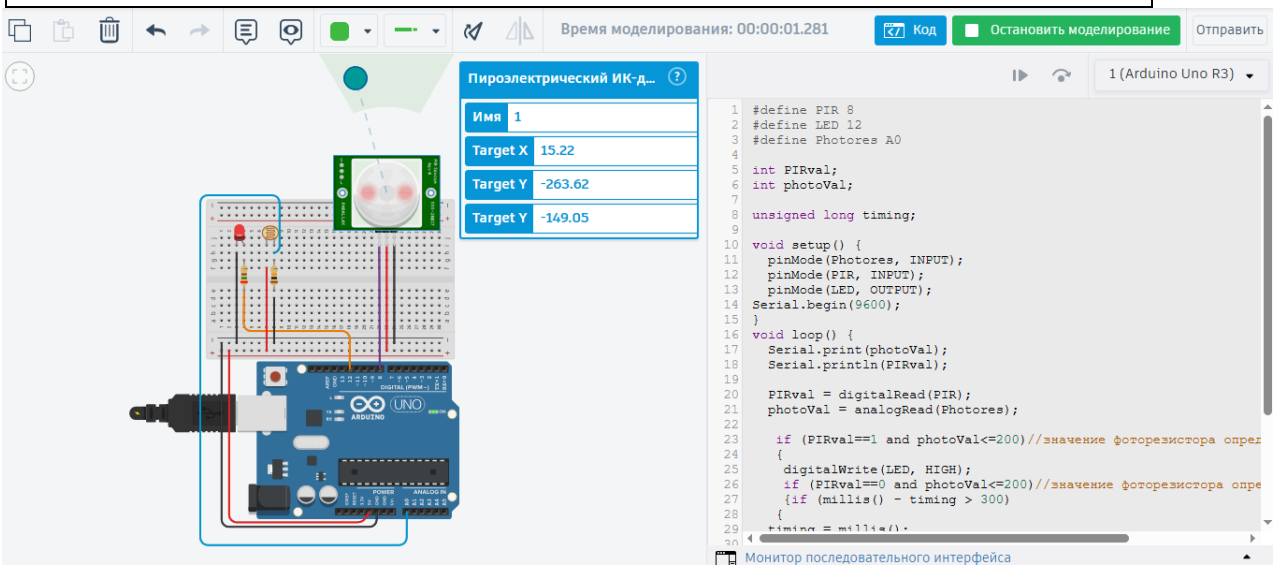
Задание № 3. Блок-схема алгоритма



Задание № 4.1. Управляющая программа и 4.2. Монитор последовательного порта

```
#define PIR 8
#define LED 12
#define Photores A0

int PIRval;
int photoVal;

unsigned long timing;

void setup() {
  pinMode(Photores, INPUT);
  pinMode(PIR, INPUT);
  pinMode(LED, OUTPUT);
Serial.begin(9600);
}
void loop() {
  Serial.print(photoVal);
  Serial.println(PIRval);

  PIRval = digitalRead(PIR);
```

```
  photoVal = analogRead(Photores);

   if (PIRval==1 and photoVal<=200)//значение
фоторезистора определяется эмпирически
    {
     digitalWrite(LED, HIGH);
     if (PIRval==0 and photoVal<=200)//значение
фоторезистора определяется эмпирически
     {if (millis() - timing > 300)
    {
  timing = millis();
  digitalWrite(LED, HIGH);
 }
   }
    }
  else
   {
     digitalWrite(LED, LOW);
   }
}
```



Кейс 1. Вариант 13.

Управление переездом.

Написать программу управления автоматикой железнодорожного переезда. Переезд снабжен: двумя шлагбаумами (на сервомоторах), двумя светофорами (на RGB-светодиодах), каждый из которых может светить белым и красным светом. При запуске программы, каждый из светодиодов по очереди мигает белым цветом с перерывом 1 сек. Через 7 секунд после запуска необходимо, чтобы светодиоды начинают мигать, также попеременно, красным светом, с перерывом 0.5 сек. Через 3 сек. после начала красного мигания светодиодов, должны опуститься шлагбаумы при этом светодиоды продолжают поочередно мигать красным светом. *Дополнительно*. По прошествии 7 секунд мигания красным светом поднимаются шлагбаумы. По прошествии 3 секунд после поднятия шлагбаума светодиоды переключаются на мигание белым светом с перерывом 1 сек. *Справочно*. Для реализации программы рекомендуется использовать функцию millis().

Оборудование:

- токоограничивающие резисторы 6 шт.
- 2 сервомотора
- 2 RGB светодиода
- Соединительные провода
- Arduino UNO и кабель питания USB

- Макетная плата

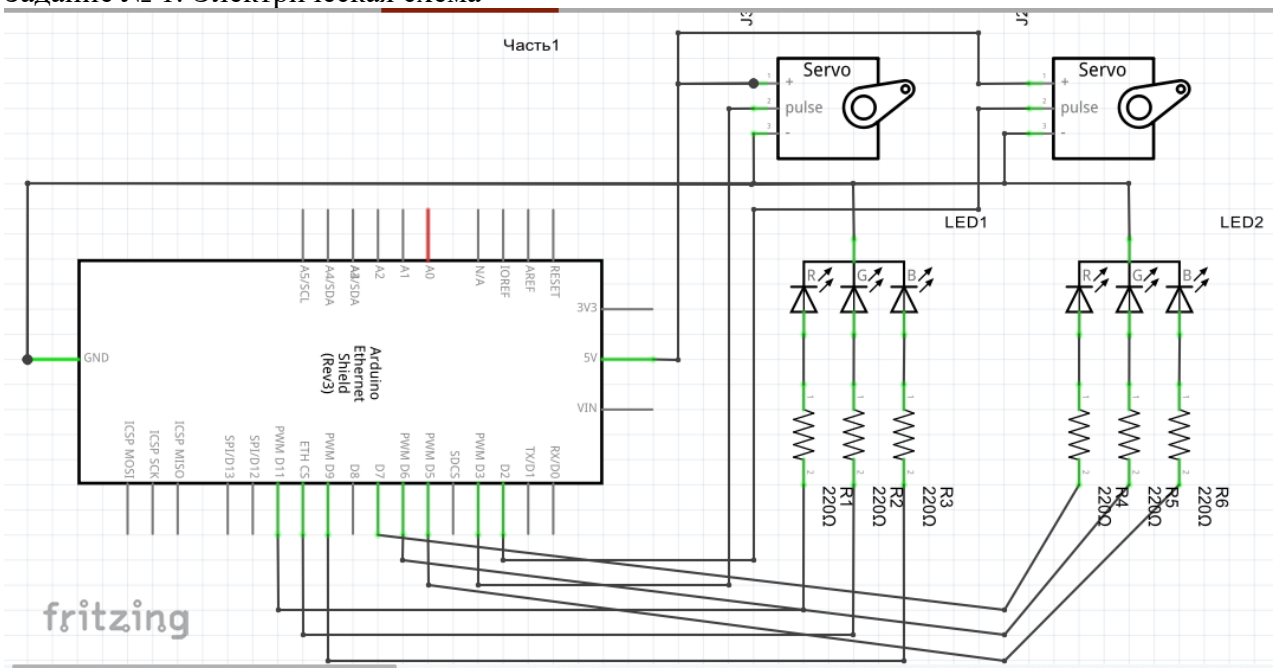
Решение.

Методические рекомендации к задаче и ответу.

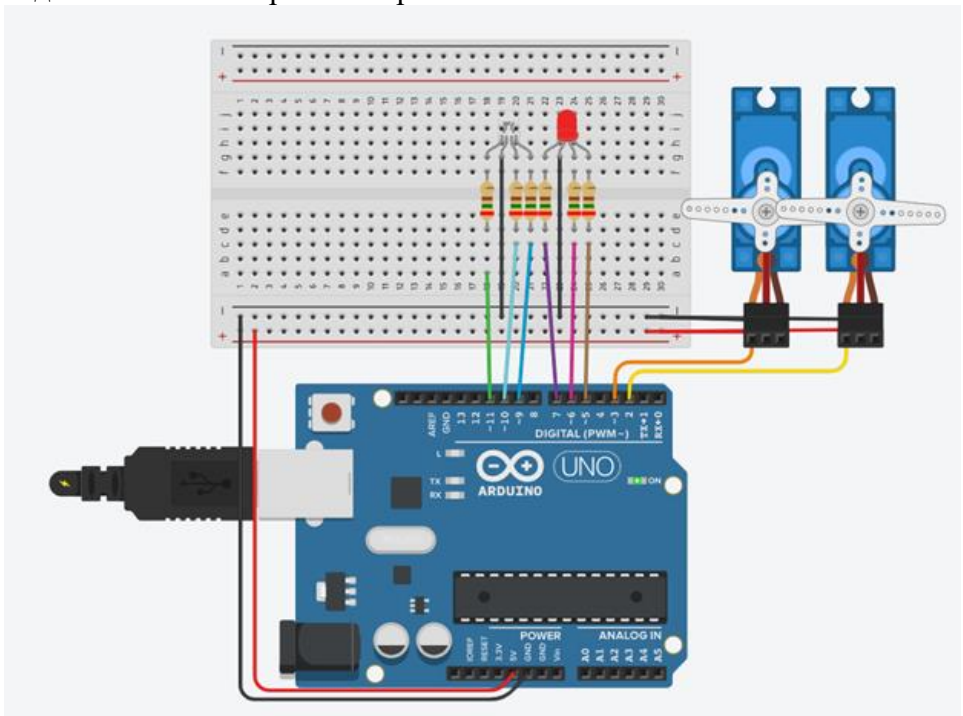
Условие. При проверке решения задачи проверяются знания основ расписанием включения-выключения устройств. Оценивается умение программирования расписания работы.

Ответ. Ответ должен содержать программный код, в котором фигурирует задача управления расписанием работы.

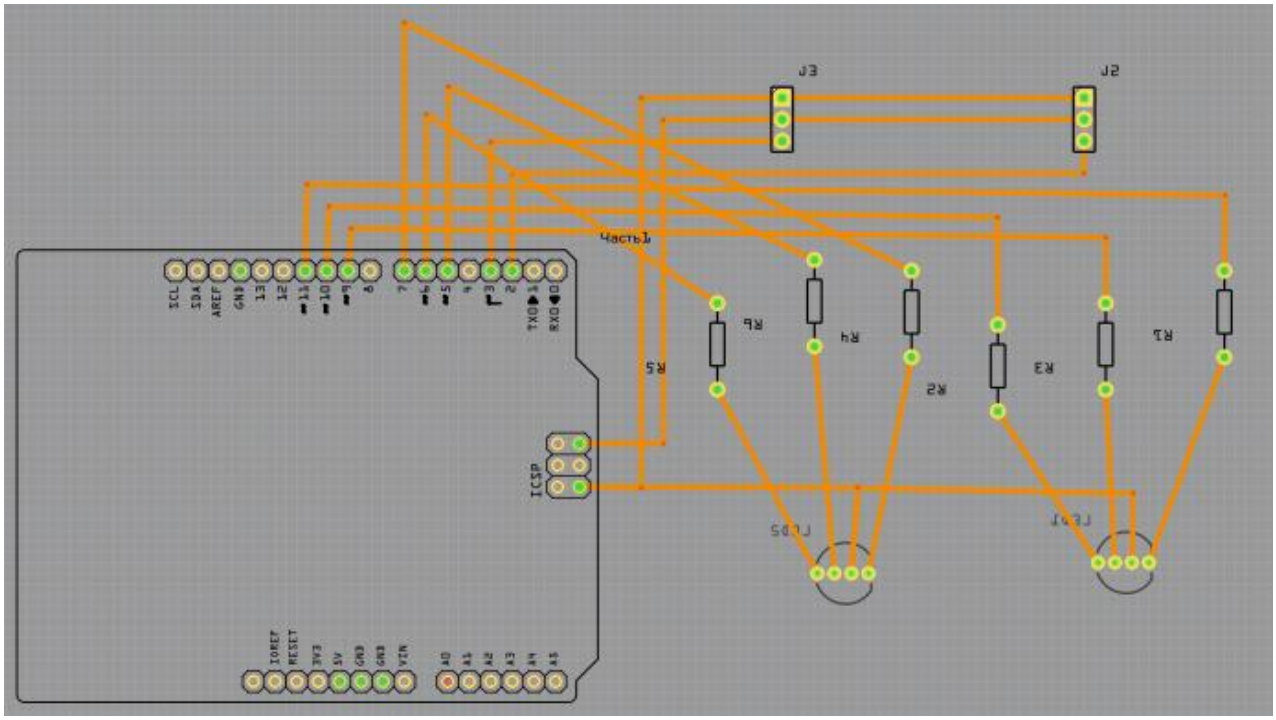
Задание № 1. Электрическая схема



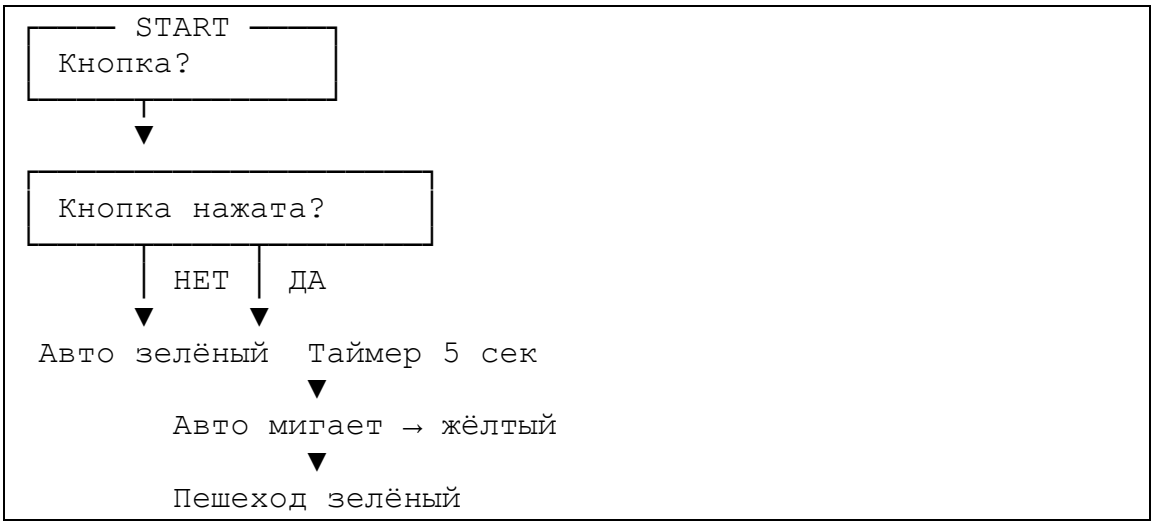
Задание № 2.1. Сборка электрической схемы



Задание № 2.2. Проектирование электрической схемы



Задание № 3. Блок-схема алгоритма



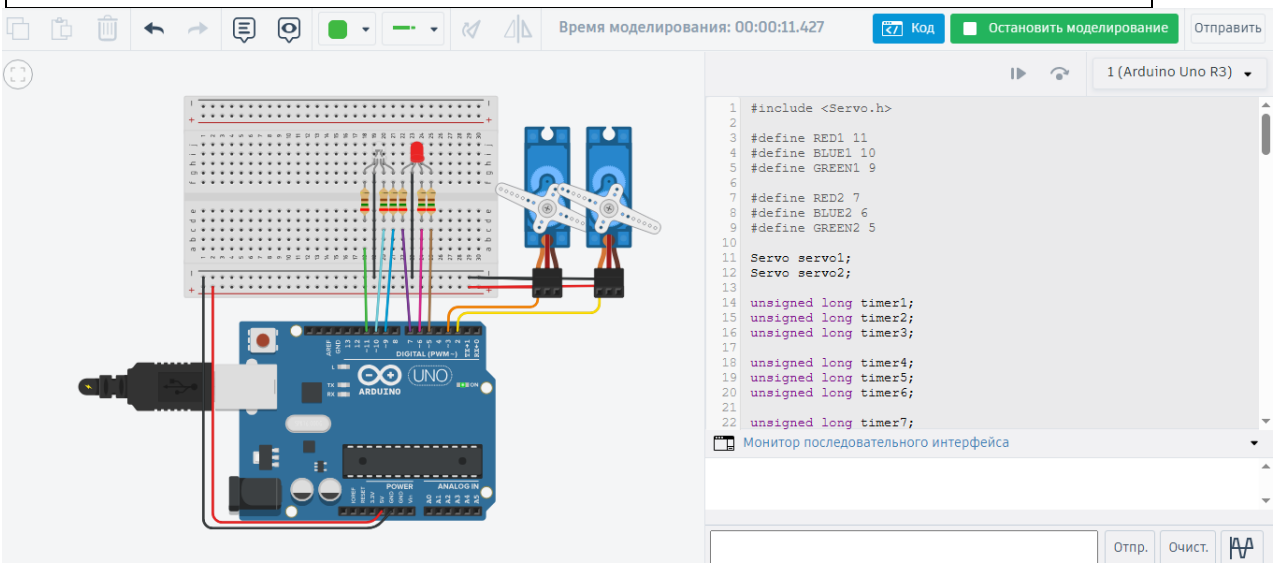
Задание № 4.1. Управляющая программа и 4.2. Монитор последовательного порта

```
#include <Servo.h>

#define RED1 11
#define BLUE1 10
#define GREEN1 9

#define RED2 7
#define BLUE2 6
#define GREEN2 5

Servo servo1;
Servo servo2;

unsigned long timer1;
unsigned long timer2;
unsigned long timer3;
```

```
unsigned long timer4;
unsigned long timer5;
unsigned long timer6;

unsigned long timer7;
unsigned long timer8;

bool ledState1 =HIGH;
bool ledState2 =LOW;

bool ledState3 =HIGH;
bool ledState4 =LOW;

void setup()
{
  pinMode(11, OUTPUT);
  pinMode(10, OUTPUT);
  pinMode(9, OUTPUT);

  pinMode(7, OUTPUT);
  pinMode(6, OUTPUT);
  pinMode(5, OUTPUT);

  servo1.attach(2);
  servo2.attach(3);

  servo1.write(0);
  servo2.write(0);
}

void loop()
{
   while(millis()< 7000)
  {
  if (millis() - timer1 > 1000)
  {
    timer1 = millis();
    ledState1=!ledState1;
    digitalWrite(RED1, ledState1);
    digitalWrite(BLUE1, ledState1);
    digitalWrite(GREEN1, ledState1);
  }

  if (millis() - timer2 > 1000)
  {
    timer2 = millis();
    ledState2=!ledState2;
    digitalWrite(RED2, ledState2);
    digitalWrite(BLUE2, ledState2);
    digitalWrite(GREEN2, ledState2);
  }
   }
  while(7000<millis()&&millis()< 11000)
  {
   if (millis() - timer5 > 500)
  {
```

```
  timer5 = millis();
  ledState1=!ledState1;
  digitalWrite(RED1, ledState1);
  digitalWrite(BLUE1, LOW);
  digitalWrite(GREEN1, LOW);
}
  if (millis() - timer6 > 500)
{
  timer6 = millis();
  ledState2=!ledState2;
  digitalWrite(RED2, ledState2);
  digitalWrite(BLUE2, LOW);
  digitalWrite(GREEN2, LOW);
}
}
  while(11000<millis() && millis() < 18000)
{
      servo1.write(90);
      servo2.write(90);

 if (millis() - timer5 > 500)
{
  timer5 = millis();
  ledState1=!ledState1;
  digitalWrite(RED1, ledState1);
  digitalWrite(BLUE1, LOW);
  digitalWrite(GREEN1, LOW);
}
  if (millis() - timer6 > 500)
{
  timer6 = millis();
  ledState2=!ledState2;
  digitalWrite(RED2, ledState2);
  digitalWrite(BLUE2, LOW);
  digitalWrite(GREEN2, LOW);
}
}
 while(18000<millis() && millis() < 21000)
{
      servo1.write(0);
      servo2.write(0);

 if (millis() - timer5 > 500)
{
  timer5 = millis();
  ledState1=!ledState1;
  digitalWrite(RED1, ledState1);
  digitalWrite(BLUE1, LOW);
  digitalWrite(GREEN1, LOW);
}
  if (millis() - timer6 > 500)
{
  timer6 = millis();
  ledState2=!ledState2;
  digitalWrite(RED2, ledState2);
  digitalWrite(BLUE2, LOW);
```

```
    digitalWrite(GREEN2, LOW);
  }
   }
    if (millis()==21000)
    {
    digitalWrite(RED1, LOW);
    digitalWrite(BLUE1, LOW);
    digitalWrite(GREEN1, LOW);

    digitalWrite(RED2, LOW);
    digitalWrite(BLUE2, LOW);
    digitalWrite(GREEN2, LOW);
    }
    while(millis()>21000)
  {
  if (millis() - timer7 > 1000)
  {
    timer7 = millis();
    ledState1=!ledState1;
    digitalWrite(RED1, ledState1);
    digitalWrite(BLUE1, ledState1);
    digitalWrite(GREEN1, ledState1);
  }
  if (millis() - timer8 > 1000)
  {
    timer8 = millis();
    ledState2=!ledState2;
    digitalWrite(RED2, ledState2);
    digitalWrite(BLUE2, ledState2);
    digitalWrite(GREEN2, ledState2);
  }
  }
}
```



Кейс 1. Вариант 14.

Светофор для перехода пешеходов.

Написать программу регулирования работы светофоров на пешеходном переходе. По умолчанию автомобильный светофор горит зелёным, а пешеходный красным. Пешеход чтобы перейти нажимает кнопку. По прошествии 5 секунд после нажатия кнопки, светофор для автомобилей начинает мигать зелёным (2 мигания с задержкой 0,5 сек), при этом для

пешеходов продолжает гореть красный. После этого автомобильный светофор переключается на жёлтый и горит 2 секунды (пешеходный всё ещё горит красным). После выключения жёлтого, на автомобильном светофоре, загорается красный свет. В это же время на пешеходном светофоре на 10 секунд включается зелёный сигнал. По прошествии 10 секунд, на пешеходном светофоре зеленый начинает мигать (2 мигания с задержкой 0,5 сек) и переключается на красный. На автомобильном загорается жёлтый на 2 секунды, затем зелёным. *Дополнительно*. По прошествии 7 секунд реализовать вывод сообщения в окно монитора порта, разрешающего пешеходу нажать кнопку.

Оборудование:

- 2 красных светодиода
- 2 зелёных светодиода
- 1 жёлтый светодиод
- 1 кнопка
- Резистор 9-11 кОм.
- Резисторы 200-300 Ом (5 штук)
- Соединительные провода
- Arduino UNO и кабель питания USB
- Макетная плата

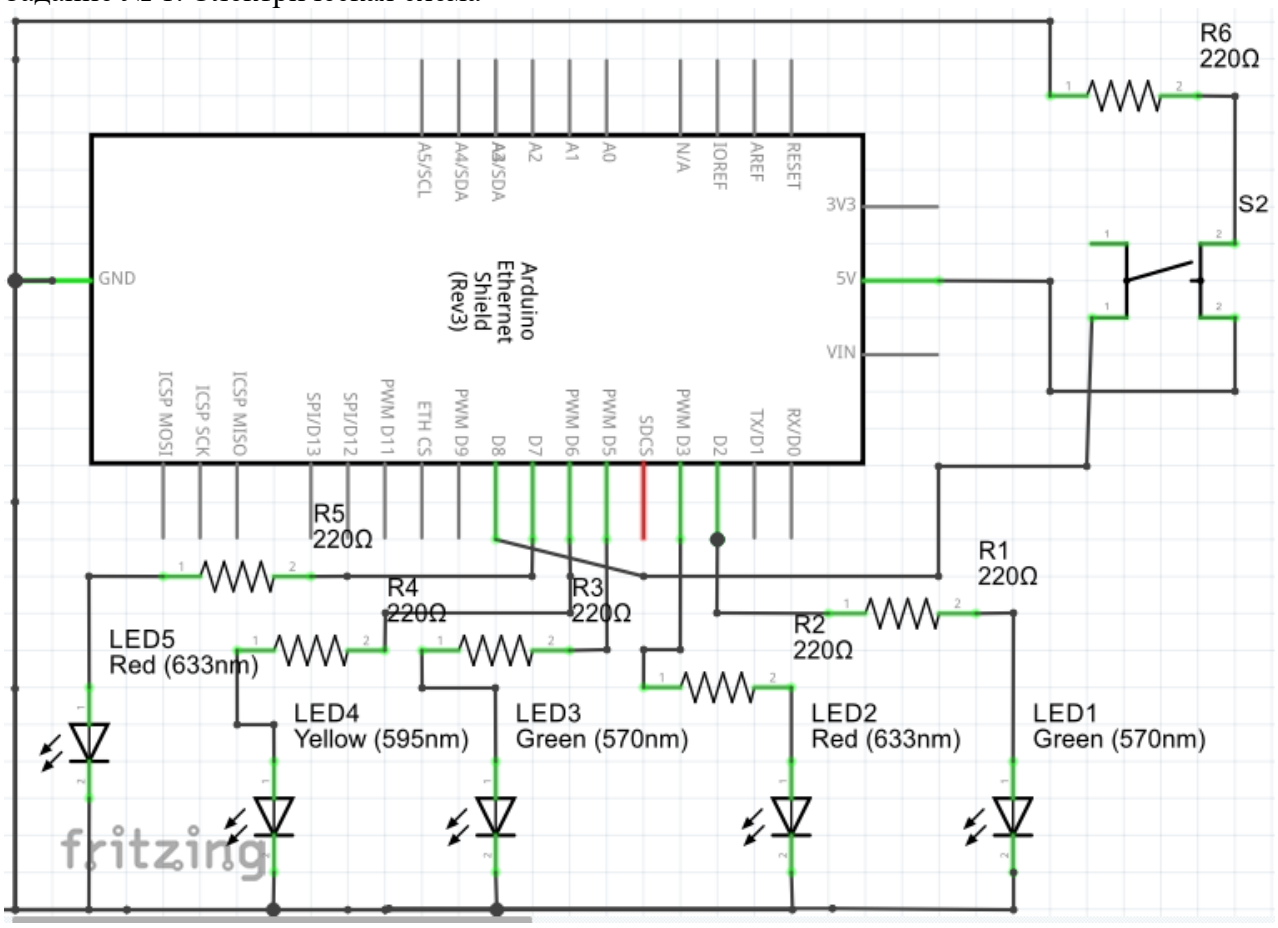
Решение.

Методические рекомендации к задаче и ответу.

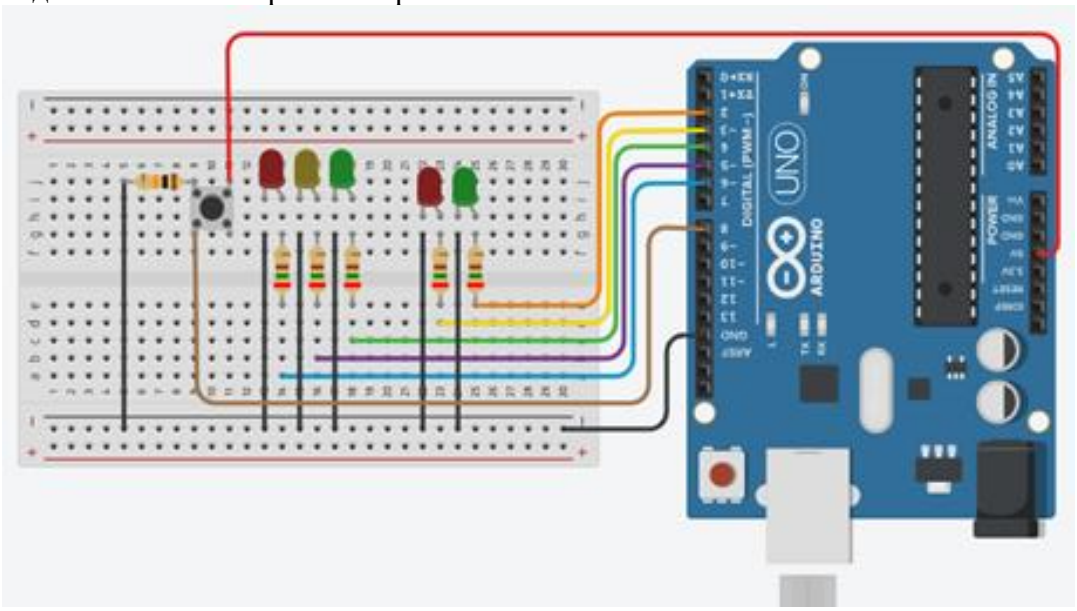
Условие. При проверке решения задачи проверяются знания основ построения расписания работы. Оценивается умение программирования расписания включения-выключения оборудования.

Ответ. Ответ должен содержать программный код, в котором фигурирует систему расписания.

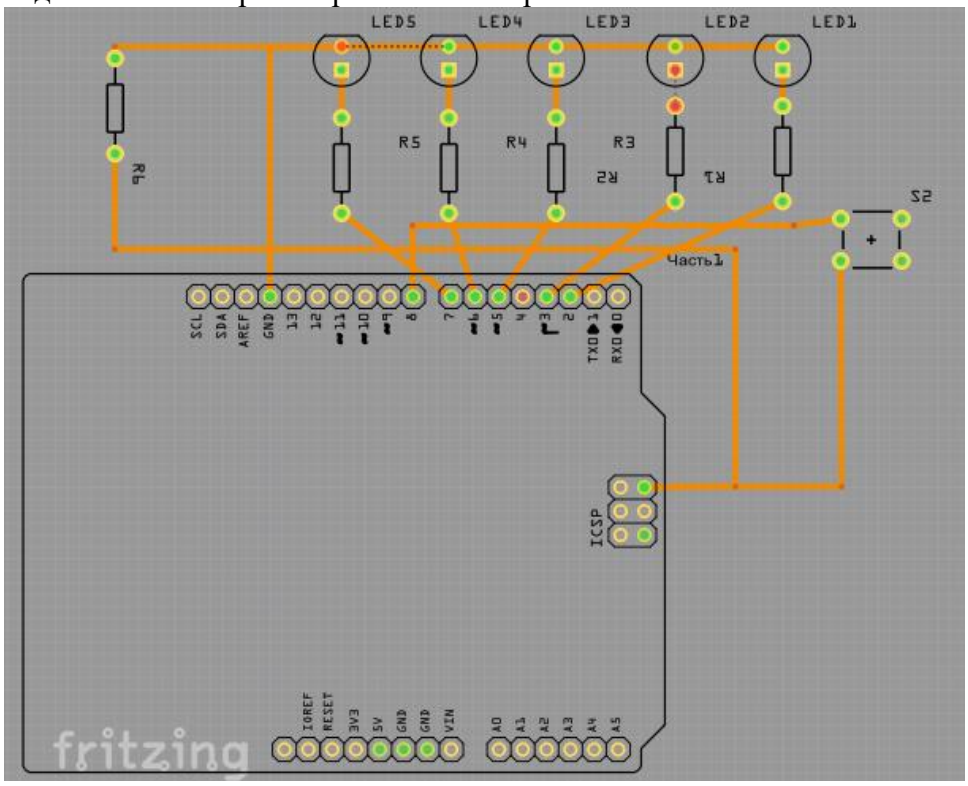
Задание № 1. Электрическая схема



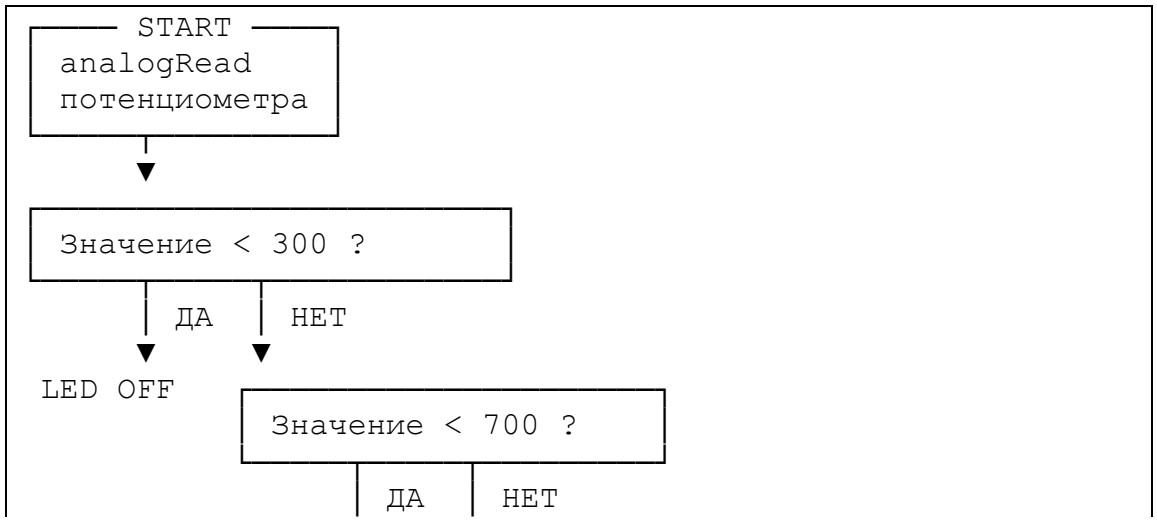
Задание № 2.1. Сборка электрической схемы



Задание № 2.2. Проектирование электрической схемы



Задание № 3. Блок-схема алгоритма

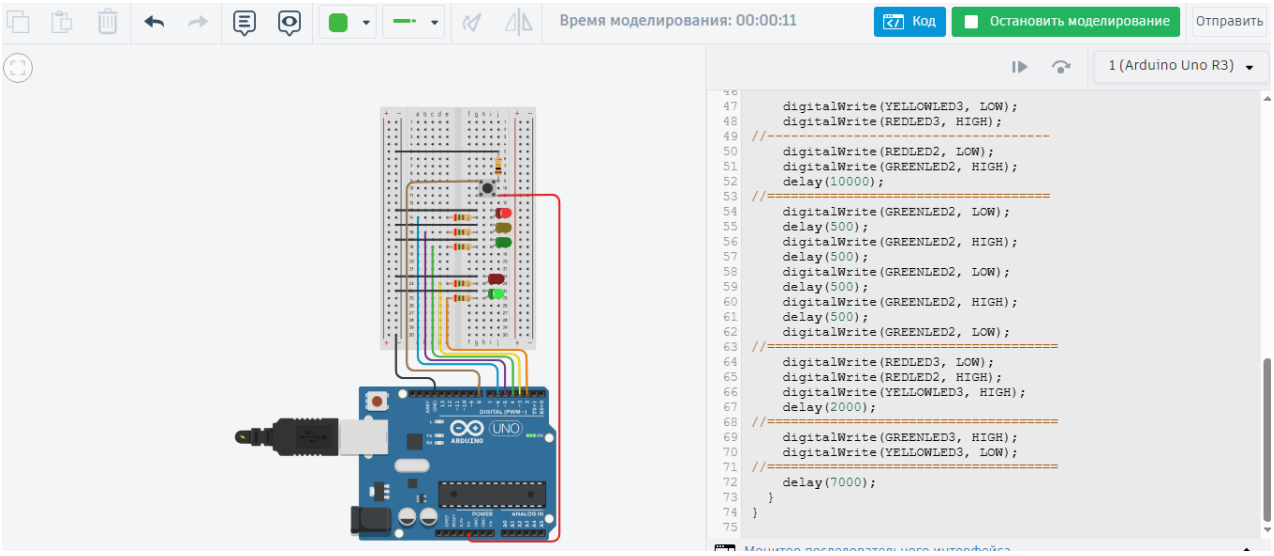


```
                    ▼     ▼
          Вращение CW   Вращение CCW
```

Задание № 4.1. Управляющая программа и 4.2. Монитор последовательного порта

```
#define REDLED3 6
#define YELLOWLED3 5
#define GREENLED3 4
#define REDLED2 3
#define GREENLED2 2
#define BUTTON 8

void setup()
{
  pinMode(REDLED3, OUTPUT);
  pinMode(YELLOWLED3, OUTPUT);
  pinMode(GREENLED3, OUTPUT);

  pinMode(REDLED2, OUTPUT);
  pinMode(GREENLED2, OUTPUT);

  pinMode(BUTTON, INPUT);

  digitalWrite(GREENLED3, HIGH);
  digitalWrite(REDLED2, HIGH);
}

void loop()
{

bool mode = digitalRead(BUTTON);
  if (mode == 1)
  {
  digitalWrite(GREENLED3, HIGH);
  digitalWrite(REDLED2, HIGH);
  delay(5000);
//===================================
    digitalWrite(GREENLED3, LOW);
    delay(500);
    digitalWrite(GREENLED3, HIGH);
    delay(500);
    digitalWrite(GREENLED3, LOW);
    delay(500);
    digitalWrite(GREENLED3, HIGH);
    delay(500);
    digitalWrite(GREENLED3, LOW);
    delay(500);
//===================================
    digitalWrite(YELLOWLED3, HIGH);
    delay(2000);

    digitalWrite(YELLOWLED3, LOW);
    digitalWrite(REDLED3, HIGH);
//-----------------------------------
    digitalWrite(REDLED2, LOW);
    digitalWrite(GREENLED2, HIGH);
```

```
    delay(10000);
//====================================
    digitalWrite(GREENLED2, LOW);
    delay(500);
    digitalWrite(GREENLED2, HIGH);
    delay(500);
    digitalWrite(GREENLED2, LOW);
    delay(500);
    digitalWrite(GREENLED2, HIGH);
    delay(500);
    digitalWrite(GREENLED2, LOW);
//====================================
    digitalWrite(REDLED3, LOW);
    digitalWrite(REDLED2, HIGH);
    digitalWrite(YELLOWLED3, HIGH);
    delay(2000);
//====================================
    digitalWrite(GREENLED3, HIGH);
    digitalWrite(YELLOWLED3, LOW);
//====================================
    delay(7000);
  }
}
```



Решения заданий ИМ, IT класс, робототехника, практика, кейс 2, вариантов 14

Кейс 2. Вариант 1.
Условие
В среде программирования Trick Studio в программе на языке Python задаются данные для перемещения мобильного трехколесного робота (или блоками). Программа должна по полученным данным вычислить среднюю скорость перемещения робота за заданное время при равномерном перемещении робота по траектории из четырех точек. Необходимо написать программу, в которую включить заданные точки маршрута для того, чтобы программа рассчитала и вывела значения длинны маршрута и средней скорости робота. Задание реализуется либо в Trick Studio, либо на реальном роботе.
Формат ввода

При написании программы необходимо учесть следующие заданные координаты точек маршрута:
x0,y0 – координаты нулевой точки (A) маршрута, м.;
x1,y1 – координаты первой точки (B) маршрута, м.;
x2,y2 – координаты второй точки (C) маршрута, м.;
x3,y3 – координаты третьей точки (D) маршрута, м.;
Робот движется равномерно по траектории. Параметры координат точек маршрута и времени его прохождения задаются в основном коде программы в виде не целых чисел. В каждой строке с координатами два числа, первое x, второе y. Числа в строке разделены одинарным пробелом. В пятой строке указывается одно число – время. Размерность координат – метры, времени – секунды и миллисекунды (ограничение в два знака после точки). Границы координат не выходят за пределы от 0 до 5 метров по x и y.
Пример задания входных данных (x0,y0), (x1,y1), (x2,y2), (x3,y3), t для программы:
1.0 1.0
2.0 2.0
3.0 3.0
4.0 4.0
1.5
Участники самостоятельно разрабатывают программный код, содержащий разделы считывания заданных координат, промежуточных расчетов и вывода данных на дисплей робота вычисленных расстояния между точками и скорости в м/с. Необходимо использовать следующие формулы для расчета параметров скорости и расстояния между всеми точками маршрута:
$V = \frac{L}{t}$, где V - скорость (м/с), L - длинна пути маршрута (м), t - время (сек),
|AB|= $\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$, где √ - корень квадратный, |AB| - модуль расстояния между соседними точками A и B с координатами (x1,y1), (x2,y2) соответственно. Остальные расстояния и формулы для участков BC, CD и DE задаются самостоятельно.
Формулы необходимо самостоятельно преобразовать к программному виду и включить в программу. Обозначения параметров функций и переменных участники могут выбирать самостоятельно.

Формат вывода

Необходимо чтобы робот проехал по заданным точкам в среде симуляции (или в реальности). Необходимо приложить к ответу видеозапись симуляции перемещения робота из Trick Studio. Код предоставлять в трех форматах: 1) скриншотами в отчете; 2) блоками в Trick Studio сохранен проект с работающей симуляцией; 3) в среде Trick Studio, но на языке Python.
Программа должна вывести две строки. В первой строке должно вывестись вычисленная программой длинна всего маршрута в виде вещественного числа с двумя знаками после запятой, единица измерения метры. Во второй строке должно вывестись вычисленная программой скорость робота в виде вещественного числа с двумя знаками после запятой, единица измерения метры в секунду (м/с). Необходимо предусмотреть только вывод чисел. В качестве разделителя целой и вещественной части – точка. Текст комментариев и единицы измерения выводить не нужно.
Пример того, что должна вывести программа (числа не связанны с данными, указанными в разделе "Формат ввода"):
9.44
3.03

Исходные точки траектории движения.
Задание входных данных (x0,y0), (x1,y1), (x2,y2), (x3,y3), t для программы:
1.0 1.0
2.0 2.0
3.0 3.0
4.0 4.0
1.5

Задание № 1. Вычисление параметров движения

Расчтеные данные движения:

Участки пути td. Путь S. Скорость V. Преобразоание угловых перемещений ena и en.

td1=atan((2.0-1.0)/(2.0-1.0))
td2=atan((3.0-2.0)/(3.0-2.0))
td3=atan((4.0-3.0)/(4.0-3.0))
D_ab=sqrt((2.0-1.0)^2+(2.0-1.0)^2)
D_bc=sqrt((3.0-2.0)^2+(3.0-2.0)^2)
D_cd=sqrt((4.0-3.0)^2+(4.0-3.0)^2)
S=D_ab+D_bc+D_cd
V=S/1.5
A_ab=15.4/2*(0.45*pi-td1)
B_bc=15.4/2*(td1+0.001*pi-td2)
C_cd=15.4/2*(td2+0.001*pi-td3)
ena1=A_ab*360/pi/5.6
ena2=B_bc*360/pi/5.6
ena3=C_cd*360/pi/5.6
en1=D_ab*360/pi/5.6
en2=D_bc*360/pi/5.6
en3=D_cd*360/pi/5.6

Задание № 2. Блок-схема алгоритма

Задание № 3.1. Управляющая программа (в базовом варианте – управление движением по времени)

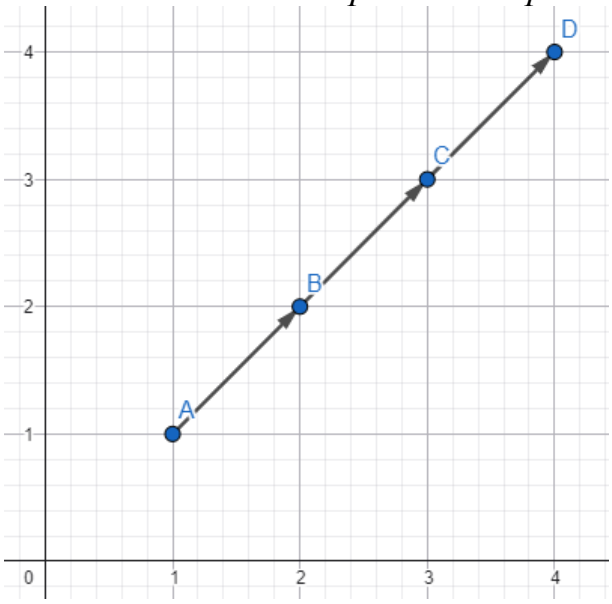
Задание № 3.2. Управляющая программа (в усложненном варианте – управление движением по показаниям энкодеров)

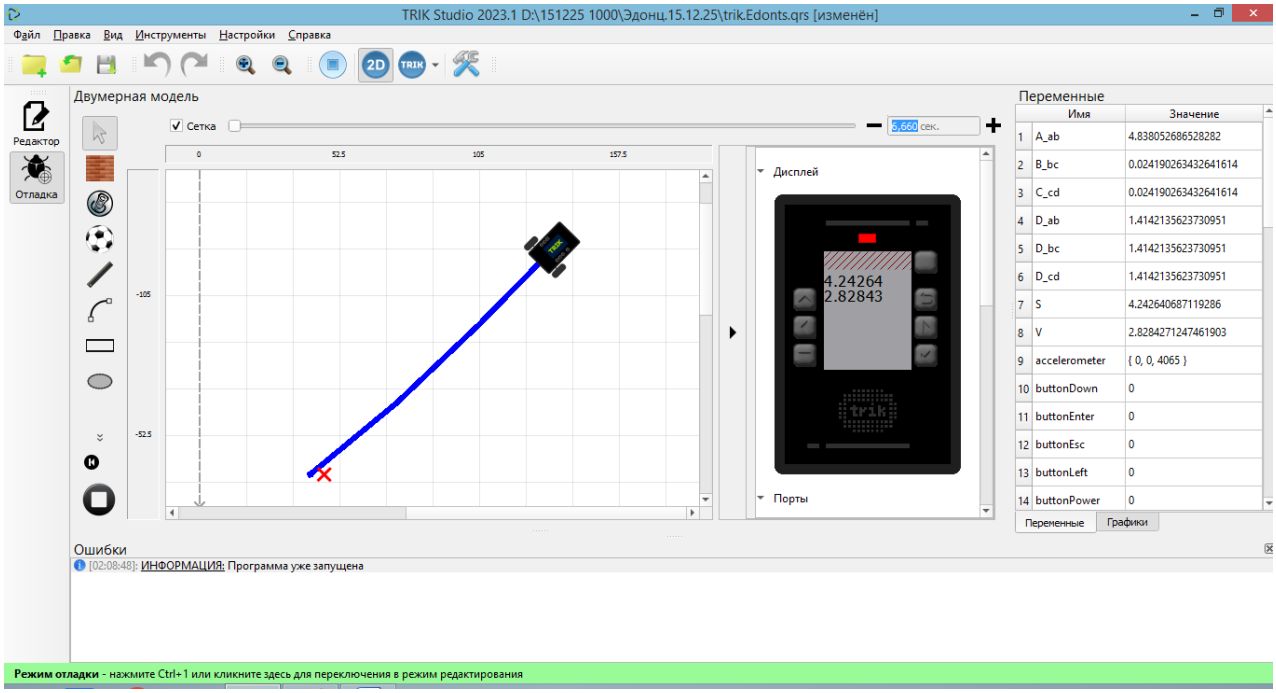
```
import sys
import time
import random
import math

class Program():
  __interpretation_started_timestamp__ = time.time() * 1000

  pi = 3.141592653589793
  A_ab = None
  B_bc = None
  C_cd = None
  D_ab = None
  D_bc = None
  D_cd = None
  S = None
  V = None
  en1 = None
  en2 = None
  en3 = None
  ena1 = None
  ena2 = None
  ena3 = None
  td1 = None
  td2 = None
  td3 = None

  def execMain(self):


    self.td1 = math.atan((2.0 - 1.0) / (2.0 - 1.0))
    self.td2 = math.atan((3.0 - 2.0) / (3.0 - 2.0))
```

```
self.td3 = math.atan((4.0 - 3.0) / (4.0 - 3.0))
self.D_ab = math.sqrt(math.pow((2.0 - 1.0), 2) + math.pow((2.0 - 1.0), 2))
self.D_bc = math.sqrt(math.pow((3.0 - 2.0), 2) + math.pow((3.0 - 2.0), 2))
self.D_cd = math.sqrt(math.pow((4.0 - 3.0), 2) + math.pow((4.0 - 3.0), 2))
self.S = self.D_ab + self.D_bc + self.D_cd
self.V = self.S / 1.5
self.A_ab = 15.4 / 2 * (0.45 * self.pi - self.td1)
self.B_bc = 15.4 / 2 * (self.td1 + 0.001 * self.pi - self.td2)
self.C_cd = 15.4 / 2 * (self.td2 + 0.001 * self.pi - self.td3)
self.ena1 = self.A_ab * 360 / self.pi / 5.6
self.ena2 = self.B_bc * 360 / self.pi / 5.6
self.ena3 = self.C_cd * 360 / self.pi / 5.6
self.en1 = self.D_ab * 360 / self.pi / 5.6
self.en2 = self.D_bc * 360 / self.pi / 5.6
self.en3 = self.D_cd * 360 / self.pi / 5.6
brick.marker().down("blue")

brick.encoder("E1").reset()
brick.encoder("E2").reset()
brick.encoder("E3").reset()
brick.encoder("E4").reset()

brick.motor("M3").setPower(100)

brick.motor("M4").setPower(-(100))

while not (brick.encoder("E3").read() > int(self.ena1)):
  script.wait(10)

brick.encoder("E1").reset()
brick.encoder("E2").reset()
brick.encoder("E3").reset()
brick.encoder("E4").reset()

brick.motor("M3").setPower(100)
brick.motor("M4").setPower(100)

while not (brick.encoder("E3").read() > int(self.en1 * 30)):
  script.wait(10)

brick.encoder("E1").reset()
brick.encoder("E2").reset()
brick.encoder("E3").reset()
brick.encoder("E4").reset()

brick.motor("M3").setPower(100)

brick.motor("M4").setPower(-(100))

while not (brick.encoder("E3").read() > int(self.ena2)):
  script.wait(10)

brick.encoder("E1").reset()
brick.encoder("E2").reset()
brick.encoder("E3").reset()
brick.encoder("E4").reset()

brick.motor("M3").setPower(100)
brick.motor("M4").setPower(100)

while not (brick.encoder("E3").read() > int(self.en2 * 30)):
  script.wait(10)

brick.encoder("E1").reset()
brick.encoder("E2").reset()
brick.encoder("E3").reset()
```

```
    brick.encoder("E4").reset()

    brick.motor("M3").setPower(100)

    brick.motor("M4").setPower(-(100))

    while not (brick.encoder("E3").read() > int(self.ena3)):
      script.wait(10)

    brick.encoder("E1").reset()
    brick.encoder("E2").reset()
    brick.encoder("E3").reset()
    brick.encoder("E4").reset()

    brick.motor("M3").setPower(100)
    brick.motor("M4").setPower(100)

    while not (brick.encoder("E3").read() > int(self.en3 * 25)):
      script.wait(10)

    brick.motor("M1").powerOff()
    brick.motor("M2").powerOff()
    brick.motor("M3").powerOff()
    brick.motor("M4").powerOff()

    brick.display().addLabel(str(self.S), 1, 1, 45)
    brick.display().redraw()

    brick.display().addLabel(str(self.V), 1, 45, 45)
    brick.display().redraw()

    script.wait(30000)

    brick.stop()
    return

def main():
  program = Program()
  program.execMain()

if __name__ == '__main__':
  main()
```

Задание № 4.1. Тестирование в среде ТРИК





Задание № 4.2. Тестирование на роботе

Робот двиггается по траектории из задания № 1. На дисплее вводится две строки: первая - длинна всего маршрута в метрах, вторая - скорость робота в м/с.

На диспле выводится:

4.23

2.82

Кейс 2. Вариант 2.

Условие

В среде программирования Trick Studio в программе на языке Python задаются данные для перемещения мобильного трехколесного робота (или блоками). Программа должна по полученным данным вычислить среднюю скорость перемещения робота за заданное время при равномерном перемещении робота по траектории из четырех точек. Необходимо написать программу, в которую включить заданные точки маршрута для того, чтобы программа рассчитала и вывела значения длинны маршрута и средней скорости робота. Задание реализуется либо в Trick Studio, либо на реальном роботе.

Формат ввода

При написании программы необходимо учесть следующие заданные координаты точек маршрута:

x0,y0 – координаты нулевой точки (A) маршрута, м.;

x1,y1 – координаты первой точки (B) маршрута, м.;

x2,y2 – координаты второй точки (C) маршрута, м.;

x3,y3 – координаты третьей точки (D) маршрута, м.;

Робот движется равномерно по траектории. Параметры координат точек маршрута и времени его прохождения задаются в основном коде программы в виде не целых чисел. В каждой строке с координатами два числа, первое x, второе y. Числа в строке разделены одинарным пробелом. В пятой строке указывается одно число – время. Размерность координат – метры, времени – секунды и миллисекунды (ограничение в два знака после точки). Границы координат не выходят за пределы от 0 до 5 метров по x и y.

Пример задания входных данных (x0,y0), (x1,y1), (x2,y2), (x3,y3), t для программы:

2.39 0.29

0.88 2.49

4.37 3.25

2.81 2.09

47.0
Участники самостоятельно разрабатывают программный код, содержащий разделы считывания заданных координат, промежуточных расчетов и вывода данных на дисплей робота вычисленных расстояния между точками и скорости в м/с. Необходимо использовать следующие формулы для расчета параметров скорости и расстояния между всеми точками маршрута:
$V = \frac{L}{t}$, где V - скорость (м/с), L - длинна пути маршрута (м), t - время (сек),
|AB|= $\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$, где √ - корень квадратный, |AB| - модуль расстояния между соседними точками A и B с координатами (x1,y1), (x2,y2) соответственно. Остальные расстояния и формулы для участков BC, CD и DE задаются самостоятельно.
Формулы необходимо самостоятельно преобразовать к программному виду и включить в программу. Обозначения параметров функций и переменных участники могут выбирать самостоятельно.

Формат вывода

Необходимо чтобы робот проехал по заданным точкам в среде симуляции (или в реальности). Необходимо приложить к ответу видеозапись симуляции перемещения робота из Trick Studio. Код предоставлять в трех форматах: 1) скриншотами в отчете; 2) блоками в Trick Studio сохранен проект с работающей симуляцией; 3) в среде Trick Studio, но на языке Python.
Программа должна вывести две строки. В первой строке должно вывестись вычисленная программой длинна всего маршрута в виде вещественного числа с двумя знаками после запятой, единица измерения метры. Во второй строке должно вывестись вычисленная программой скорость робота в виде вещественного числа с двумя знаками после запятой, единица измерения метры в секунду (м/с). Необходимо предусмотреть только вывод чисел. В качестве разделителя целой и вещественной части – точка. Текст комментариев и единицы измерения выводить не нужно.
Пример того, что должна вывести программа (числа не связанны с данными, указанными в разделе "Формат ввода"):
9.44
3.03

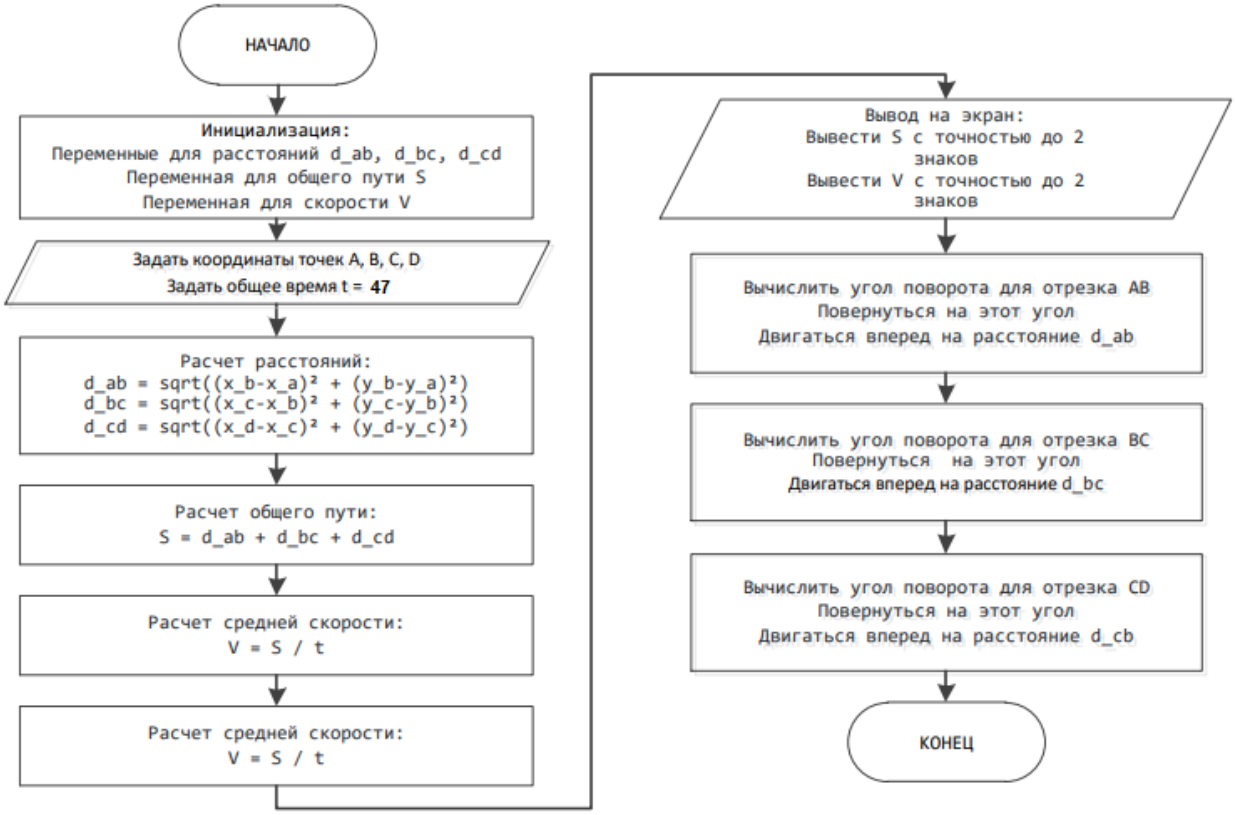
Исходные точки траектории движения.
Задание входных данных (x0,y0), (x1,y1), (x2,y2), (x3,y3), t для программы:
2.39 0.29
0.88 2.49
4.37 3.25
2.81 2.09
47.0

Задание № 1. Вычисление параметров движения

Расчтеные данные движения:
Участки пути.
td1=atan((2.49-0.29)/(2.39-0.88))
td2=atan((3.25-2.49)/(4.73-0.88))
td3=atan((3.25-2.09)/(4.37-2.81))
D_ab=sqrt((0.88-2.39)^2+(2.49-0.29)^2)
D_bc=sqrt((4.37-0.88)^2+(3.25-2.49)^2)
D_cd=sqrt((2.81-4.37)^2+(2.09-3.25)^2)
S=D_ab+D_bc+D_cd
V=S/47
A_ab=15.4/2*(0.98*pi-td1)
B_bc=15.4/2*(td1+1.15*pi-td2)
C_cd=15.4/2*(td2+1.3*pi-td3)
ena1=A_ab*360/pi/5.6

ena2=B_bc*360/pi/5.6
ena3=C_cd*360/pi/5.6
en1=D_ab*360/pi/5.6
en2=D_bc*360/pi/5.6
en3=D_cd*360/pi/5.6

Задание № 2. Блок-схема алгоритма



Задание № 3.1. Управляющая программа (в базовом варианте – управление движением по времени)

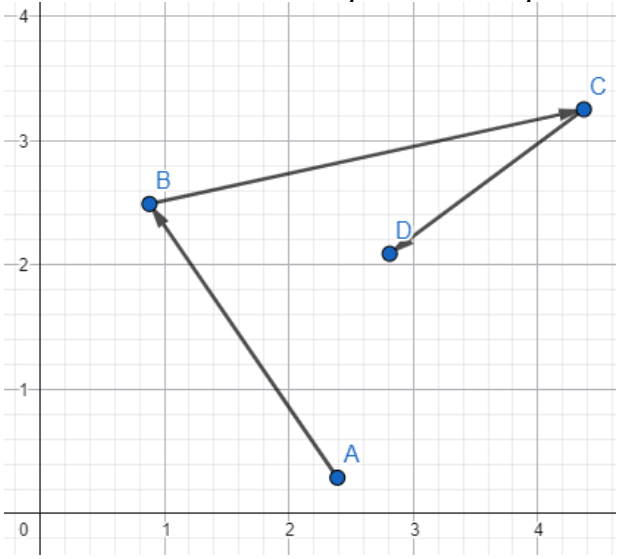
Задание № 3.2. Управляющая программа (в усложненном варианте – управление движением по показаниям энкодеров)

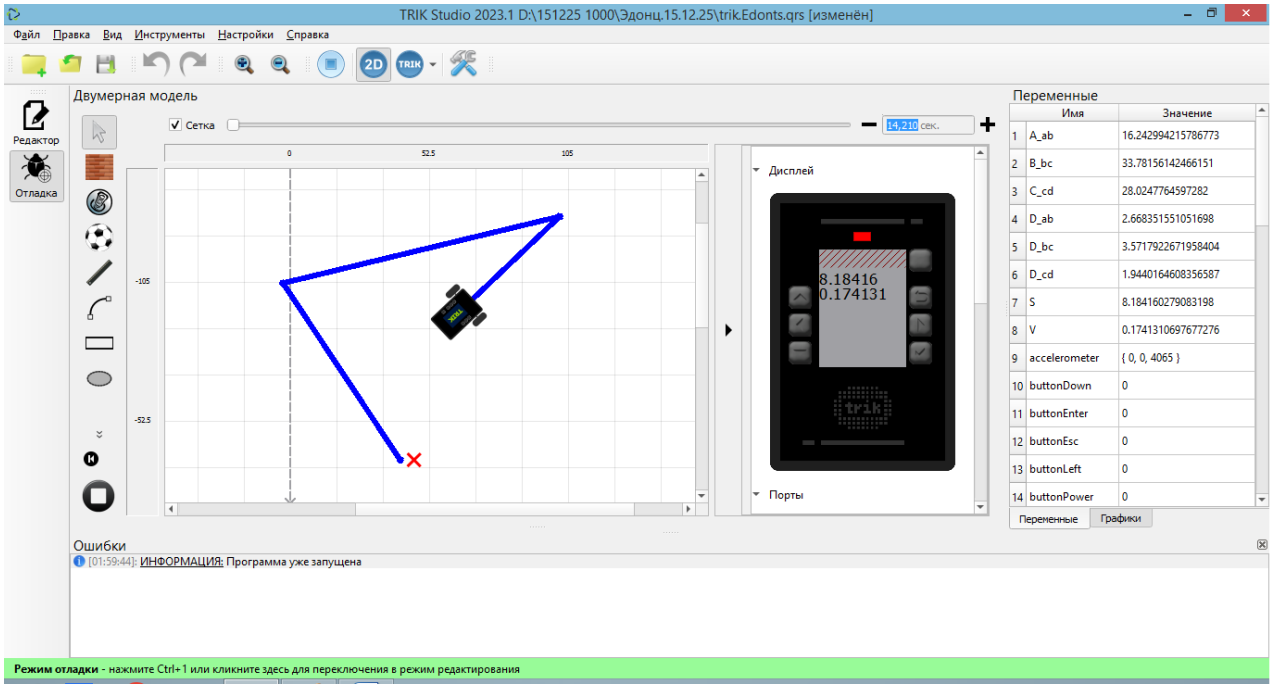
```
import sys
import time
import random
import math

class Program():
  __interpretation_started_timestamp__ = time.time() * 1000

  pi = 3.141592653589793
  A_ab = None
  B_bc = None
  C_cd = None
  D_ab = None
  D_bc = None
  D_cd = None
  S = None
  V = None
  en1 = None
  en2 = None
  en3 = None
  ena1 = None
  ena2 = None
  ena3 = None
  td1 = None
  td2 = None
  td3 = None
```

```
def execMain(self):


  self.td1 = math.atan((2.49 - 0.29) / (2.39 - 0.88))
  self.td2 = math.atan((3.25 - 2.49) / (4.73 - 0.88))
  self.td3 = math.atan((3.25 - 2.09) / (4.37 - 2.81))
  self.D_ab = math.sqrt(math.pow((0.88 - 2.39), 2) + math.pow((2.49 - 0.29), 2))
  self.D_bc = math.sqrt(math.pow((4.37 - 0.88), 2) + math.pow((3.25 - 2.49), 2))
  self.D_cd = math.sqrt(math.pow((2.81 - 4.37), 2) + math.pow((2.09 - 3.25), 2))
  self.S = self.D_ab + self.D_bc + self.D_cd
  self.V = self.S / 47
  self.A_ab = 15.4 / 2 * (0.98 * self.pi - self.td1)
  self.B_bc = 15.4 / 2 * (self.td1 + 1.15 * self.pi - self.td2)
  self.C_cd = 15.4 / 2 * (self.td2 + 1.3 * self.pi - self.td3)
  self.ena1 = self.A_ab * 360 / self.pi / 5.6
  self.ena2 = self.B_bc * 360 / self.pi / 5.6
  self.ena3 = self.C_cd * 360 / self.pi / 5.6
  self.en1 = self.D_ab * 360 / self.pi / 5.6
  self.en2 = self.D_bc * 360 / self.pi / 5.6
  self.en3 = self.D_cd * 360 / self.pi / 5.6
  brick.marker().down("blue")

  brick.encoder("E1").reset()
  brick.encoder("E2").reset()
  brick.encoder("E3").reset()
  brick.encoder("E4").reset()

  brick.motor("M3").setPower(100)

  brick.motor("M4").setPower(-(100))

  while not (brick.encoder("E3").read() > int(self.ena1)):
    script.wait(10)

  brick.encoder("E1").reset()
  brick.encoder("E2").reset()
  brick.encoder("E3").reset()
  brick.encoder("E4").reset()

  brick.motor("M3").setPower(100)
  brick.motor("M4").setPower(100)

  while not (brick.encoder("E3").read() > int(self.en1 * 30)):
    script.wait(10)

  brick.encoder("E1").reset()
  brick.encoder("E2").reset()
  brick.encoder("E3").reset()
  brick.encoder("E4").reset()

  brick.motor("M3").setPower(100)

  brick.motor("M4").setPower(-(100))

  while not (brick.encoder("E3").read() > int(self.ena2)):
    script.wait(10)

  brick.encoder("E1").reset()
  brick.encoder("E2").reset()
  brick.encoder("E3").reset()
  brick.encoder("E4").reset()

  brick.motor("M3").setPower(100)
  brick.motor("M4").setPower(100)

  while not (brick.encoder("E3").read() > int(self.en2 * 30)):
```

```
      script.wait(10)

    brick.encoder("E1").reset()
    brick.encoder("E2").reset()
    brick.encoder("E3").reset()
    brick.encoder("E4").reset()

    brick.motor("M3").setPower(100)

    brick.motor("M4").setPower(-(100))

    while not (brick.encoder("E3").read() > int(self.ena3)):
      script.wait(10)

    brick.encoder("E1").reset()
    brick.encoder("E2").reset()
    brick.encoder("E3").reset()
    brick.encoder("E4").reset()

    brick.motor("M3").setPower(100)
    brick.motor("M4").setPower(100)

    while not (brick.encoder("E3").read() > int(self.en3 * 25)):
      script.wait(10)

    brick.motor("M1").powerOff()
    brick.motor("M2").powerOff()
    brick.motor("M3").powerOff()
    brick.motor("M4").powerOff()

    brick.display().addLabel(str(self.S), 1, 1, 45)
    brick.display().redraw()

    brick.display().addLabel(str(self.V), 1, 45, 45)
    brick.display().redraw()

    script.wait(30000)

    brick.stop()
    return

def main():
  program = Program()
  program.execMain()

if __name__ == '__main__':
  main()
```

Задание № 4.1. Тестирование в среде ТРИК





Задание № 4.2. Тестирование на роботе

Робот двиггается по траектории из задания № 1. На дисплее вводится две строки: первая - длинна всего маршрута в метрах, вторая - скорость робота в м/с.

На диспле выводится:

8.18

0.17

Кейс 2. Вариант 3.

Условие

В среде программирования Trick Studio в программе на языке Python задаются данные для перемещения мобильного трехколесного робота (или блоками). Программа должна по полученным данным вычислить среднюю скорость перемещения робота за заданное время при равномерном перемещении робота по траектории из четырех точек. Необходимо написать программу, в которую включить заданные точки маршрута для того, чтобы программа рассчитала и вывела значения длинны маршрута и средней скорости робота. Задание реализуется либо в Trick Studio, либо на реальном роботе.

Формат ввода

При написании программы необходимо учесть следующие заданные координаты точек маршрута:

x0,y0 – координаты нулевой точки (A) маршрута, м.;

x1,y1 – координаты первой точки (B) маршрута, м.;

x2,y2 – координаты второй точки (C) маршрута, м.;

x3,y3 – координаты третьей точки (D) маршрута, м.;

Робот движется равномерно по траектории. Параметры координат точек маршрута и времени его прохождения задаются в основном коде программы в виде не целых чисел. В каждой строке с координатами два числа, первое x, второе y. Числа в строке разделены одинарным пробелом. В пятой строке указывается одно число – время. Размерность координат – метры, времени – секунды и миллисекунды (ограничение в два знака после точки). Границы координат не выходят за пределы от 0 до 5 метров по x и y.

Пример задания входных данных (x0,y0), (x1,y1), (x2,y2), (x3,y3), t для программы:

3.27 3.07

0.7 2.52

4.3 4.97

3.73 1.56

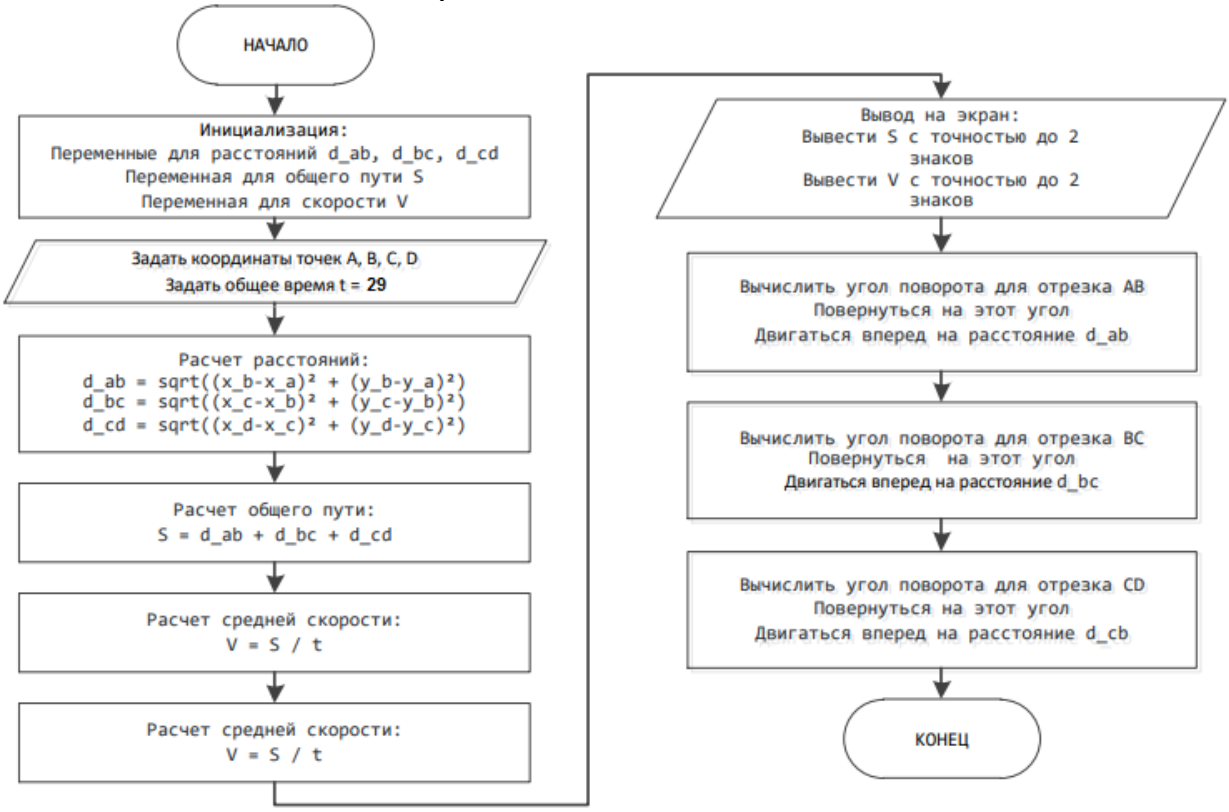
29.0
Участники самостоятельно разрабатывают программный код, содержащий разделы считывания заданных координат, промежуточных расчетов и вывода данных на дисплей робота вычисленных расстояния между точками и скорости в м/с. Необходимо использовать следующие формулы для расчета параметров скорости и расстояния между всеми точками маршрута:
$V = \frac{L}{t}$, где V - скорость (м/с), L - длинна пути маршрута (м), t - время (сек),
|AB|= $\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$, где √ - корень квадратный, |AB| - модуль расстояния между соседними точками A и B с координатами (x1,y1), (x2,y2) соответственно. Остальные расстояния и формулы для участков BC, CD и DE задаются самостоятельно.
Формулы необходимо самостоятельно преобразовать к программному виду и включить в программу. Обозначения параметров функций и переменных участники могут выбирать самостоятельно.
Формат вывода
Необходимо чтобы робот проехал по заданным точкам в среде симуляции (или в реальности). Необходимо приложить к ответу видеозапись симуляции перемещения робота из Trick Studio. Код предоставлять в трех форматах: 1) скриншотами в отчете; 2) блоками в Trick Studio сохранен проект с работающей симуляцией; 3) в среде Trick Studio, но на языке Python.
Программа должна вывести две строки. В первой строке должно вывестись вычисленная программой длинна всего маршрута в виде вещественного числа с двумя знаками после запятой, единица измерения метры. Во второй строке должно вывестись вычисленная программой скорость робота в виде вещественного числа с двумя знаками после запятой, единица измерения метры в секунду (м/с). Необходимо предусмотреть только вывод чисел. В качестве разделителя целой и вещественной части – точка. Текст комментариев и единицы измерения выводить не нужно.
Пример того, что должна вывести программа (числа не связанны с данными, указанными в разделе "Формат ввода"):
9.44
3.03

Исходные точки траектории движения.
Задание входных данных (x0,y0), (x1,y1), (x2,y2), (x3,y3), t для программы:
3.27 3.07
0.7 2.52
4.3 4.97
3.73 1.56
29.0

Задание № 1. Вычисление параметров движения
Расчтеные данные движения:
Участки пути.
td1=atan((3.07-2.52)/(3.27-0.7))
td2=atan((4.97-2.52)/(4.3-0.7))
td3=atan((4.97-1.56)/(4.3-3.73))
D_ab=sqrt((0.7-3.27)^2+(2.52-3.07)^2)
D_bc=sqrt((4.3-0.7)^2+(4.97-2.52)^2)
D_cd=sqrt((3.73-4.3)^2+(1.56-4.97)^2)
S=D_ab+D_bc+D_cd
V=S/29
A_ab=15.4/2*(1.15*pi-td1)
B_bc=15.4/2*(td1+1.25*pi-td2)
C_cd=15.4/2*(td2+1.48*pi-td3)
ena1=A_ab*360/pi/5.6

ena2=B_bc*360/pi/5.6
ena3=C_cd*360/pi/5.6
en1=D_ab*360/pi/5.6
en2=D_bc*360/pi/5.6
en3=D_cd*360/pi/5.6

Задание № 2. Блок-схема алгоритма



Задание № 3.1. Управляющая программа (в базовом варианте – управление движением по времени)

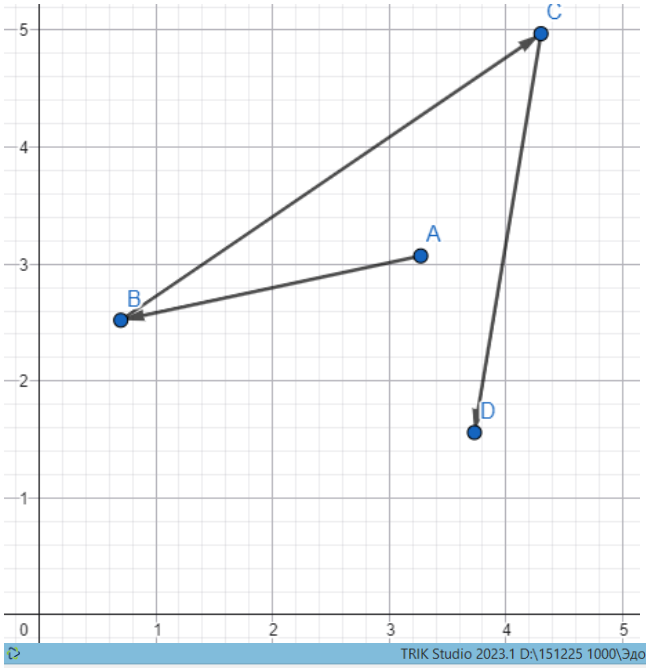
Задание № 3.2. Управляющая программа (в усложненном варианте – управление движением по показаниям энкодеров)

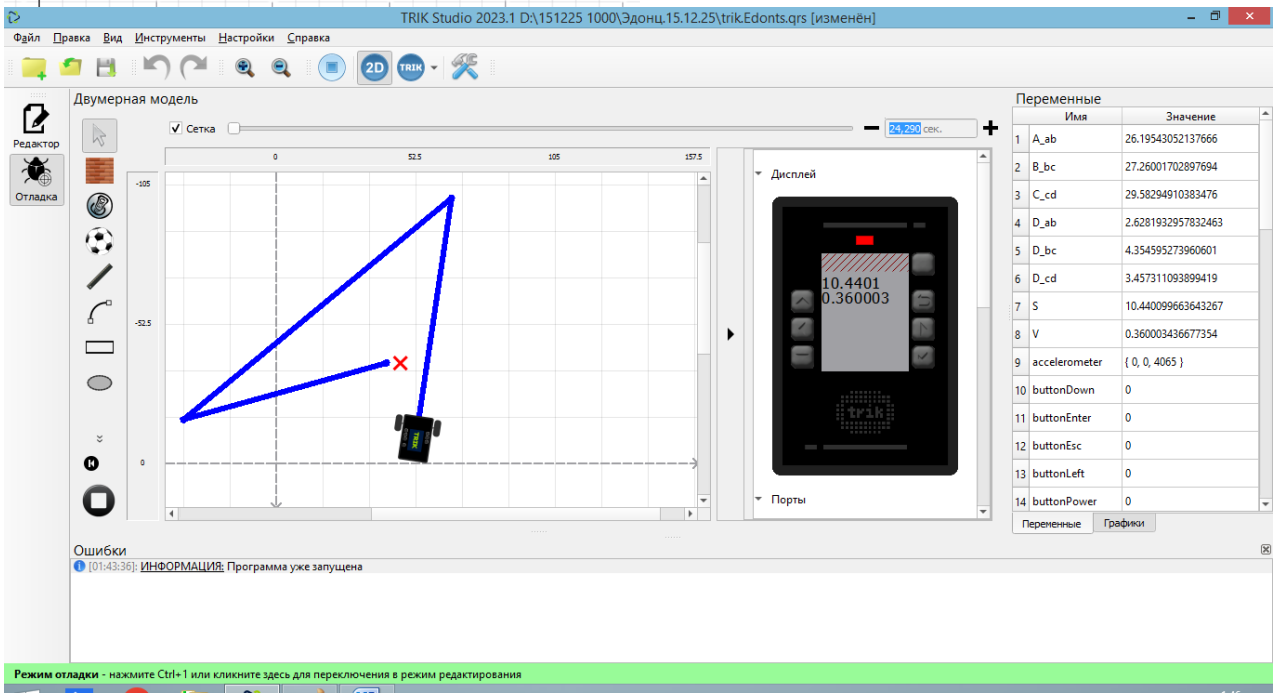
```
import sys
import time
import random
import math

class Program():
  __interpretation_started_timestamp__ = time.time() * 1000

  pi = 3.141592653589793
  A_ab = None
  B_bc = None
  C_cd = None
  D_ab = None
  D_bc = None
  D_cd = None
  S = None
  V = None
  en1 = None
  en2 = None
  en3 = None
  ena1 = None
  ena2 = None
  ena3 = None
  td1 = None
  td2 = None
```

```
td3 = None

def execMain(self):


  self.td1 = math.atan((3.07 - 2.52) / (3.27 - 0.7))
  self.td2 = math.atan((4.97 - 2.52) / (4.3 - 0.7))
  self.td3 = math.atan((4.97 - 1.56) / (4.3 - 3.73))
  self.D_ab = math.sqrt(math.pow((0.7 - 3.27), 2) + math.pow((2.52 - 3.07), 2))
  self.D_bc = math.sqrt(math.pow((4.3 - 0.7), 2) + math.pow((4.97 - 2.52), 2))
  self.D_cd = math.sqrt(math.pow((3.73 - 4.3), 2) + math.pow((1.56 - 4.97), 2))
  self.S = self.D_ab + self.D_bc + self.D_cd
  self.V = self.S / 29
  self.A_ab = 15.4 / 2 * (1.15 * self.pi - self.td1)
  self.B_bc = 15.4 / 2 * (self.td1 + 1.25 * self.pi - self.td2)
  self.C_cd = 15.4 / 2 * (self.td2 + 1.48 * self.pi - self.td3)
  self.ena1 = self.A_ab * 360 / self.pi / 5.6
  self.ena2 = self.B_bc * 360 / self.pi / 5.6
  self.ena3 = self.C_cd * 360 / self.pi / 5.6
  self.en1 = self.D_ab * 360 / self.pi / 5.6
  self.en2 = self.D_bc * 360 / self.pi / 5.6
  self.en3 = self.D_cd * 360 / self.pi / 5.6
  brick.marker().down("blue")

  brick.encoder("E1").reset()
  brick.encoder("E2").reset()
  brick.encoder("E3").reset()
  brick.encoder("E4").reset()

  brick.motor("M3").setPower(100)

  brick.motor("M4").setPower(-(100))

  while not (brick.encoder("E3").read() > int(self.ena1)):
    script.wait(10)

  brick.encoder("E1").reset()
  brick.encoder("E2").reset()
  brick.encoder("E3").reset()
  brick.encoder("E4").reset()

  brick.motor("M3").setPower(100)
  brick.motor("M4").setPower(100)

  while not (brick.encoder("E3").read() > int(self.en1 * 30)):
    script.wait(10)

  brick.encoder("E1").reset()
  brick.encoder("E2").reset()
  brick.encoder("E3").reset()
  brick.encoder("E4").reset()

  brick.motor("M3").setPower(100)

  brick.motor("M4").setPower(-(100))

  while not (brick.encoder("E3").read() > int(self.ena2)):
    script.wait(10)

  brick.encoder("E1").reset()
  brick.encoder("E2").reset()
  brick.encoder("E3").reset()
  brick.encoder("E4").reset()

  brick.motor("M3").setPower(100)
  brick.motor("M4").setPower(100)
```

```
    while not (brick.encoder("E3").read() > int(self.en2 * 30)):
      script.wait(10)

    brick.encoder("E1").reset()
    brick.encoder("E2").reset()
    brick.encoder("E3").reset()
    brick.encoder("E4").reset()

    brick.motor("M3").setPower(100)

    brick.motor("M4").setPower(-(100))

    while not (brick.encoder("E3").read() > int(self.ena3)):
      script.wait(10)

    brick.encoder("E1").reset()
    brick.encoder("E2").reset()
    brick.encoder("E3").reset()
    brick.encoder("E4").reset()

    brick.motor("M3").setPower(100)
    brick.motor("M4").setPower(100)

    while not (brick.encoder("E3").read() > int(self.en3 * 25)):
      script.wait(10)

    brick.motor("M1").powerOff()
    brick.motor("M2").powerOff()
    brick.motor("M3").powerOff()
    brick.motor("M4").powerOff()

    brick.display().addLabel(str(self.S), 1, 1, 45)
    brick.display().redraw()

    brick.display().addLabel(str(self.V), 1, 45, 45)
    brick.display().redraw()

    script.wait(30000)

    brick.stop()
    return

def main():
  program = Program()
  program.execMain()

if __name__ == '__main__':
  main()
```

Задание № 4.1. Тестирование в среде ТРИК





Задание № 4.2. Тестирование на роботе

Робот двиггается по траектории из задания № 1. На дисплее вводится две строки: первая - длинна всего маршрута в метрах, вторая - скорость робота в м/с.

На диспле выводится:

10.44

0.36

Кейс 2. Вариант 4.

Условие

В среде программирования Trick Studio в программе на языке Python задаются данные для перемещения мобильного трехколесного робота (или блоками). Программа должна по полученным данным вычислить среднюю скорость перемещения робота за заданное время при равномерном перемещении робота по траектории из четырех точек. Необходимо написать программу, в которую включить заданные точки маршрута для того, чтобы программа рассчитала и вывела значения длинны маршрута и средней скорости робота. Задание реализуется либо в Trick Studio, либо на реальном роботе.

Формат ввода

При написании программы необходимо учесть следующие заданные координаты точек маршрута:
x0,y0 – координаты нулевой точки (A) маршрута, м.;
x1,y1 – координаты первой точки (B) маршрута, м.;
x2,y2 – координаты второй точки (C) маршрута, м.;
x3,y3 – координаты третьей точки (D) маршрута, м.;
Робот движется равномерно по траектории. Параметры координат точек маршрута и времени его прохождения задаются в основном коде программы в виде не целых чисел. В каждой строке с координатами два числа, первое x, второе y. Числа в строке разделены одинарным пробелом. В пятой строке указывается одно число – время. Размерность координат – метры, времени – секунды и миллисекунды (ограничение в два знака после точки). Границы координат не выходят за пределы от 0 до 5 метров по x и y.
Пример задания входных данных (x0,y0), (x1,y1), (x2,y2), (x3,y3), t для программы:
0.78 0.66
0.64 4.24
2.25 0.37
3.48 2.65
42.0
Участники самостоятельно разрабатывают программный код, содержащий разделы считывания заданных координат, промежуточных расчетов и вывода данных на дисплей робота вычисленных расстояния между точками и скорости в м/с. Необходимо использовать следующие формулы для расчета параметров скорости и расстояния между всеми точками маршрута:
$V = \frac{L}{t}$, где V - скорость (м/с), L - длинна пути маршрута (м), t - время (сек),
|AB|= $\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$, где √ - корень квадратный, |AB| - модуль расстояния между соседними точками A и B с координатами (x1,y1), (x2,y2) соответственно. Остальные расстояния и формулы для участков BC, CD и DE задаются самостоятельно.
Формулы необходимо самостоятельно преобразовать к программному виду и включить в программу. Обозначения параметров функций и переменных участники могут выбирать самостоятельно.

Формат вывода

Необходимо чтобы робот проехал по заданным точкам в среде симуляции (или в реальности). Необходимо приложить к ответу видеозапись симуляции перемещения робота из Trick Studio. Код предоставлять в трех форматах: 1) скриншотами в отчете; 2) блоками в Trick Studio сохранен проект с работающей симуляцией; 3) в среде Trick Studio, но на языке Python.
Программа должна вывести две строки. В первой строке должно вывестись вычисленная программой длинна всего маршрута в виде вещественного числа с двумя знаками после запятой, единица измерения метры. Во второй строке должно вывестись вычисленная программой скорость робота в виде вещественного числа с двумя знаками после запятой, единица измерения метры в секунду (м/с). Необходимо предусмотреть только вывод чисел. В качестве разделителя целой и вещественной части – точка. Текст комментариев и единицы измерения выводить не нужно.
Пример того, что должна вывести программа (числа не связанны с данными, указанными в разделе "Формат ввода"):
9.44
3.03

Исходные точки траектории движения.
Задание входных данных (x0,y0), (x1,y1), (x2,y2), (x3,y3), t для программы:
0.78 0.66
0.64 4.24
2.25 0.37
3.48 2.65
42.0

Задание № 1. Вычисление параметров движения

Расчтеные данные движения:

Участки пути.

td1=atan((4.24-0.66)/(0.78-0.64))

td2=atan((4.24-0.37)/(2.25-0.64))

td3=atan((2.65-0.37)/(3.48-2.25))

D_ab=sqrt((0.64-0.78)^2+(4.24-0.66)^2)

D_bc=sqrt((2.25-0.64)^2+(0.37-4.24)^2)

D_cd=sqrt((3.48-2.25)^2+(2.65-0.37)^2)

S=D_ab+D_bc+D_cd

V=S/42

A_ab=15.4/2*(0.99*pi-td1)

B_bc=15.4/2*(td1+pi-td2)

C_cd=15.4/2*(td2+0.65*pi-td3)

ena1=A_ab*360/pi/5.6

ena2=B_bc*360/pi/5.6

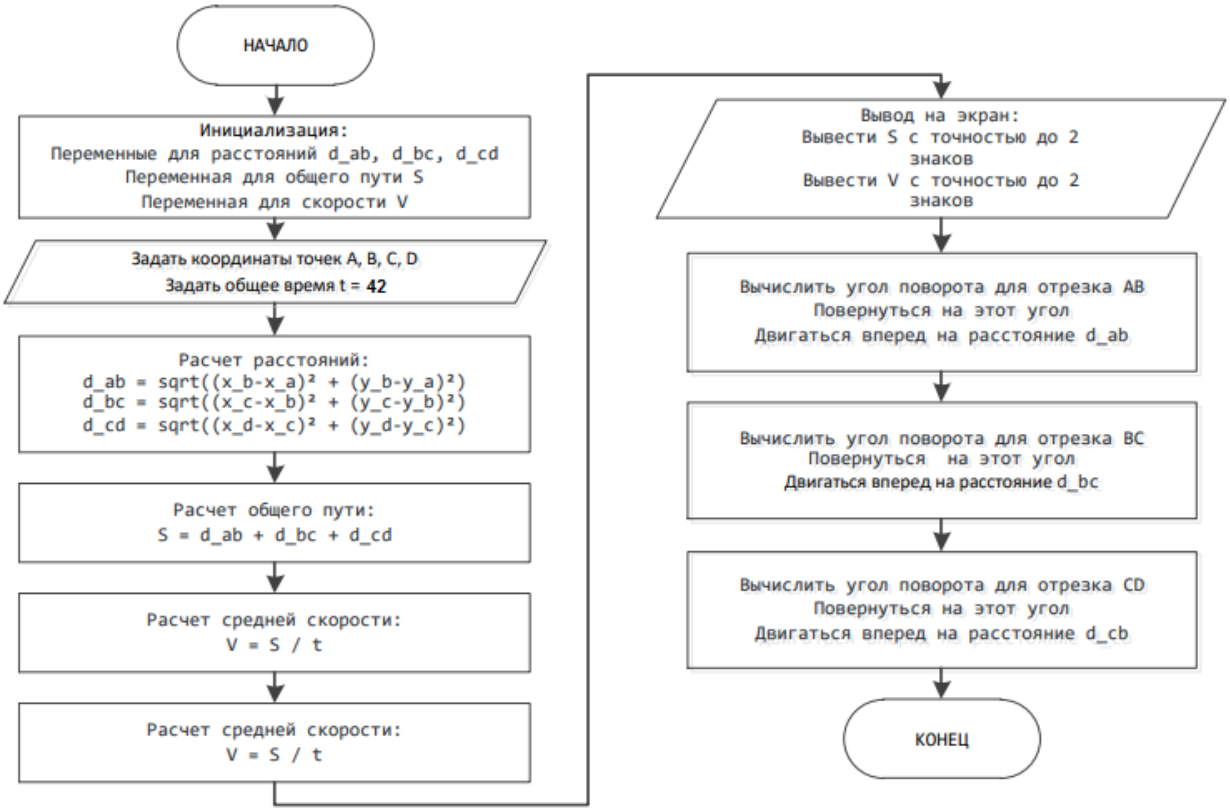
ena3=C_cd*360/pi/5.6

en1=D_ab*360/pi/5.6

en2=D_bc*360/pi/5.6

en3=D_cd*360/pi/5.6

Задание № 2. Блок-схема алгоритма



Задание № 3.1. Управляющая программа (в базовом варианте – управление движением по времени)

Задание № 3.2. Управляющая программа (в усложненном варианте – управление движением по показаниям энкодеров)

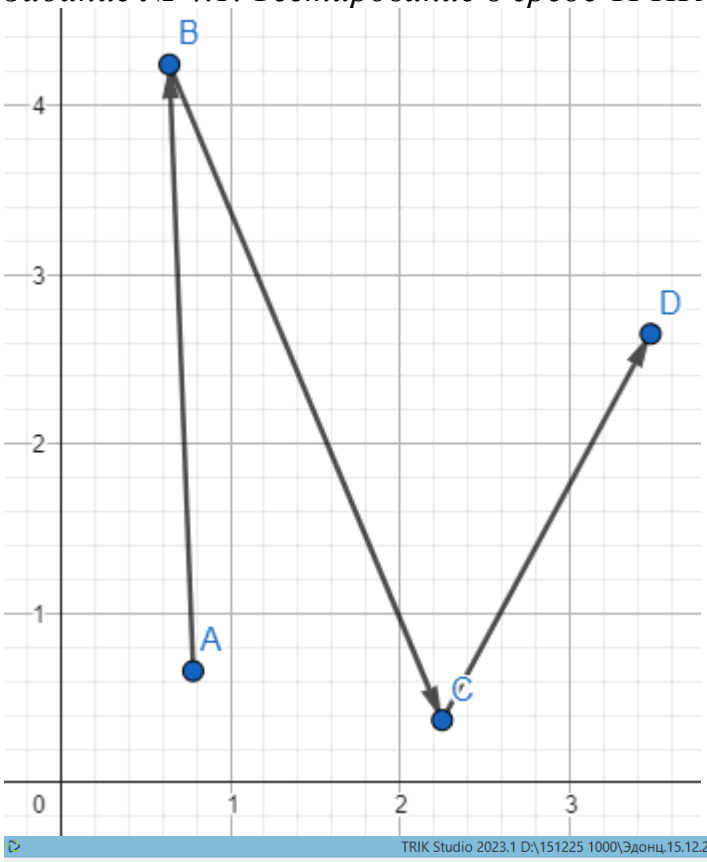
```
import sys
import time
import random
import math

class Program():
  __interpretation_started_timestamp__ = time.time() * 1000
```

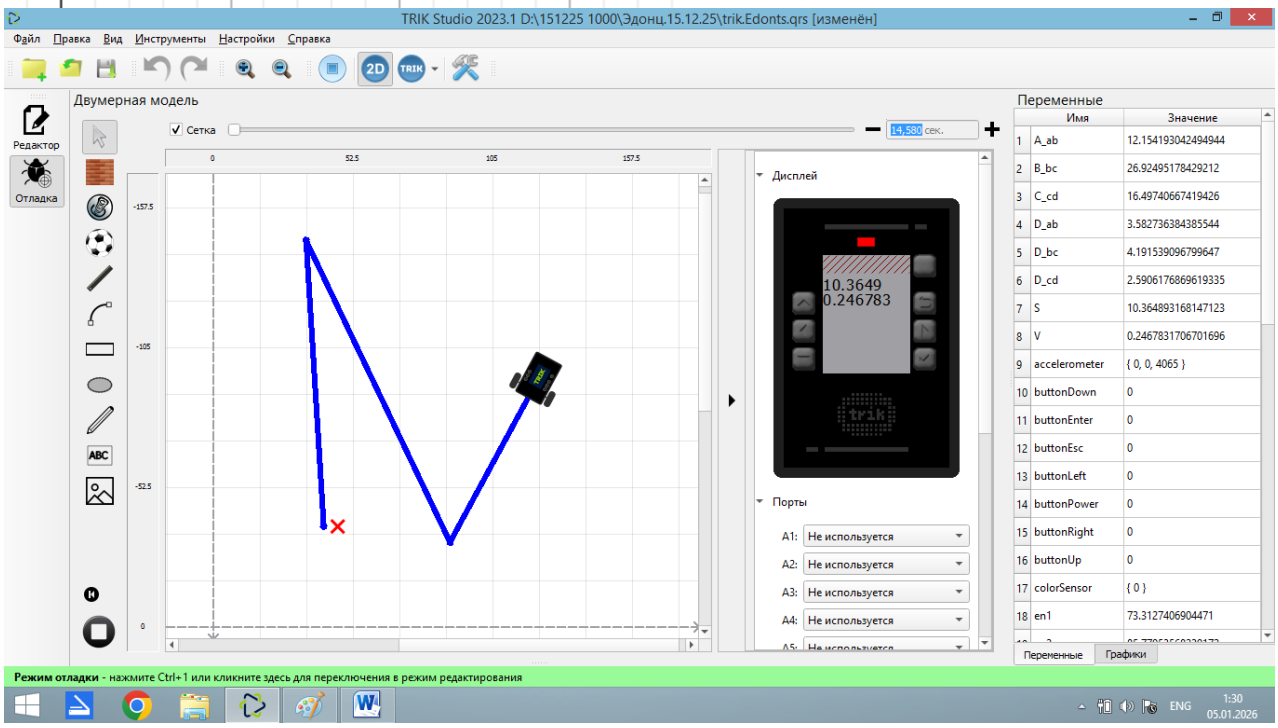
```
pi = 3.141592653589793
A_ab = None
B_bc = None
C_cd = None
D_ab = None
D_bc = None
D_cd = None
S = None
V = None
en1 = None
en2 = None
en3 = None
ena1 = None
ena2 = None
ena3 = None
td1 = None
td2 = None
td3 = None

def execMain(self):


  self.td1 = math.atan((4.24 - 0.66) / (0.78 - 0.64))
  self.td2 = math.atan((4.24 - 0.37) / (2.25 - 0.64))
  self.td3 = math.atan((2.65 - 0.37) / (3.48 - 2.25))
  self.D_ab = math.sqrt(math.pow((0.64 - 0.78), 2) + math.pow((4.24 - 0.66), 2))
  self.D_bc = math.sqrt(math.pow((2.25 - 0.64), 2) + math.pow((0.37 - 4.24), 2))
  self.D_cd = math.sqrt(math.pow((3.48 - 2.25), 2) + math.pow((2.65 - 0.37), 2))
  self.S = self.D_ab + self.D_bc + self.D_cd
  self.V = self.S / 42
  self.A_ab = 15.4 / 2 * (0.99 * self.pi - self.td1)
  self.B_bc = 15.4 / 2 * (self.td1 + self.pi - self.td2)
  self.C_cd = 15.4 / 2 * (self.td2 + 0.65 * self.pi - self.td3)
  self.ena1 = self.A_ab * 360 / self.pi / 5.6
  self.ena2 = self.B_bc * 360 / self.pi / 5.6
  self.ena3 = self.C_cd * 360 / self.pi / 5.6
  self.en1 = self.D_ab * 360 / self.pi / 5.6
  self.en2 = self.D_bc * 360 / self.pi / 5.6
  self.en3 = self.D_cd * 360 / self.pi / 5.6
  brick.marker().down("blue")

  brick.encoder("E1").reset()
  brick.encoder("E2").reset()
  brick.encoder("E3").reset()
  brick.encoder("E4").reset()

  brick.motor("M3").setPower(100)

  brick.motor("M4").setPower(-(100))

  while not (brick.encoder("E3").read() > int(self.ena1)):
    script.wait(10)

  brick.encoder("E1").reset()
  brick.encoder("E2").reset()
  brick.encoder("E3").reset()
  brick.encoder("E4").reset()

  brick.motor("M3").setPower(100)
  brick.motor("M4").setPower(100)

  while not (brick.encoder("E3").read() > int(self.en1 * 30)):
    script.wait(10)

  brick.encoder("E1").reset()
```

```
    brick.encoder("E2").reset()
    brick.encoder("E3").reset()
    brick.encoder("E4").reset()

    brick.motor("M3").setPower(100)

    brick.motor("M4").setPower(-(100))

    while not (brick.encoder("E3").read() > int(self.ena2)):
      script.wait(10)

    brick.encoder("E1").reset()
    brick.encoder("E2").reset()
    brick.encoder("E3").reset()
    brick.encoder("E4").reset()

    brick.motor("M3").setPower(100)
    brick.motor("M4").setPower(100)

    while not (brick.encoder("E3").read() > int(self.en2 * 30)):
      script.wait(10)

    brick.encoder("E1").reset()
    brick.encoder("E2").reset()
    brick.encoder("E3").reset()
    brick.encoder("E4").reset()

    brick.motor("M3").setPower(100)

    brick.motor("M4").setPower(-(100))

    while not (brick.encoder("E3").read() > int(self.ena3)):
      script.wait(10)

    brick.encoder("E1").reset()
    brick.encoder("E2").reset()
    brick.encoder("E3").reset()
    brick.encoder("E4").reset()

    brick.motor("M3").setPower(100)
    brick.motor("M4").setPower(100)

    while not (brick.encoder("E3").read() > int(self.en3 * 25)):
      script.wait(10)

    brick.motor("M1").powerOff()
    brick.motor("M2").powerOff()
    brick.motor("M3").powerOff()
    brick.motor("M4").powerOff()

    brick.display().addLabel(str(self.S), 1, 1, 45)
    brick.display().redraw()

    brick.display().addLabel(str(self.V), 1, 45, 45)
    brick.display().redraw()

    script.wait(30000)

    brick.stop()
    return

def main():
  program = Program()
  program.execMain()

if __name__ == '__main__':
```

main()

Задание № 4.1. Тестирование в среде ТРИК





Задание № 4.2. Тестирование на роботе

Робот двиггается по траектории из п. 1. На дисплее вводится две строки: первая - длинна всего маршрута в метрах, вторая - скорость робота в м/с. На диспле выводится:

10.36

0.25

Кейс 2. Вариант 5.

Условие

В среде программирования Trick Studio в программе на языке Python задаются данные для перемещения мобильного трехколесного робота (или блоками). Программа должна по полученным данным вычислить среднюю скорость перемещения робота за заданное время при равномерном перемещении робота по траектории из четырех точек. Необходимо написать программу, в которую включить заданные точки маршрута для того, чтобы программа рассчитала и вывела значения длинны маршрута и средней скорости робота. Задание реализуется либо в Trick Studio, либо на реальном роботе.

Формат ввода

При написании программы необходимо учесть следующие заданные координаты точек маршрута:

x0,y0 – координаты нулевой точки (A) маршрута, м.;
x1,y1 – координаты первой точки (B) маршрута, м.;
x2,y2 – координаты второй точки (C) маршрута, м.;
x3,y3 – координаты третьей точки (D) маршрута, м.;

Робот движется равномерно по траектории. Параметры координат точек маршрута и времени его прохождения задаются в основном коде программы в виде не целых чисел. В каждой строке с координатами два числа, первое x, второе y. Числа в строке разделены одинарным пробелом. В пятой строке указывается одно число – время. Размерность координат – метры, времени – секунды и миллисекунды (ограничение в два знака после точки). Границы координат не выходят за пределы от 0 до 5 метров по x и y.

Пример задания входных данных (x0,y0), (x1,y1), (x2,y2), (x3,y3), t для программы:

4.42 2.03
1.1 0.99
3.31 0.19
0.59 2.06
16.0

Участники самостоятельно разрабатывают программный код, содержащий разделы считывания заданных координат, промежуточных расчетов и вывода данных на дисплей робота вычисленных расстояния между точками и скорости в м/с. Необходимо использовать следующие формулы для расчета параметров скорости и расстояния между всеми точками маршрута:

$V = \frac{L}{t}$, где V - скорость (м/с), L - длинна пути маршрута (м), t - время (сек),

|AB|= $\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$, где √ - корень квадратный, |AB| - модуль расстояния между соседними точками A и B с координатами (x1,y1), (x2,y2) соответственно. Остальные расстояния и формулы для участков BC, CD и DE задаются самостоятельно.

Формулы необходимо самостоятельно преобразовать к программному виду и включить в программу. Обозначения параметров функций и переменных участники могут выбирать самостоятельно.

Формат вывода

Необходимо чтобы робот проехал по заданным точкам в среде симуляции (или в реальности). Необходимо приложить к ответу видеозапись симуляции перемещения робота из Trick Studio. Код предоставлять в трех форматах: 1) скриншотами в отчете; 2) блоками в Trick Studio сохранен проект с работающей симуляцией; 3) в среде Trick Studio, но на языке Python.

Программа должна вывести две строки. В первой строке должно вывестись вычисленная программой длинна всего маршрута в виде вещественного числа с двумя знаками после запятой, единица измерения метры. Во второй строке должно вывестись вычисленная программой скорость робота в виде вещественного числа с двумя знаками после запятой, единица измерения метры в секунду (м/с). Необходимо предусмотреть только вывод чисел. В качестве разделителя целой и вещественной части – точка. Текст комментариев и единицы измерения выводить не нужно.

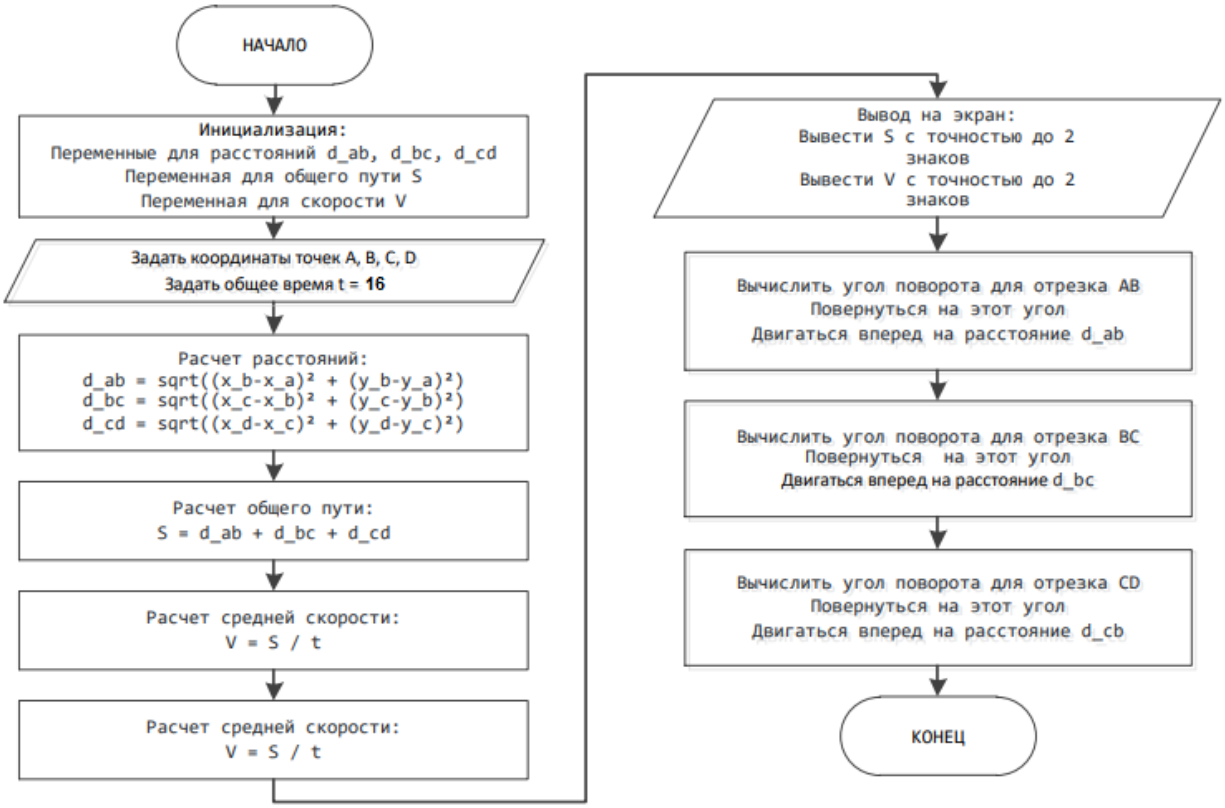
Пример того, что должна вывести программа (числа не связанны с данными, указанными в разделе "Формат ввода"):

9.44
3.03

Исходные точки траектории движения.
Задание входных данных (x0,y0), (x1,y1), (x2,y2), (x3,y3), t для программы:
4.42 2.03
1.1 0.99
3.31 0.19
0.59 2.06
16.0

Задание № 1. Вычисление параметров движения
Расчтеные данные движения:
Участки пути.
td1=atan((2.03-0.99)/(4.42-1.1))
td2=atan((0.99-0.19)/(3.31-1.1))
td3=atan((2.06-0.19)/(3.31-0.59))
D_ab=sqrt((1.1-4.42)^2+(0.99-2.03)^2)
D_bc=sqrt((3.31-1.1)^2+(0.19-0.99)^2)
D_cd=sqrt((0.59-3.31)^2+(2.06-0.19)^2)
S=D_ab+D_bc+D_cd
V=S/16
A_ab=15.4/2*(1.2*pi-td1)
B_bc=15.4/2*(td1+0.8*pi-td2)
C_cd=15.4/2*(td2+pi-td3)
ena1=A_ab*360/pi/5.6
ena2=B_bc*360/pi/5.6
ena3=C_cd*360/pi/5.6
en1=D_ab*360/pi/5.6
en2=D_bc*360/pi/5.6
en3=D_cd*360/pi/5.6

Задание № 2. Блок-схема алгоритма



Задание № 3.1. Управляющая программа (в базовом варианте – управление движением по времени)

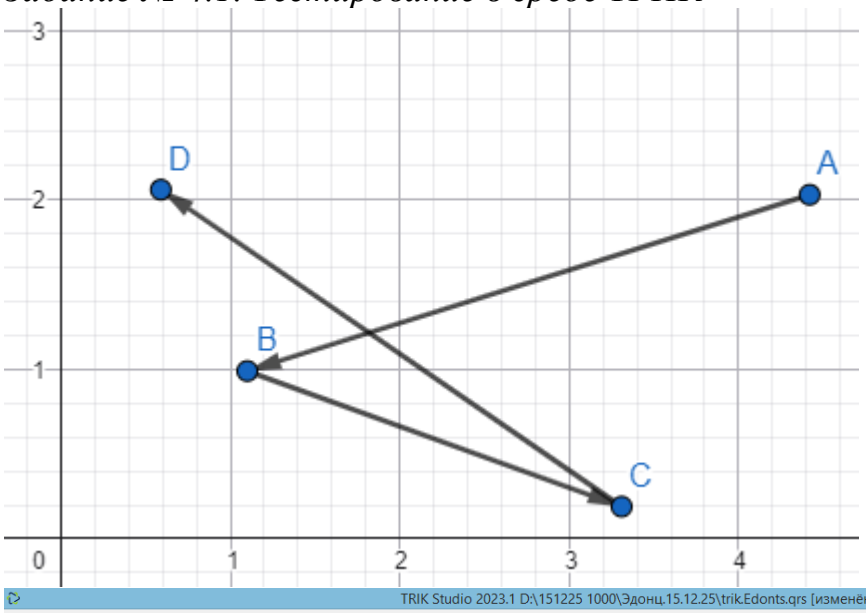
Задание № 3.2. Управляющая программа (в усложненном варианте – управление движением по показаниям энкодеров)

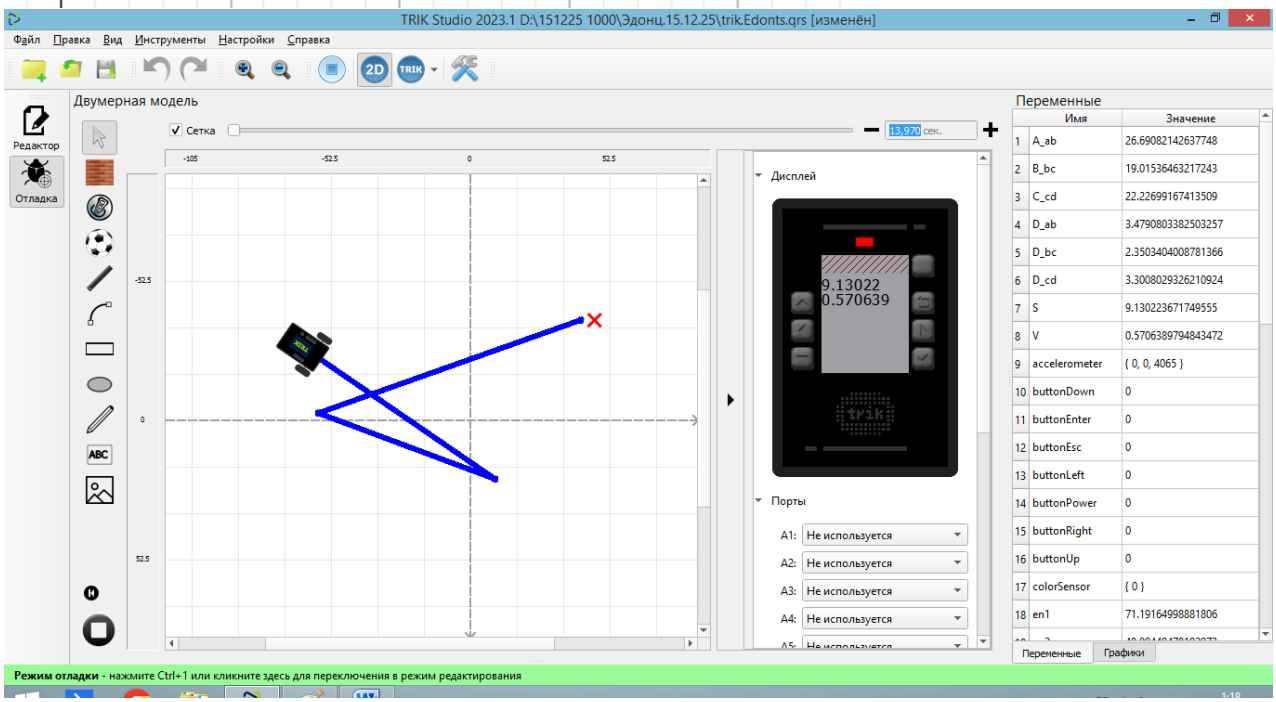
```
import sys
import time
import random
import math

class Program():
  __interpretation_started_timestamp__ = time.time() * 1000

  pi = 3.141592653589793
  A_ab = None
  B_bc = None
  C_cd = None
  D_ab = None
  D_bc = None
  D_cd = None
  S = None
  V = None
  en1 = None
  en2 = None
  en3 = None
  ena1 = None
  ena2 = None
  ena3 = None
  td1 = None
  td2 = None
  td3 = None

  def execMain(self):


    self.td1 = math.atan((2.03 - 0.99) / (4.42 - 1.1))
    self.td2 = math.atan((0.99 - 0.19) / (3.31 - 1.1))
    self.td3 = math.atan((2.06 - 0.19) / (3.31 - 0.59))
    self.D_ab = math.sqrt(math.pow((1.1 - 4.42), 2) + math.pow((0.99 - 2.03), 2))
    self.D_bc = math.sqrt(math.pow((3.31 - 1.1), 2) + math.pow((0.19 - 0.99), 2))
    self.D_cd = math.sqrt(math.pow((0.59 - 3.31), 2) + math.pow((2.06 - 0.19), 2))
    self.S = self.D_ab + self.D_bc + self.D_cd
    self.V = self.S / 16
    self.A_ab = 15.4 / 2 * (1.2 * self.pi - self.td1)
    self.B_bc = 15.4 / 2 * (self.td1 + 0.8 * self.pi - self.td2)
    self.C_cd = 15.4 / 2 * (self.td2 + self.pi - self.td3)
    self.ena1 = self.A_ab * 360 / self.pi / 5.6
    self.ena2 = self.B_bc * 360 / self.pi / 5.6
    self.ena3 = self.C_cd * 360 / self.pi / 5.6
    self.en1 = self.D_ab * 360 / self.pi / 5.6
    self.en2 = self.D_bc * 360 / self.pi / 5.6
    self.en3 = self.D_cd * 360 / self.pi / 5.6
    brick.marker().down("blue")

    brick.encoder("E1").reset()
    brick.encoder("E2").reset()
    brick.encoder("E3").reset()
    brick.encoder("E4").reset()

    brick.motor("M3").setPower(100)

    brick.motor("M4").setPower(-(100))

    while not (brick.encoder("E3").read() > int(self.ena1)):
      script.wait(10)

    brick.encoder("E1").reset()
    brick.encoder("E2").reset()
    brick.encoder("E3").reset()
```

```
brick.encoder("E4").reset()

brick.motor("M3").setPower(100)
brick.motor("M4").setPower(100)

while not (brick.encoder("E3").read() > int(self.en1 * 30)):
  script.wait(10)

brick.encoder("E1").reset()
brick.encoder("E2").reset()
brick.encoder("E3").reset()
brick.encoder("E4").reset()

brick.motor("M3").setPower(100)

brick.motor("M4").setPower(-(100))

while not (brick.encoder("E3").read() > int(self.ena2)):
  script.wait(10)

brick.encoder("E1").reset()
brick.encoder("E2").reset()
brick.encoder("E3").reset()
brick.encoder("E4").reset()

brick.motor("M3").setPower(100)
brick.motor("M4").setPower(100)

while not (brick.encoder("E3").read() > int(self.en2 * 30)):
  script.wait(10)

brick.encoder("E1").reset()
brick.encoder("E2").reset()
brick.encoder("E3").reset()
brick.encoder("E4").reset()

brick.motor("M3").setPower(100)

brick.motor("M4").setPower(-(100))

while not (brick.encoder("E3").read() > int(self.ena3)):
  script.wait(10)

brick.encoder("E1").reset()
brick.encoder("E2").reset()
brick.encoder("E3").reset()
brick.encoder("E4").reset()

brick.motor("M3").setPower(100)
brick.motor("M4").setPower(100)

while not (brick.encoder("E3").read() > int(self.en3 * 25)):
  script.wait(10)

brick.motor("M1").powerOff()
brick.motor("M2").powerOff()
brick.motor("M3").powerOff()
brick.motor("M4").powerOff()

brick.display().addLabel(str(self.S), 1, 1, 45)
brick.display().redraw()

brick.display().addLabel(str(self.V), 1, 45, 45)
brick.display().redraw()

script.wait(30000)
```

```
    brick.stop()
    return

def main():
  program = Program()
  program.execMain()

if __name__ == '__main__':
  main()
```

Задание № 4.1. Тестирование в среде ТРИК





Задание № 4.2. Тестирование на роботе

Робот двиггается по траектории из п. 1. На дисплее вводится две строки: первая - длинна всего маршрута в метрах, вторая - скорость робота в м/с. На диспле выводится:

9.13

0.57

Кейс 2. Вариант 6.

Условие

В среде программирования Trick Studio в программе на языке Python задаются данные для перемещения мобильного трехколесного робота (или блоками). Программа должна по полученным данным вычислить среднюю скорость перемещения робота за заданное время при равномерном перемещении робота по траектории из четырех точек. Необходимо написать программу, в которую включить заданные точки маршрута для того, чтобы программа рассчитала и вывела значения длинны маршрута и средней скорости робота. Задание реализуется либо в Trick Studio, либо на реальном роботе.

Формат ввода

При написании программы необходимо учесть следующие заданные координаты точек маршрута:

x0,y0 – координаты нулевой точки (A) маршрута, м.;
x1,y1 – координаты первой точки (B) маршрута, м.;
x2,y2 – координаты второй точки (C) маршрута, м.;
x3,y3 – координаты третьей точки (D) маршрута, м.;

Робот движется равномерно по траектории. Параметры координат точек маршрута и времени его прохождения задаются в основном коде программы в виде не целых чисел. В каждой строке с координатами два числа, первое x, второе y. Числа в строке разделены одинарным пробелом. В пятой строке указывается одно число – время. Размерность координат – метры, времени – секунды и миллисекунды (ограничение в два знака после точки). Границы координат не выходят за пределы от 0 до 5 метров по x и y.

Пример задания входных данных (x0,y0), (x1,y1), (x2,y2), (x3,y3), t для программы:

1.95 2.16
0.81 2.57
3.34 1.6
3.63 3.48
17.0

Участники самостоятельно разрабатывают программный код, содержащий разделы считывания заданных координат, промежуточных расчетов и вывода данных на дисплей робота вычисленных расстояния между точками и скорости в м/с. Необходимо использовать следующие формулы для расчета параметров скорости и расстояния между всеми точками маршрута:

$V = \frac{L}{t}$, где V - скорость (м/с), L - длинна пути маршрута (м), t - время (сек),

|AB|= $\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$, где √ - корень квадратный, |AB| - модуль расстояния между соседними точками A и B с координатами (x1,y1), (x2,y2) соответственно. Остальные расстояния и формулы для участков BC, CD и DE задаются самостоятельно.

Формулы необходимо самостоятельно преобразовать к программному виду и включить в программу. Обозначения параметров функций и переменных участники могут выбирать самостоятельно.

Формат вывода

Необходимо чтобы робот проехал по заданным точкам в среде симуляции (или в реальности). Необходимо приложить к ответу видеозапись симуляции перемещения робота из Trick Studio. Код предоставлять в трех форматах: 1) скриншотами в отчете; 2) блоками в Trick Studio сохранен проект с работающей симуляцией; 3) в среде Trick Studio, но на языке Python.

Программа должна вывести две строки. В первой строке должно вывестись вычисленная программой длинна всего маршрута в виде вещественного числа с двумя знаками после запятой, единица измерения метры. Во второй строке должно вывестись вычисленная программой скорость робота в виде вещественного числа с двумя знаками после запятой, единица измерения метры в секунду (м/с). Необходимо предусмотреть только вывод чисел. В качестве разделителя целой и вещественной части – точка. Текст комментариев и единицы измерения выводить не нужно.

Пример того, что должна вывести программа (числа не связанны с данными, указанными в разделе "Формат ввода"):

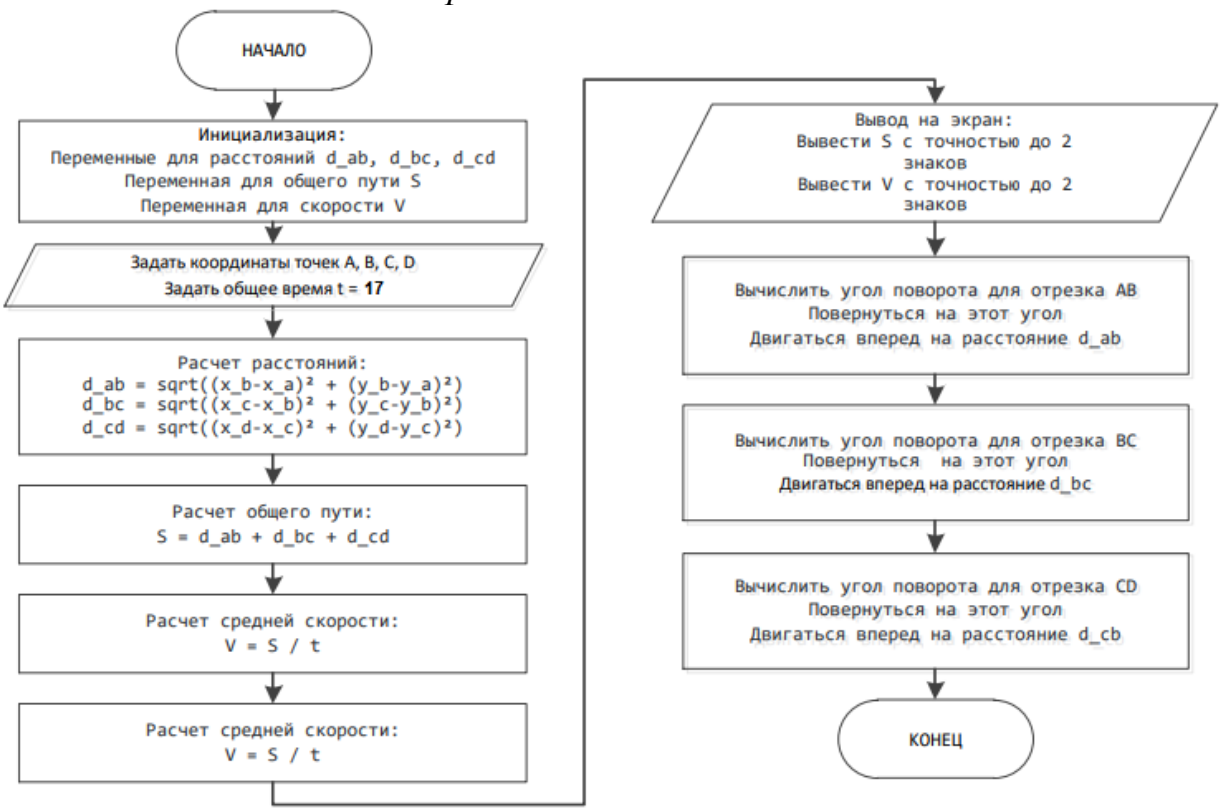
9.44
3.03

Исходные точки траектории движения.
Задание входных данных (x0,y0), (x1,y1), (x2,y2), (x3,y3), t для программы:
1.95 2.16
0.81 2.57
3.34 1.6
3.63 3.48
17.0

Задание № 1. Вычисление параметров движения

Расчтеные данные движения:
Участки пути.
td1=atan((2.57-2.16)/(1.95-0.81))
td2=atan((2.57-1.6)/(3.34-0.81))
td3=atan((3.48-1.6)/(3.63-3.34))
D_ab=sqrt((0.81-1.95)^2+(2.57-2.16)^2)
D_bc=sqrt((3.34-0.81)^2+(1.6-2.57)^2)
D_cd=sqrt((3.63-3.34)^2+(3.48-1.6)^2)
S=D_ab+D_bc+D_cd
V=S/17
A_ab=15.4/2*(pi-td1)
B_bc=15.4/2*(td1+pi-td2)
C_cd=15.4/2*(td2+0.86*pi-td3)
ena1=A_ab*360/pi/5.6
ena2=B_bc*360/pi/5.6
ena3=C_cd*360/pi/5.6
en1=D_ab*360/pi/5.6
en2=D_bc*360/pi/5.6
en3=D_cd*360/pi/5.6

Задание № 2. Блок-схема алгоритма



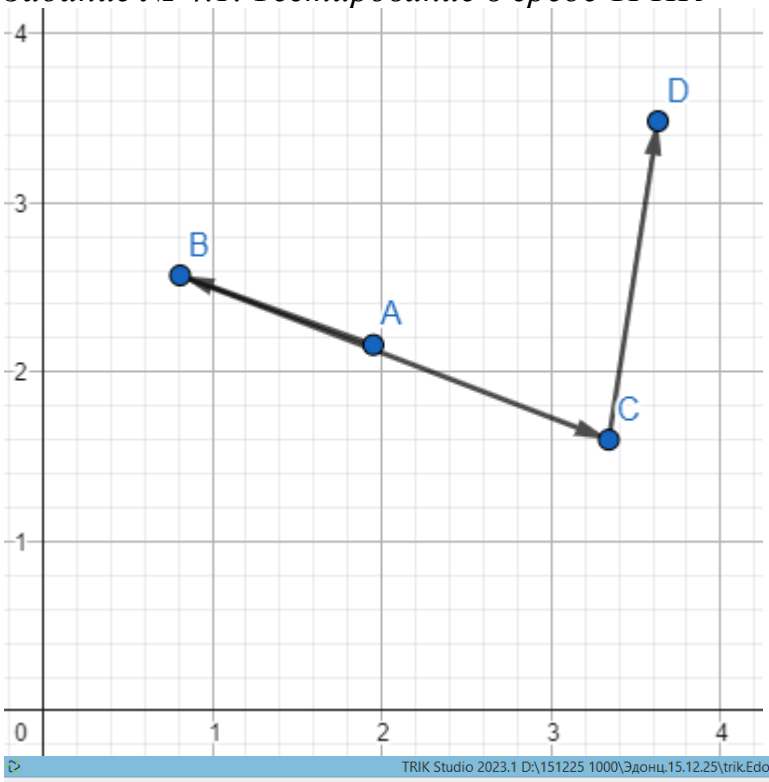
Задание № 3.1. Управляющая программа (в базовом варианте – управление движением по времени)
Задание № 3.2. Управляющая программа (в усложненном варианте – управление движением по показаниям энкодеров)

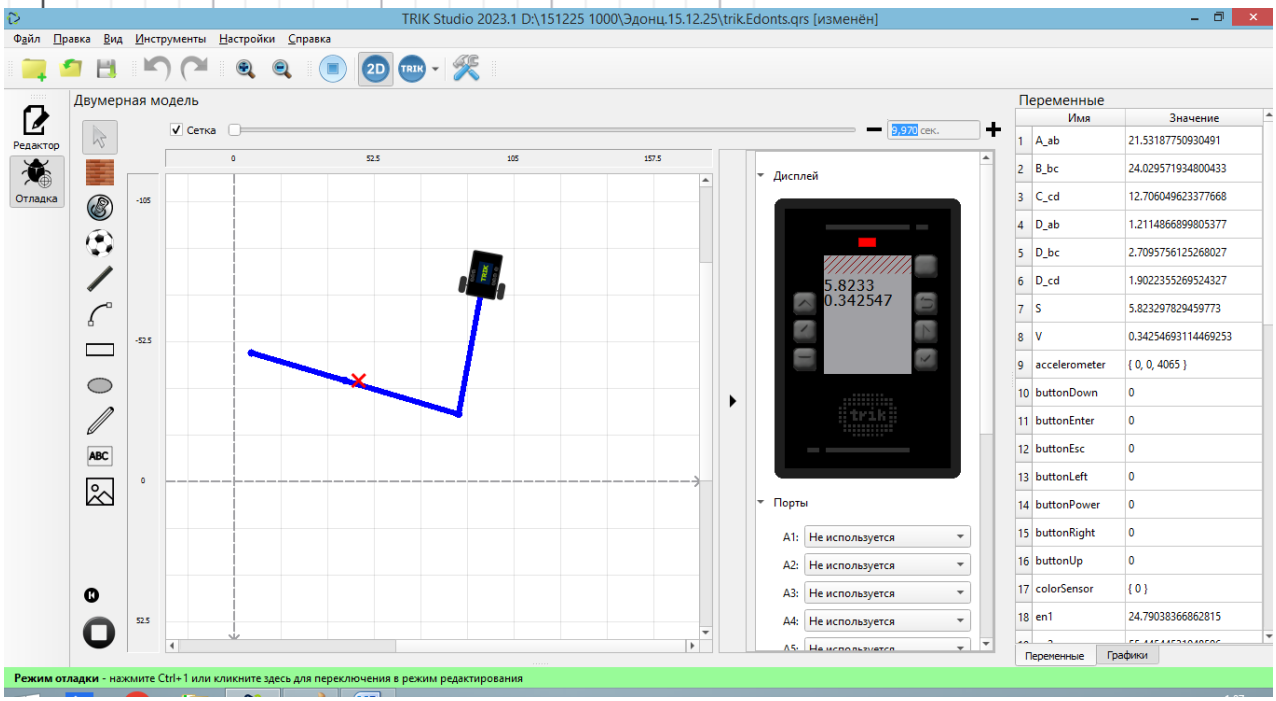
```
import sys
import time
import random
import math

class Program():
  __interpretation_started_timestamp__ = time.time() * 1000

  pi = 3.141592653589793
  A_ab = None
  B_bc = None
  C_cd = None
  D_ab = None
  D_bc = None
  D_cd = None
  S = None
  V = None
  en1 = None
  en2 = None
  en3 = None
  ena1 = None
  ena2 = None
  ena3 = None
  td1 = None
  td2 = None
  td3 = None

  def execMain(self):


    self.td1 = math.atan((2.57 - 2.16) / (1.95 - 0.81))
    self.td2 = math.atan((2.57 - 1.6) / (3.34 - 0.81))
    self.td3 = math.atan((3.48 - 1.6) / (3.63 - 3.34))
    self.D_ab = math.sqrt(math.pow((0.81 - 1.95), 2) + math.pow((2.57 - 2.16), 2))
    self.D_bc = math.sqrt(math.pow((3.34 - 0.81), 2) + math.pow((1.6 - 2.57), 2))
    self.D_cd = math.sqrt(math.pow((3.63 - 3.34), 2) + math.pow((3.48 - 1.6), 2))
    self.S = self.D_ab + self.D_bc + self.D_cd
    self.V = self.S / 17
    self.A_ab = 15.4 / 2 * (self.pi - self.td1)
    self.B_bc = 15.4 / 2 * (self.td1 + self.pi - self.td2)
    self.C_cd = 15.4 / 2 * (self.td2 + 0.86 * self.pi - self.td3)
    self.ena1 = self.A_ab * 360 / self.pi / 5.6
    self.ena2 = self.B_bc * 360 / self.pi / 5.6
    self.ena3 = self.C_cd * 360 / self.pi / 5.6
    self.en1 = self.D_ab * 360 / self.pi / 5.6
    self.en2 = self.D_bc * 360 / self.pi / 5.6
    self.en3 = self.D_cd * 360 / self.pi / 5.6
    brick.marker().down("blue")

    brick.encoder("E1").reset()
    brick.encoder("E2").reset()
    brick.encoder("E3").reset()
    brick.encoder("E4").reset()

    brick.motor("M3").setPower(100)

    brick.motor("M4").setPower(-(100))

    while not (brick.encoder("E3").read() > int(self.ena1)):
      script.wait(10)
```

```
brick.encoder("E1").reset()
brick.encoder("E2").reset()
brick.encoder("E3").reset()
brick.encoder("E4").reset()

brick.motor("M3").setPower(100)
brick.motor("M4").setPower(100)

while not (brick.encoder("E3").read() > int(self.en1 * 30)):
  script.wait(10)

brick.encoder("E1").reset()
brick.encoder("E2").reset()
brick.encoder("E3").reset()
brick.encoder("E4").reset()

brick.motor("M3").setPower(100)

brick.motor("M4").setPower(-(100))

while not (brick.encoder("E3").read() > int(self.ena2)):
  script.wait(10)

brick.encoder("E1").reset()
brick.encoder("E2").reset()
brick.encoder("E3").reset()
brick.encoder("E4").reset()

brick.motor("M3").setPower(100)
brick.motor("M4").setPower(100)

while not (brick.encoder("E3").read() > int(self.en2 * 30)):
  script.wait(10)

brick.encoder("E1").reset()
brick.encoder("E2").reset()
brick.encoder("E3").reset()
brick.encoder("E4").reset()

brick.motor("M3").setPower(100)

brick.motor("M4").setPower(-(100))

while not (brick.encoder("E3").read() > int(self.ena3)):
  script.wait(10)

brick.encoder("E1").reset()
brick.encoder("E2").reset()
brick.encoder("E3").reset()
brick.encoder("E4").reset()

brick.motor("M3").setPower(100)
brick.motor("M4").setPower(100)

while not (brick.encoder("E3").read() > int(self.en3 * 25)):
  script.wait(10)

brick.motor("M1").powerOff()
brick.motor("M2").powerOff()
brick.motor("M3").powerOff()
brick.motor("M4").powerOff()

brick.display().addLabel(str(self.S), 1, 1, 45)
brick.display().redraw()

brick.display().addLabel(str(self.V), 1, 45, 45)
```

```
    brick.display().redraw()

    script.wait(30000)

    brick.stop()
    return

def main():
  program = Program()
  program.execMain()

if __name__ == '__main__':
  main()
```

Задание № 4.1. Тестирование в среде ТРИК





Задание № 4.2. Тестирование на роботе

Робот двиггается по траектории из п. 1. На дисплее вводится две строки: первая - длинна всего маршрута в метрах, вторая - скорость робота в м/с. На диспле выводится:

5.82
0.34

Кейс 2. Вариант 7.

Условие

В среде программирования Trick Studio в программе на языке Python задаются данные для перемещения мобильного трехколесного робота (или блоками). Программа должна по полученным данным вычислить среднюю скорость перемещения робота за заданное время при равномерном перемещении робота по траектории из четырех точек. Необходимо написать программу, в которую включить заданные точки маршрута для того, чтобы программа рассчитала и вывела значения длинны маршрута и средней скорости робота. Задание реализуется либо в Trick Studio, либо на реальном роботе.

Формат ввода

При написании программы необходимо учесть следующие заданные координаты точек маршрута:

x0,y0 – координаты нулевой точки (A) маршрута, м.;
x1,y1 – координаты первой точки (B) маршрута, м.;
x2,y2 – координаты второй точки (C) маршрута, м.;
x3,y3 – координаты третьей точки (D) маршрута, м.;

Робот движется равномерно по траектории. Параметры координат точек маршрута и времени его прохождения задаются в основном коде программы в виде не целых чисел. В каждой строке с координатами два числа, первое x, второе y. Числа в строке разделены одинарным пробелом. В пятой строке указывается одно число – время. Размерность координат – метры, времени – секунды и миллисекунды (ограничение в два знака после точки). Границы координат не выходят за пределы от 0 до 5 метров по x и y.

Пример задания входных данных (x0,y0), (x1,y1), (x2,y2), (x3,y3), t для программы:

4.49 2.59
0.41 2.6
4.78 1.37
3.99 1.11
28.0

Участники самостоятельно разрабатывают программный код, содержащий разделы считывания заданных координат, промежуточных расчетов и вывода данных на дисплей робота вычисленных расстояния между точками и скорости в м/с. Необходимо использовать следующие формулы для расчета параметров скорости и расстояния между всеми точками маршрута:

$V = \frac{L}{t}$, где V - скорость (м/с), L - длинна пути маршрута (м), t - время (сек),

|AB|= $\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$, где √ - корень квадратный, |AB| - модуль расстояния между соседними точками A и B с координатами (x1,y1), (x2,y2) соответственно. Остальные расстояния и формулы для участков BC, CD и DE задаются самостоятельно.

Формулы необходимо самостоятельно преобразовать к программному виду и включить в программу. Обозначения параметров функций и переменных участники могут выбирать самостоятельно.

Формат вывода

Необходимо чтобы робот проехал по заданным точкам в среде симуляции (или в реальности). Необходимо приложить к ответу видеозапись симуляции перемещения робота из Trick Studio. Код предоставлять в трех форматах: 1) скриншотами в отчете; 2) блоками в Trick Studio сохранен проект с работающей симуляцией; 3) в среде Trick Studio, но на языке Python.

Программа должна вывести две строки. В первой строке должно вывестись вычисленная программой длинна всего маршрута в виде вещественного числа с двумя знаками после запятой, единица измерения метры. Во второй строке должно вывестись вычисленная программой скорость робота в виде вещественного числа с двумя знаками после запятой, единица измерения метры в секунду (м/с). Необходимо предусмотреть только вывод чисел.

В качестве разделителя целой и вещественной части – точка. Текст комментариев и единицы измерения выводить не нужно.
Пример того, что должна вывести программа (числа не связанны с данными, указанными в разделе "Формат ввода"):
9.44
3.03

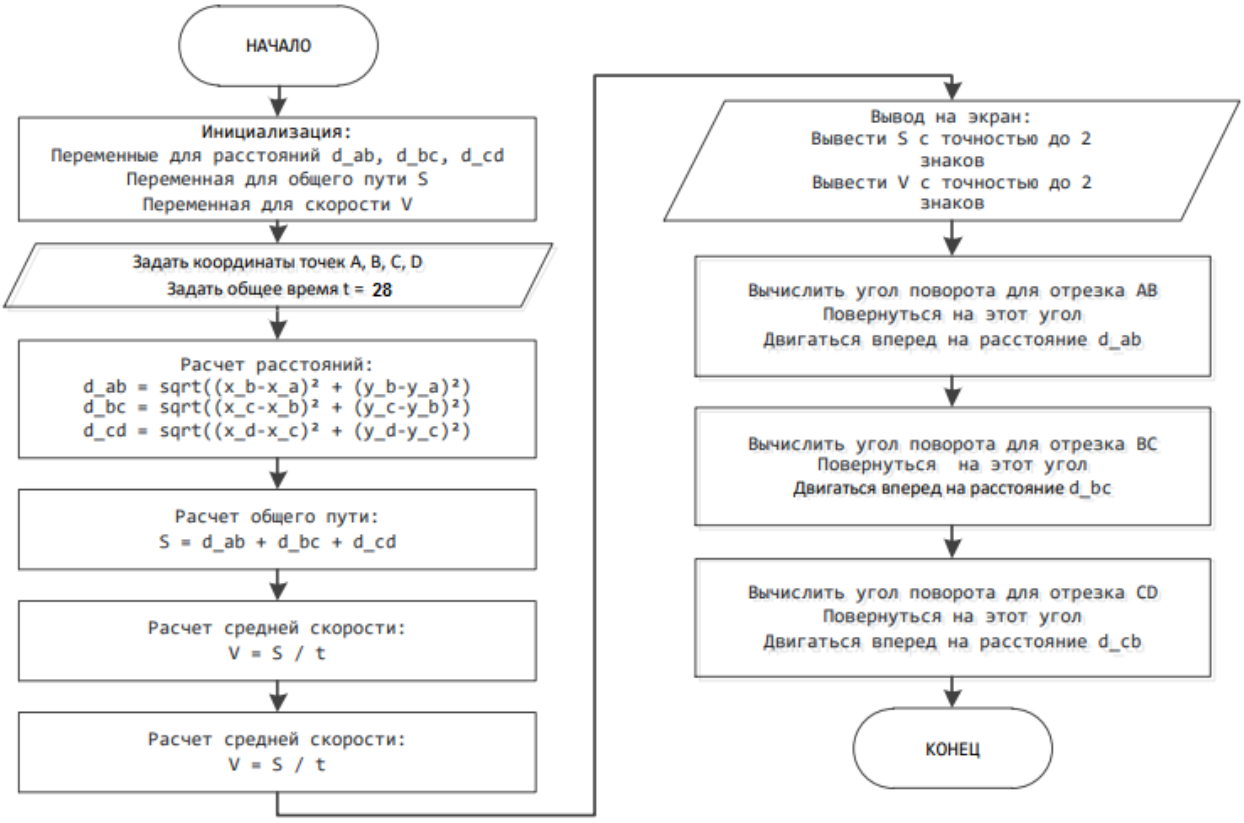
Исходные точки траектории движения.

Задание входных данных (x0,y0), (x1,y1), (x2,y2), (x3,y3), t для программы:
4.49 2.59
0.41 2.6
4.78 1.37
3.99 1.11
28.0

Задание № 1. Вычисление параметров движения

Расчтеные данные движения:
Участки пути.
td1=atan((2.6-2.59)/(4.49-0.41))
td2=atan((2.6-1.37)/(4.78-0.41))
td3=atan((1.37-1.11)/(4.78-3.99))
D_ab=sqrt((0.41-4.49)^2+(2.6-2.59)^2)
D_bc=sqrt((4.78-0.41)^2+(1.37-2.6)^2)
D_cd=sqrt((3.99-4.78)^2+(1.11-1.37)^2)
S=D_ab+D_bc+D_cd
V=S/28
A_ab=15.4/2*(0.985*pi-td1)
B_bc=15.4/2*(td1+pi-td2)
C_cd=15.4/2*(td2+1.15*pi-td3)
ena1=A_ab*360/pi/5.6
ena2=B_bc*360/pi/5.6
ena3=C_cd*360/pi/5.6
en1=D_ab*360/pi/5.6
en2=D_bc*360/pi/5.6
en3=D_cd*360/pi/5.6

Задание № 2. Блок-схема алгоритма



Задание № 3.1. Управляющая программа (в базовом варианте – управление движением по времени)

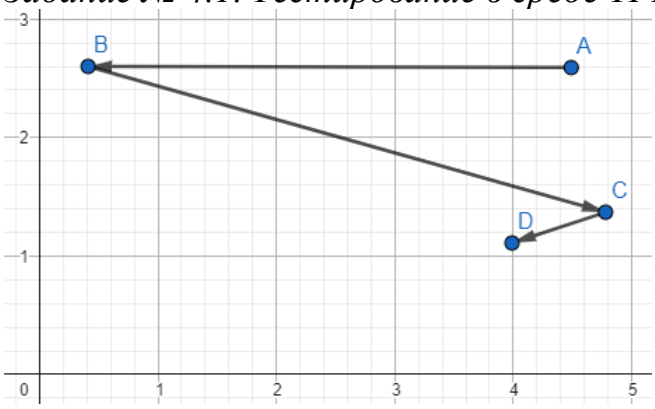
Задание № 3.2. Управляющая программа (в усложненном варианте – управление движением по показаниям энкодеров)

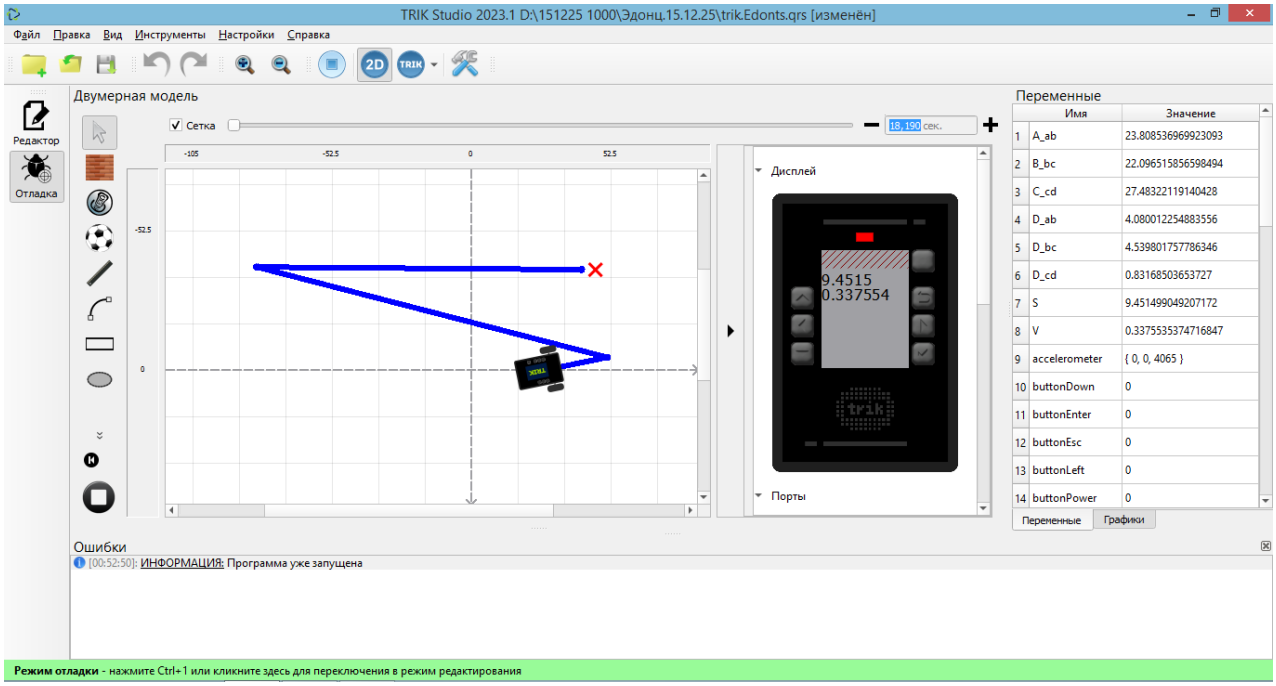
```
import sys
import time
import random
import math

class Program():
  __interpretation_started_timestamp__ = time.time() * 1000

  pi = 3.141592653589793
  A_ab = None
  B_bc = None
  C_cd = None
  D_ab = None
  D_bc = None
  D_cd = None
  S = None
  V = None
  en1 = None
  en2 = None
  en3 = None
  ena1 = None
  ena2 = None
  ena3 = None
  td1 = None
  td2 = None
  td3 = None

  def execMain(self):


    self.td1 = math.atan((2.6 - 2.59) / (4.49 - 0.41))
    self.td2 = math.atan((2.6 - 1.37) / (4.78 - 0.41))
    self.td3 = math.atan((1.37 - 1.11) / (4.78 - 3.99))
```

```
self.D_ab = math.sqrt(math.pow((0.41 - 4.49), 2) + math.pow((2.6 - 2.59), 2))
self.D_bc = math.sqrt(math.pow((4.78 - 0.41), 2) + math.pow((1.37 - 2.6), 2))
self.D_cd = math.sqrt(math.pow((3.99 - 4.78), 2) + math.pow((1.11 - 1.37), 2))
self.S = self.D_ab + self.D_bc + self.D_cd
self.V = self.S / 28
self.A_ab = 15.4 / 2 * (0.985 * self.pi - self.td1)
self.B_bc = 15.4 / 2 * (self.td1 + self.pi - self.td2)
self.C_cd = 15.4 / 2 * (self.td2 + 1.15 * self.pi - self.td3)
self.ena1 = self.A_ab * 360 / self.pi / 5.6
self.ena2 = self.B_bc * 360 / self.pi / 5.6
self.ena3 = self.C_cd * 360 / self.pi / 5.6
self.en1 = self.D_ab * 360 / self.pi / 5.6
self.en2 = self.D_bc * 360 / self.pi / 5.6
self.en3 = self.D_cd * 360 / self.pi / 5.6
brick.marker().down("blue")

brick.encoder("E1").reset()
brick.encoder("E2").reset()
brick.encoder("E3").reset()
brick.encoder("E4").reset()

brick.motor("M3").setPower(100)

brick.motor("M4").setPower(-(100))

while not (brick.encoder("E3").read() > int(self.ena1)):
  script.wait(10)

brick.encoder("E1").reset()
brick.encoder("E2").reset()
brick.encoder("E3").reset()
brick.encoder("E4").reset()

brick.motor("M3").setPower(100)
brick.motor("M4").setPower(100)

while not (brick.encoder("E3").read() > int(self.en1 * 30)):
  script.wait(10)

brick.encoder("E1").reset()
brick.encoder("E2").reset()
brick.encoder("E3").reset()
brick.encoder("E4").reset()

brick.motor("M3").setPower(100)

brick.motor("M4").setPower(-(100))

while not (brick.encoder("E3").read() > int(self.ena2)):
  script.wait(10)

brick.encoder("E1").reset()
brick.encoder("E2").reset()
brick.encoder("E3").reset()
brick.encoder("E4").reset()

brick.motor("M3").setPower(100)
brick.motor("M4").setPower(100)

while not (brick.encoder("E3").read() > int(self.en2 * 30)):
  script.wait(10)

brick.encoder("E1").reset()
brick.encoder("E2").reset()
brick.encoder("E3").reset()
brick.encoder("E4").reset()
```

```
    brick.motor("M3").setPower(100)

    brick.motor("M4").setPower(-(100))

    while not (brick.encoder("E3").read() > int(self.ena3)):
      script.wait(10)

    brick.encoder("E1").reset()
    brick.encoder("E2").reset()
    brick.encoder("E3").reset()
    brick.encoder("E4").reset()

    brick.motor("M3").setPower(100)
    brick.motor("M4").setPower(100)

    while not (brick.encoder("E3").read() > int(self.en3 * 25)):
      script.wait(10)

    brick.motor("M1").powerOff()
    brick.motor("M2").powerOff()
    brick.motor("M3").powerOff()
    brick.motor("M4").powerOff()

    brick.display().addLabel(str(self.S), 1, 1, 45)
    brick.display().redraw()

    brick.display().addLabel(str(self.V), 1, 45, 45)
    brick.display().redraw()

    script.wait(30000)

    brick.stop()
    return

def main():
  program = Program()
  program.execMain()

if __name__ == '__main__':
  main()
```

Задание № 4.1. Тестирование в среде ТРИК





Задание № 4.2. Тестирование на роботе

Робот двиггается по траектории из п. 1. На дисплее вводится две строки: первая - длинна всего маршрута в метрах, вторая - скорость робота в м/с.
На диспле выводится:

9.45
0.34

Кейс 2. Вариант 8.

Условие

В среде программирования Trick Studio в программе на языке Python задаются данные для перемещения мобильного трехколесного робота (или блоками). Программа должна по полученным данным вычислить среднюю скорость перемещения робота за заданное время при равномерном перемещении робота по траектории из четырех точек. Необходимо написать программу, в которую включить заданные точки маршрута для того, чтобы программа рассчитала и вывела значения длинны маршрута и средней скорости робота. Задание реализуется либо в Trick Studio, либо на реальном роботе.

Формат ввода

При написании программы необходимо учесть следующие заданные координаты точек маршрута:
x0,y0 – координаты нулевой точки (A) маршрута, м.;
x1,y1 – координаты первой точки (B) маршрута, м.;
x2,y2 – координаты второй точки (C) маршрута, м.;
x3,y3 – координаты третьей точки (D) маршрута, м.;
Робот движется равномерно по траектории. Параметры координат точек маршрута и времени его прохождения задаются в основном коде программы в виде не целых чисел. В каждой строке с координатами два числа, первое x, второе y. Числа в строке разделены одинарным пробелом. В пятой строке указывается одно число – время. Размерность координат – метры, времени – секунды и миллисекунды (ограничение в два знака после точки). Границы координат не выходят за пределы от 0 до 5 метров по x и y.
Пример задания входных данных (x0,y0), (x1,y1), (x2,y2), (x3,y3), t для программы:
0.34 1.94
2.03 1.61
1.72 0.65
0.49 0.67
31.0

Участники самостоятельно разрабатывают программный код, содержащий разделы считывания заданных координат, промежуточных расчетов и вывода данных на дисплей робота вычисленных расстояния между точками и скорости в м/с. Необходимо использовать следующие формулы для расчета параметров скорости и расстояния между всеми точками маршрута:

$V = \frac{L}{t}$, где V - скорость (м/с), L - длинна пути маршрута (м), t - время (сек),

|AB|= $\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$, где √ - корень квадратный, |AB| - модуль расстояния между соседними точками A и B с координатами (x1,y1), (x2,y2) соответственно. Остальные расстояния и формулы для участков BC, CD и DE задаются самостоятельно.

Формулы необходимо самостоятельно преобразовать к программному виду и включить в программу. Обозначения параметров функций и переменных участники могут выбирать самостоятельно.

Формат вывода

Необходимо чтобы робот проехал по заданным точкам в среде симуляции (или в реальности). Необходимо приложить к ответу видеозапись симуляции перемещения робота из Trick Studio. Код предоставлять в трех форматах: 1) скриншотами в отчете; 2) блоками в Trick Studio сохранен проект с работающей симуляцией; 3) в среде Trick Studio, но на языке Python.

Программа должна вывести две строки. В первой строке должно вывестись вычисленная программой длинна всего маршрута в виде вещественного числа с двумя знаками после запятой, единица измерения метры. Во второй строке должно вывестись вычисленная программой скорость робота в виде вещественного числа с двумя знаками после запятой, единица измерения метры в секунду (м/с). Необходимо предусмотреть только вывод чисел. В качестве разделителя целой и вещественной части – точка. Текст комментариев и единицы измерения выводить не нужно.

Пример того, что должна вывести программа (числа не связанны с данными, указанными в разделе "Формат ввода"):

9.44
3.03

Исходные точки траектории движения.

Задание входных данных (x0,y0), (x1,y1), (x2,y2), (x3,y3), t для программы:

0.34 1.94
2.03 1.61
1.72 0.65
0.49 0.67
31.0

Задание № 1. Вычисление параметров движения

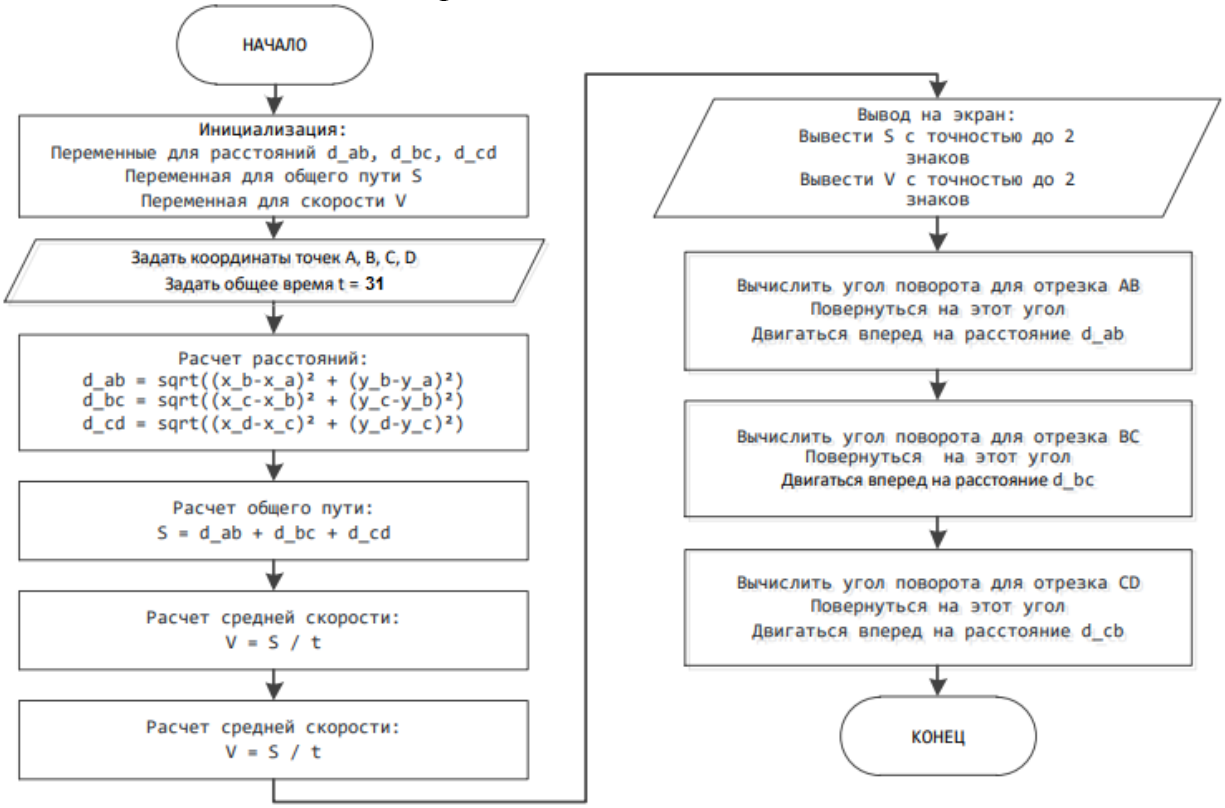
Расчтеные данные движения:

Участки пути.

```
td1=atan((1.94-1.61)/( 2.03-0.34))
td2=atan((1.61-0.65)/( 1.72-2.03))
td3=atan((0.67-0.65)/( 1.72-0.49))
D_ab=sqrt((2.03-0.34)^2+(1.61-1.94)^2)
D_bc=sqrt((1.72-2.03)^2+(0.65-1.61)^2)
D_cd=sqrt((0.49-1.72)^2+(0.67-0.65)^2)
S=D_ab+D_bc+D_cd
V=S/31
A_ab=15.4/2*(2*pi-td1)
B_bc=15.4/2*(td1+pi-td2)
C_cd=15.4/2*(td2+2*pi-td3)
ena1=A_ab*360/pi/5.6
ena2=B_bc*360/pi/5.6
```

ena3=C_cd*360/pi/5.6
en1=D_ab*360/pi/5.6
en2=D_bc*360/pi/5.6
en3=D_cd*360/pi/5.6

Задание № 2. Блок-схема алгоритма



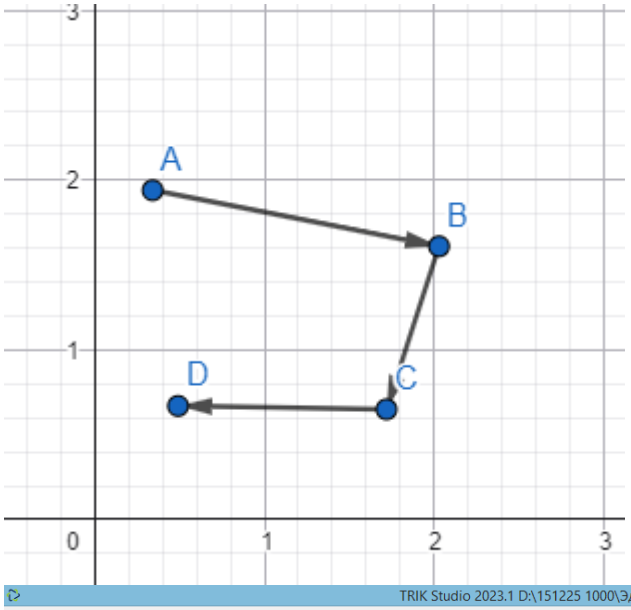
Задание № 3.1. Управляющая программа (в базовом варианте – управление движением по времени)
Задание № 3.2. Управляющая программа (в усложненном варианте – управление движением по показаниям энкодеров)

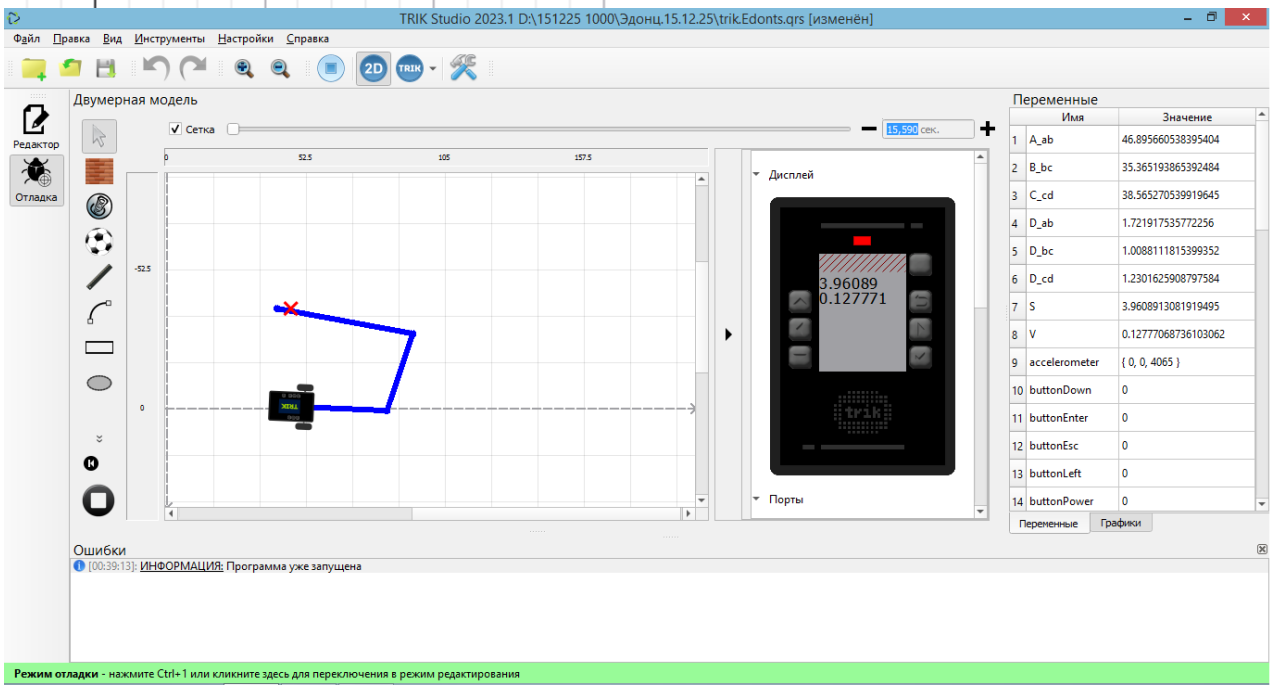
```
import sys
import time
import random
import math

class Program():
  __interpretation_started_timestamp__ = time.time() * 1000

  pi = 3.141592653589793
  A_ab = None
  B_bc = None
  C_cd = None
  D_ab = None
  D_bc = None
  D_cd = None
  S = None
  V = None
  en1 = None
  en2 = None
  en3 = None
  ena1 = None
  ena2 = None
  ena3 = None
  td1 = None
  td2 = None
  td3 = None
```

```
def execMain(self):


  self.td1 = math.atan((1.94 - 1.61) / (2.03 - 0.34))
  self.td2 = math.atan((1.61 - 0.65) / (1.72 - 2.03))
  self.td3 = math.atan((0.67 - 0.65) / (1.72 - 0.49))
  self.D_ab = math.sqrt(math.pow((2.03 - 0.34), 2) + math.pow((1.61 - 1.94), 2))
  self.D_bc = math.sqrt(math.pow((1.72 - 2.03), 2) + math.pow((0.65 - 1.61), 2))
  self.D_cd = math.sqrt(math.pow((0.49 - 1.72), 2) + math.pow((0.67 - 0.65), 2))
  self.S = self.D_ab + self.D_bc + self.D_cd
  self.V = self.S / 31
  self.A_ab = 15.4 / 2 * (2 * self.pi - self.td1)
  self.B_bc = 15.4 / 2 * (self.td1 + self.pi - self.td2)
  self.C_cd = 15.4 / 2 * (self.td2 + 2 * self.pi - self.td3)
  self.ena1 = self.A_ab * 360 / self.pi / 5.6
  self.ena2 = self.B_bc * 360 / self.pi / 5.6
  self.ena3 = self.C_cd * 360 / self.pi / 5.6
  self.en1 = self.D_ab * 360 / self.pi / 5.6
  self.en2 = self.D_bc * 360 / self.pi / 5.6
  self.en3 = self.D_cd * 360 / self.pi / 5.6
  brick.marker().down("blue")

  brick.encoder("E1").reset()
  brick.encoder("E2").reset()
  brick.encoder("E3").reset()
  brick.encoder("E4").reset()

  brick.motor("M3").setPower(100)

  brick.motor("M4").setPower(-(100))

  while not (brick.encoder("E3").read() > int(self.ena1)):
    script.wait(10)

  brick.encoder("E1").reset()
  brick.encoder("E2").reset()
  brick.encoder("E3").reset()
  brick.encoder("E4").reset()

  brick.motor("M3").setPower(100)
  brick.motor("M4").setPower(100)

  while not (brick.encoder("E3").read() > int(self.en1 * 30)):
    script.wait(10)

  brick.encoder("E1").reset()
  brick.encoder("E2").reset()
  brick.encoder("E3").reset()
  brick.encoder("E4").reset()

  brick.motor("M3").setPower(100)

  brick.motor("M4").setPower(-(100))

  while not (brick.encoder("E3").read() > int(self.ena2)):
    script.wait(10)

  brick.encoder("E1").reset()
  brick.encoder("E2").reset()
  brick.encoder("E3").reset()
  brick.encoder("E4").reset()

  brick.motor("M3").setPower(100)
  brick.motor("M4").setPower(100)

  while not (brick.encoder("E3").read() > int(self.en2 * 30)):
```

```
      script.wait(10)

    brick.encoder("E1").reset()
    brick.encoder("E2").reset()
    brick.encoder("E3").reset()
    brick.encoder("E4").reset()

    brick.motor("M3").setPower(100)

    brick.motor("M4").setPower(-(100))

    while not (brick.encoder("E3").read() > int(self.ena3)):
      script.wait(10)

    brick.encoder("E1").reset()
    brick.encoder("E2").reset()
    brick.encoder("E3").reset()
    brick.encoder("E4").reset()

    brick.motor("M3").setPower(100)
    brick.motor("M4").setPower(100)

    while not (brick.encoder("E3").read() > int(self.en3 * 25)):
      script.wait(10)

    brick.motor("M1").powerOff()
    brick.motor("M2").powerOff()
    brick.motor("M3").powerOff()
    brick.motor("M4").powerOff()

    brick.display().addLabel(str(self.S), 1, 1, 45)
    brick.display().redraw()

    brick.display().addLabel(str(self.V), 1, 45, 45)
    brick.display().redraw()

    script.wait(30000)

    brick.stop()
    return

def main():
  program = Program()
  program.execMain()

if __name__ == '__main__':
  main()
```

Задание № 4.1. Тестирование в среде ТРИК





Задание № 4.2. Тестирование на роботе

Робот двиггается по траектории из п. 1. На дисплее вводится две строки: первая - длинна всего маршрута в метрах, вторая - скорость робота в м/с. На диспле выводится:

3.96

0.13

Кейс 2. Вариант 9.

Условие

В среде программирования Trick Studio в программе на языке Python задаются данные для перемещения мобильного трехколесного робота (или блоками). Программа должна по полученным данным вычислить среднюю скорость перемещения робота за заданное время при равномерном перемещении робота по траектории из четырех точек. Необходимо написать программу, в которую включить заданные точки маршрута для того, чтобы программа рассчитала и вывела значения длинны маршрута и средней скорости робота. Задание реализуется либо в Trick Studio, либо на реальном роботе.

Формат ввода

При написании программы необходимо учесть следующие заданные координаты точек маршрута:

x0,y0 – координаты нулевой точки (A) маршрута, м.;

x1,y1 – координаты первой точки (B) маршрута, м.;
x2,y2 – координаты второй точки (C) маршрута, м.;
x3,y3 – координаты третьей точки (D) маршрута, м.;
Робот движется равномерно по траектории. Параметры координат точек маршрута и времени его прохождения задаются в основном коде программы в виде не целых чисел. В каждой строке с координатами два числа, первое x, второе y. Числа в строке разделены одинарным пробелом. В пятой строке указывается одно число – время. Размерность координат – метры, времени – секунды и миллисекунды (ограничение в два знака после точки). Границы координат не выходят за пределы от 0 до 5 метров по x и y.
Пример задания входных данных (x0,y0), (x1,y1), (x2,y2), (x3,y3), t для программы:
3.78 0.38
0.61 3.33
0.41 1.11
3.61 2.04
27.0
Участники самостоятельно разрабатывают программный код, содержащий разделы считывания заданных координат, промежуточных расчетов и вывода данных на дисплей робота вычисленных расстояния между точками и скорости в м/с. Необходимо использовать следующие формулы для расчета параметров скорости и расстояния между всеми точками маршрута:
$V = \frac{L}{t}$, где V - скорость (м/с), L - длинна пути маршрута (м), t - время (сек),
|AB|= $\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$, где √ - корень квадратный, |AB| - модуль расстояния между соседними точками A и B с координатами (x1,y1), (x2,y2) соответственно. Остальные расстояния и формулы для участков BC, CD и DE задаются самостоятельно.
Формулы необходимо самостоятельно преобразовать к программному виду и включить в программу. Обозначения параметров функций и переменных участники могут выбирать самостоятельно.

Формат вывода

Необходимо чтобы робот проехал по заданным точкам в среде симуляции (или в реальности). Необходимо приложить к ответу видеозапись симуляции перемещения робота из Trick Studio. Код предоставлять в трех форматах: 1) скриншотами в отчете; 2) блоками в Trick Studio сохранен проект с работающей симуляцией; 3) в среде Trick Studio, но на языке Python.
Программа должна вывести две строки. В первой строке должно вывестись вычисленная программой длинна всего маршрута в виде вещественного числа с двумя знаками после запятой, единица измерения метры. Во второй строке должно вывестись вычисленная программой скорость робота в виде вещественного числа с двумя знаками после запятой, единица измерения метры в секунду (м/с). Необходимо предусмотреть только вывод чисел. В качестве разделителя целой и вещественной части – точка. Текст комментариев и единицы измерения выводить не нужно.
Пример того, что должна вывести программа (числа не связанны с данными, указанными в разделе "Формат ввода"):
9.44
3.03

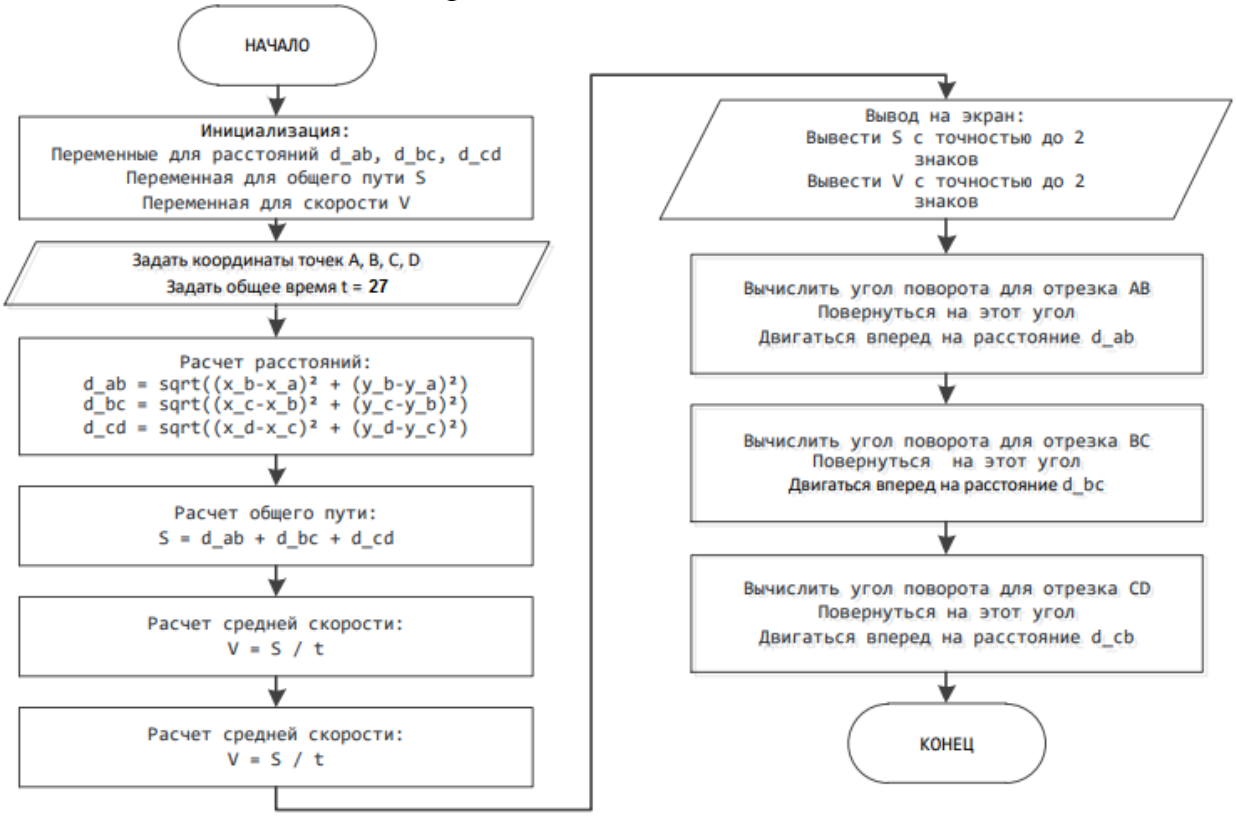
Исходные точки траектории движения.

Задание входных данных (x0,y0), (x1,y1), (x2,y2), (x3,y3), t для программы:
3.78 0.38
0.61 3.33
0.41 1.11
3.61 2.04
27.0

Задание № 1. Вычисление параметров движения

Расчтеные данные движения:
Участки пути.
td1=atan((3.33-0.38)/(3.78-0.61))
td2=atan((3.33-1.11)/(0.61-0.41))
td3=atan((2.04-1.11)/(3.61-0.41))
D_ab=sqrt((0.61-3.78)^2+(3.33-0.38)^2)
D_bc=sqrt((0.41-0.61)^2+(1.11-3.33)^2)
D_cd=sqrt((3.61-0.41)^2+(2.04-1.11)^2)
S=D_ab+D_bc+D_cd
V=S/27
A_ab=15.4/2*(2*pi-td1)
B_bc=15.4/2*(td1+pi-td2)
C_cd=15.4/2*(td2+pi-td3)
ena1=A_ab*360/pi/5.6
ena2=B_bc*360/pi/5.6
ena3=C_cd*360/pi/5.6
en1=D_ab*360/pi/5.6
en2=D_bc*360/pi/5.6
en3=D_cd*360/pi/5.6

Задание № 2. Блок-схема алгоритма



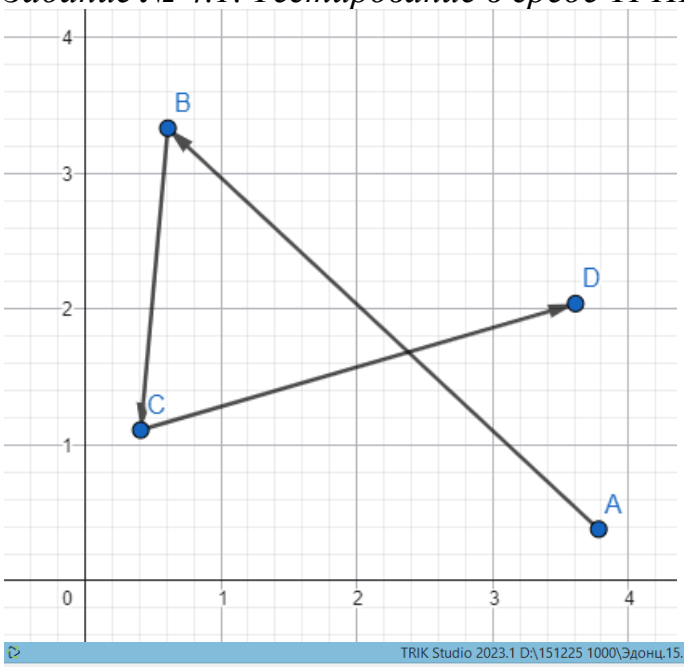
Задание № 3.1. Управляющая программа (в базовом варианте – управление движением по времени)
Задание № 3.2. Управляющая программа (в усложненном варианте – управление движением по показаниям энкодеров)

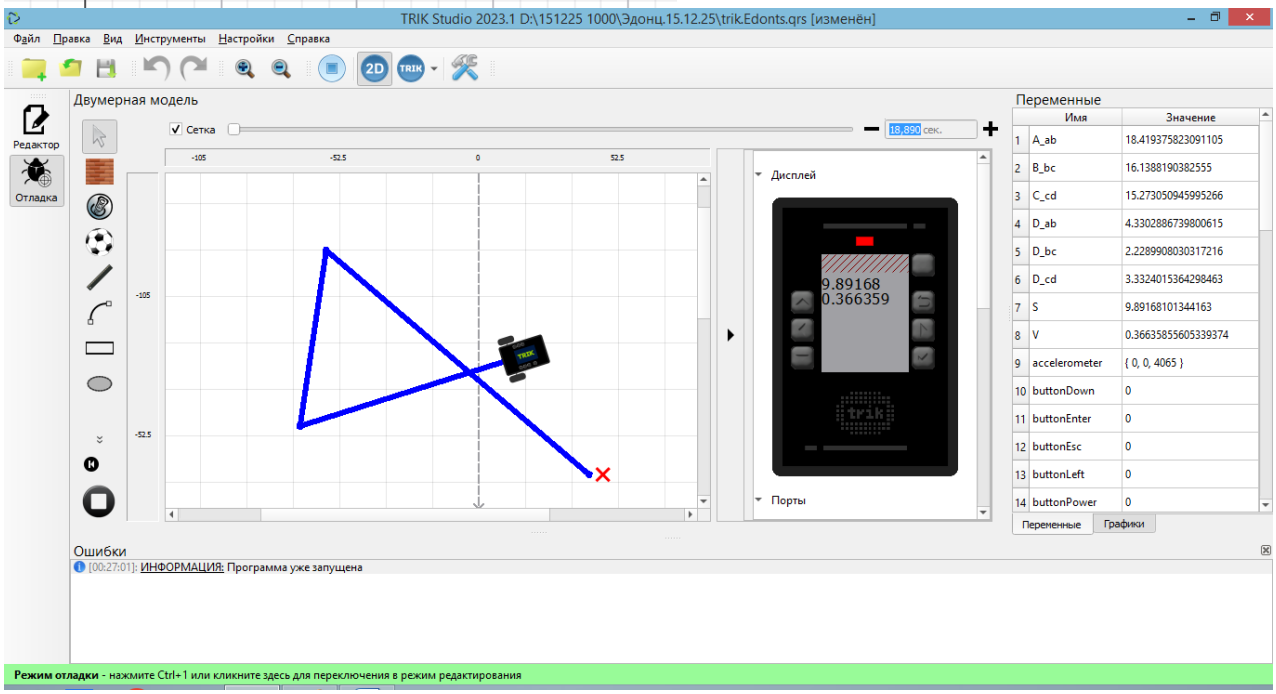
```
import sys
import time
import random
import math

class Program():
  __interpretation_started_timestamp__ = time.time() * 1000

  pi = 3.141592653589793
```

```
A_ab = None
B_bc = None
C_cd = None
D_ab = None
D_bc = None
D_cd = None
S = None
V = None
en1 = None
en2 = None
en3 = None
ena1 = None
ena2 = None
ena3 = None
td1 = None
td2 = None
td3 = None

def execMain(self):


  self.td1 = math.atan((3.33 - 0.38) / (3.78 - 0.61))
  self.td2 = math.atan((3.33 - 1.11) / (0.61 - 0.41))
  self.td3 = math.atan((2.04 - 1.11) / (3.61 - 0.41))
  self.D_ab = math.sqrt(math.pow((0.61 - 3.78), 2) + math.pow((3.33 - 0.38), 2))
  self.D_bc = math.sqrt(math.pow((0.41 - 0.61), 2) + math.pow((1.11 - 3.33), 2))
  self.D_cd = math.sqrt(math.pow((3.61 - 0.41), 2) + math.pow((2.04 - 1.11), 2))
  self.S = self.D_ab + self.D_bc + self.D_cd
  self.V = self.S / 27
  self.A_ab = 15.4 / 2 * (self.pi - self.td1)
  self.B_bc = 15.4 / 2 * (self.td1 + 0.9 * self.pi - self.td2)
  self.C_cd = 15.4 / 2 * (self.td2 + 0.25 * self.pi - self.td3)
  self.ena1 = self.A_ab * 360 / self.pi / 5.6
  self.ena2 = self.B_bc * 360 / self.pi / 5.6
  self.ena3 = self.C_cd * 360 / self.pi / 5.6
  self.en1 = self.D_ab * 360 / self.pi / 5.6
  self.en2 = self.D_bc * 360 / self.pi / 5.6
  self.en3 = self.D_cd * 360 / self.pi / 5.6
  brick.marker().down("blue")

  brick.encoder("E1").reset()
  brick.encoder("E2").reset()
  brick.encoder("E3").reset()
  brick.encoder("E4").reset()

  brick.motor("M3").setPower(100)

  brick.motor("M4").setPower(-(100))

  while not (brick.encoder("E3").read() > int(self.ena1)):
    script.wait(10)

  brick.encoder("E1").reset()
  brick.encoder("E2").reset()
  brick.encoder("E3").reset()
  brick.encoder("E4").reset()

  brick.motor("M3").setPower(100)
  brick.motor("M4").setPower(100)

  while not (brick.encoder("E3").read() > int(self.en1 * 30)):
    script.wait(10)

  brick.encoder("E1").reset()
  brick.encoder("E2").reset()
  brick.encoder("E3").reset()
```

```
    brick.encoder("E4").reset()

    brick.motor("M3").setPower(100)

    brick.motor("M4").setPower(-(100))

    while not (brick.encoder("E3").read() > int(self.ena2)):
      script.wait(10)

    brick.encoder("E1").reset()
    brick.encoder("E2").reset()
    brick.encoder("E3").reset()
    brick.encoder("E4").reset()

    brick.motor("M3").setPower(100)
    brick.motor("M4").setPower(100)

    while not (brick.encoder("E3").read() > int(self.en2 * 30)):
      script.wait(10)

    brick.encoder("E1").reset()
    brick.encoder("E2").reset()
    brick.encoder("E3").reset()
    brick.encoder("E4").reset()

    brick.motor("M3").setPower(100)

    brick.motor("M4").setPower(-(100))

    while not (brick.encoder("E3").read() > int(self.ena3)):
      script.wait(10)

    brick.encoder("E1").reset()
    brick.encoder("E2").reset()
    brick.encoder("E3").reset()
    brick.encoder("E4").reset()

    brick.motor("M3").setPower(100)
    brick.motor("M4").setPower(100)

    while not (brick.encoder("E3").read() > int(self.en3 * 25)):
      script.wait(10)

    brick.motor("M1").powerOff()
    brick.motor("M2").powerOff()
    brick.motor("M3").powerOff()
    brick.motor("M4").powerOff()

    brick.display().addLabel(str(self.S), 1, 1, 45)
    brick.display().redraw()

    brick.display().addLabel(str(self.V), 1, 45, 45)
    brick.display().redraw()

    script.wait(30000)

    brick.stop()
    return

def main():
  program = Program()
  program.execMain()

if __name__ == '__main__':
  main()
```

Задание № 4.1. Тестирование в среде ТРИК





Задание № 4.2. Тестирование на роботе

Робот двиггается по траектории из п. 1. На дисплее вводится две строки: первая - длинна всего маршрута в метрах, вторая - скорость робота в м/с. На диспле выводится:

9.89
0.37

Кейс 2. Вариант 10.

Условие

В среде программирования Trick Studio в программе на языке Python задаются данные для перемещения мобильного трехколесного робота (или блоками). Программа должна по полученным данным вычислить среднюю скорость перемещения робота за заданное время при равномерном перемещении робота по траектории из четырех точек. Необходимо написать программу, в которую включить заданные точки маршрута для того, чтобы программа рассчитала и вывела значения длинны маршрута и средней скорости робота. Задание реализуется либо в Trick Studio, либо на реальном роботе.

Формат ввода

При написании программы необходимо учесть следующие заданные координаты точек маршрута:
x0,y0 – координаты нулевой точки (A) маршрута, м.;
x1,y1 – координаты первой точки (B) маршрута, м.;
x2,y2 – координаты второй точки (C) маршрута, м.;
x3,y3 – координаты третьей точки (D) маршрута, м.;
Робот движется равномерно по траектории. Параметры координат точек маршрута и времени его прохождения задаются в основном коде программы в виде не целых чисел. В каждой строке с координатами два числа, первое x, второе y. Числа в строке разделены одинарным пробелом. В пятой строке указывается одно число – время. Размерность координат – метры, времени – секунды и миллисекунды (ограничение в два знака после точки). Границы координат не выходят за пределы от 0 до 5 метров по x и y.
Пример задания входных данных (x0,y0), (x1,y1), (x2,y2), (x3,y3), t для программы:
2.73 2.04
4.72 0.93
0.83 2.65
1.79 1.01
24.0
Участники самостоятельно разрабатывают программный код, содержащий разделы считывания заданных координат, промежуточных расчетов и вывода данных на дисплей робота вычисленных расстояния между точками и скорости в м/с. Необходимо использовать следующие формулы для расчета параметров скорости и расстояния между всеми точками маршрута:
$V = \frac{L}{t}$, где V - скорость (м/с), L - длинна пути маршрута (м), t - время (сек),
|AB|= $\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$, где √ - корень квадратный, |AB| - модуль расстояния между соседними точками A и B с координатами (x1,y1), (x2,y2) соответственно. Остальные расстояния и формулы для участков BC, CD и DE задаются самостоятельно.
Формулы необходимо самостоятельно преобразовать к программному виду и включить в программу. Обозначения параметров функций и переменных участники могут выбирать самостоятельно.

Формат вывода

Необходимо чтобы робот проехал по заданным точкам в среде симуляции (или в реальности). Необходимо приложить к ответу видеозапись симуляции перемещения робота из Trick Studio. Код предоставлять в трех форматах: 1) скриншотами в отчете; 2) блоками в Trick Studio сохранен проект с работающей симуляцией; 3) в среде Trick Studio, но на языке Python.
Программа должна вывести две строки. В первой строке должно вывестись вычисленная программой длинна всего маршрута в виде вещественного числа с двумя знаками после запятой, единица измерения метры. Во второй строке должно вывестись вычисленная программой скорость робота в виде вещественного числа с двумя знаками после запятой, единица измерения метры в секунду (м/с). Необходимо предусмотреть только вывод чисел. В качестве разделителя целой и вещественной части – точка. Текст комментариев и единицы измерения выводить не нужно.
Пример того, что должна вывести программа (числа не связанны с данными, указанными в разделе "Формат ввода"):
9.44
3.03

Исходные точки траектории движения.

Задание входных данных (x0,y0), (x1,y1), (x2,y2), (x3,y3), t для программы:
2.73 2.04
4.72 0.93
0.83 2.65
1.79 1.01

24.0

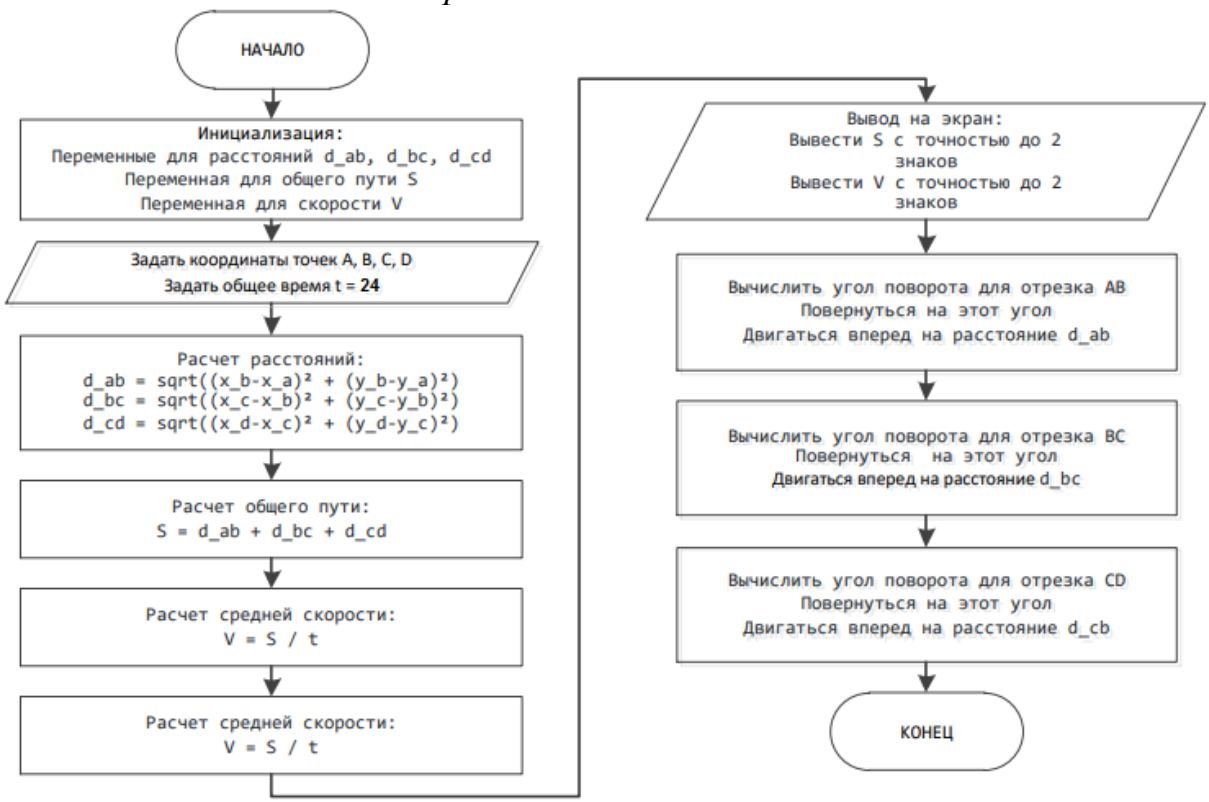
Задание № 1. Вычисление параметров движения

Расчтеные данные движения:

Участки пути.

td1=atan((2.04-0.93)/(4.72-2.73))
td2=atan((2.65-0.93)/(4.72-0.83))
td3=atan((2.65-1.01)/(1.79-0.83))
D_ab=sqrt((4.72-2.73)^2+(0.93-2.04)^2)
D_bc=sqrt((0.83-4.72)^2+(2.65-0.93)^2)
D_cd=sqrt((1.79-0.83)^2+(1.01-2.65)^2)
S=D_ab+D_bc+D_cd
V=S/24
A_ab=15.4/2*(2*pi-td1)
B_bc=15.4/2*(td1+pi-td2)
C_cd=15.4/2*(td2+pi-td3)
ena1=A_ab*360/pi/5.6
ena2=B_bc*360/pi/5.6
ena3=C_cd*360/pi/5.6
en1=D_ab*360/pi/5.6
en2=D_bc*360/pi/5.6
en3=D_cd*360/pi/5.6

Задание № 2. Блок-схема алгоритма



Задание № 3.1. Управляющая программа (в базовом варианте – управление движением по времени)

Задание № 3.2. Управляющая программа (в усложненном варианте – управление движением по показаниям энкодеров)

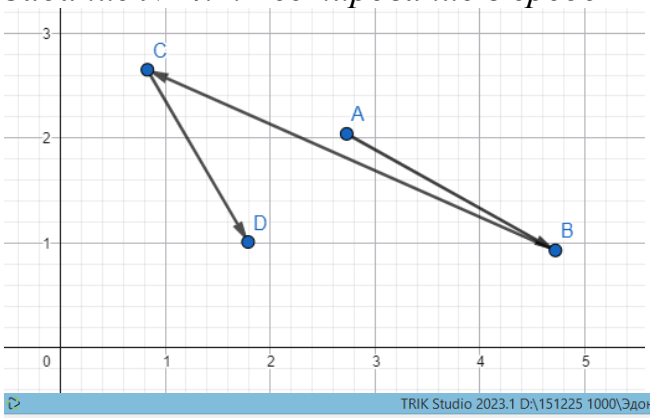
```
import sys
import time
import random
import math
```

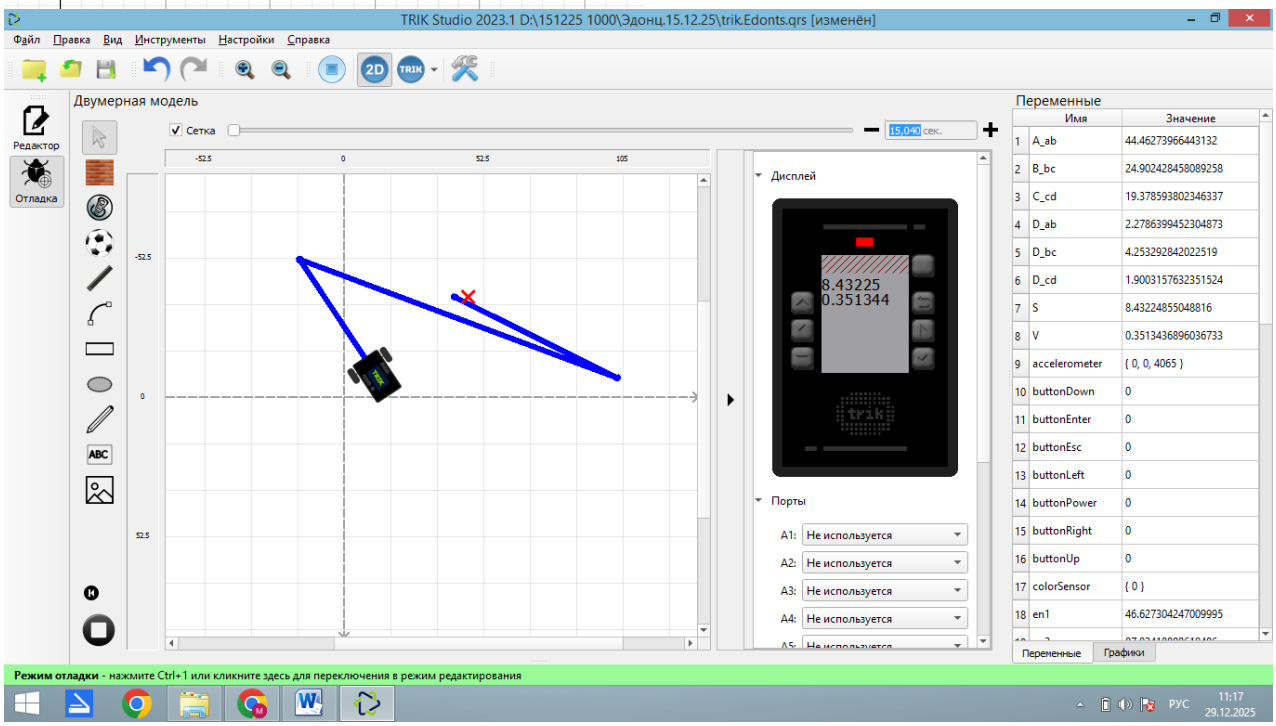
```
class Program():
  __interpretation_started_timestamp__ = time.time() * 1000

  pi = 3.141592653589793
  A_ab = None
  B_bc = None
  C_cd = None
  D_ab = None
  D_bc = None
  D_cd = None
  S = None
  V = None
  en1 = None
  en2 = None
  en3 = None
  ena1 = None
  ena2 = None
  ena3 = None
  td1 = None
  td2 = None
  td3 = None

  def execMain(self):

    self.td1 = math.atan((2.04 - 0.93) / (4.72 - 2.73))
    self.td2 = math.atan((2.65 - 0.93) / (4.72 - 0.83))
    self.td3 = math.atan((2.65 - 1.01) / (1.79 - 0.83))
    self.D_ab = math.sqrt(math.pow((4.72 - 2.73), 2) + math.pow((0.93 - 2.04), 2))
    self.D_bc = math.sqrt(math.pow((0.83 - 4.72), 2) + math.pow((2.65 - 0.93), 2))
    self.D_cd = math.sqrt(math.pow((1.79 - 0.83), 2) + math.pow((1.01 - 2.65), 2))
    self.S = self.D_ab + self.D_bc + self.D_cd
    self.V = self.S / 24
    self.A_ab = 15.4 / 2 * (2 * self.pi - self.td1)
    self.B_bc = 15.4 / 2 * (self.td1 + self.pi - self.td2)
    self.C_cd = 15.4 / 2 * (self.td2 + self.pi - self.td3)
    self.ena1 = self.A_ab * 360 / self.pi / 5.6
    self.ena2 = self.B_bc * 360 / self.pi / 5.6
    self.ena3 = self.C_cd * 360 / self.pi / 5.6
    self.en1 = self.D_ab * 360 / self.pi / 5.6
    self.en2 = self.D_bc * 360 / self.pi / 5.6
    self.en3 = self.D_cd * 360 / self.pi / 5.6
    brick.marker().down("blue")

    brick.encoder("E1").reset()
    brick.encoder("E2").reset()
    brick.encoder("E3").reset()
    brick.encoder("E4").reset()

    brick.motor("M3").setPower(100)

    brick.motor("M4").setPower(-(100))

    while not (brick.encoder("E3").read() > int(self.ena1)):
      script.wait(10)

    brick.encoder("E1").reset()
    brick.encoder("E2").reset()
    brick.encoder("E3").reset()
    brick.encoder("E4").reset()

    brick.motor("M3").setPower(100)
    brick.motor("M4").setPower(100)

    while not (brick.encoder("E3").read() > int(self.en1 * 30)):
      script.wait(10)
```

```
    brick.encoder("E1").reset()
    brick.encoder("E2").reset()
    brick.encoder("E3").reset()
    brick.encoder("E4").reset()

    brick.motor("M3").setPower(100)

    brick.motor("M4").setPower(-(100))

    while not (brick.encoder("E3").read() > int(self.ena2)):
      script.wait(10)

    brick.encoder("E1").reset()
    brick.encoder("E2").reset()
    brick.encoder("E3").reset()
    brick.encoder("E4").reset()

    brick.motor("M3").setPower(100)
    brick.motor("M4").setPower(100)

    while not (brick.encoder("E3").read() > int(self.en2 * 30)):
      script.wait(10)

    brick.encoder("E1").reset()
    brick.encoder("E2").reset()
    brick.encoder("E3").reset()
    brick.encoder("E4").reset()

    brick.motor("M3").setPower(100)

    brick.motor("M4").setPower(-(100))

    while not (brick.encoder("E3").read() > int(self.ena3)):
      script.wait(10)

    brick.encoder("E1").reset()
    brick.encoder("E2").reset()
    brick.encoder("E3").reset()
    brick.encoder("E4").reset()

    brick.motor("M3").setPower(100)
    brick.motor("M4").setPower(100)

    while not (brick.encoder("E3").read() > int(self.en3 * 30)):
      script.wait(10)

    brick.motor("M1").powerOff()
    brick.motor("M2").powerOff()
    brick.motor("M3").powerOff()
    brick.motor("M4").powerOff()

    brick.display().addLabel(str(self.S), 1, 1, 45)
    brick.display().redraw()

    brick.display().addLabel(str(self.V), 1, 45, 45)
    brick.display().redraw()

    script.wait(30000)

    brick.stop()
    return

def main():
  program = Program()
  program.execMain()
```

```
if __name__ == '__main__':
  main()
```

Задание № 4.1. Тестирование в среде ТРИК





Задание № 4.2. Тестирование на роботе

Робот двиггается по траектории из п. 1. На дисплее вводится две строки: первая - длинна всего маршрута в метрах, вторая - скорость робота в м/с. На диспле выводится:

8.43
0.35

Кейс 2. Вариант 11.

Условие

В среде программирования Trick Studio в программе на языке Python задаются данные для перемещения мобильного трехколесного робота (или блоками). Программа должна по полученным данным вычислить среднюю скорость перемещения робота за заданное время при равномерном перемещении робота по траектории из четырех точек. Необходимо написать программу, в которую включить заданные точки маршрута для того, чтобы программа рассчитала и вывела значения длинны маршрута и средней скорости робота. Задание реализуется либо в Trick Studio, либо на реальном роботе.

Формат ввода

При написании программы необходимо учесть следующие заданные координаты точек маршрута:

x0,y0 – координаты нулевой точки (A) маршрута, м.;

x1,y1 – координаты первой точки (B) маршрута, м.;
x2,y2 – координаты второй точки (C) маршрута, м.;
x3,y3 – координаты третьей точки (D) маршрута, м.;
Робот движется равномерно по траектории. Параметры координат точек маршрута и времени его прохождения задаются в основном коде программы в виде не целых чисел. В каждой строке с координатами два числа, первое x, второе y. Числа в строке разделены одинарным пробелом. В пятой строке указывается одно число – время. Размерность координат – метры, времени – секунды и миллисекунды (ограничение в два знака после точки). Границы координат не выходят за пределы от 0 до 5 метров по x и y.
Пример задания входных данных (x0,y0), (x1,y1), (x2,y2), (x3,y3), t для программы:
3.65 3.57
3.82 4.48
2.36 2.35
0.06 3.11
34.0
Участники самостоятельно разрабатывают программный код, содержащий разделы считывания заданных координат, промежуточных расчетов и вывода данных на дисплей робота вычисленных расстояния между точками и скорости в м/с. Необходимо использовать следующие формулы для расчета параметров скорости и расстояния между всеми точками маршрута:
$V = \frac{L}{t}$, где V - скорость (м/с), L - длинна пути маршрута (м), t - время (сек),
|AB|= $\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$, где √ - корень квадратный, |AB| - модуль расстояния между соседними точками A и B с координатами (x1,y1), (x2,y2) соответственно. Остальные расстояния и формулы для участков BC, CD и DE задаются самостоятельно.
Формулы необходимо самостоятельно преобразовать к программному виду и включить в программу. Обозначения параметров функций и переменных участники могут выбирать самостоятельно.

Формат вывода

Необходимо чтобы робот проехал по заданным точкам в среде симуляции (или в реальности). Необходимо приложить к ответу видеозапись симуляции перемещения робота из Trick Studio. Код предоставлять в трех форматах: 1) скриншотами в отчете; 2) блоками в Trick Studio сохранен проект с работающей симуляцией; 3) в среде Trick Studio, но на языке Python.
Программа должна вывести две строки. В первой строке должно вывестись вычисленная программой длинна всего маршрута в виде вещественного числа с двумя знаками после запятой, единица измерения метры. Во второй строке должно вывестись вычисленная программой скорость робота в виде вещественного числа с двумя знаками после запятой, единица измерения метры в секунду (м/с). Необходимо предусмотреть только вывод чисел. В качестве разделителя целой и вещественной части – точка. Текст комментариев и единицы измерения выводить не нужно.
Пример того, что должна вывести программа (числа не связанны с данными, указанными в разделе "Формат ввода"):
9.44
3.03

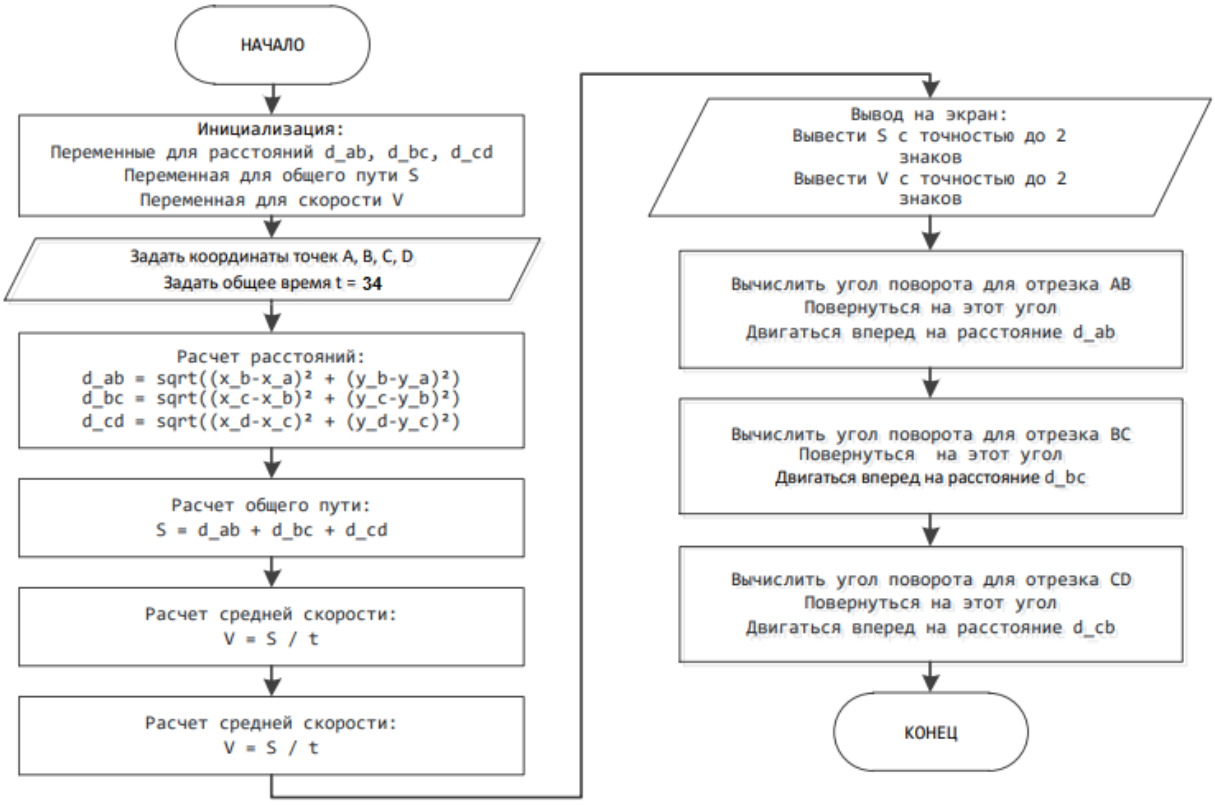
Исходные точки траектории движения.

Задание входных данных (x0,y0), (x1,y1), (x2,y2), (x3,y3), t для программы:
3.65 3.57
3.82 4.48
2.36 2.35
0.06 3.11
34.0

Задание № 1. Вычисление параметров движения

Расчтеные данные движения:
Участки пути.
td1=atan((4.48-3.57)/(3.82-3.65))
td2=atan((4.48-2.35)/(3.82-2.36))
td3=atan((3.11-2.35)/(2.36-0.06))
D_ab=sqrt((3.82-3.65)^2+(4.48-3.57)^2)
D_bc=sqrt((2.36-3.82)^2+(2.35-4.48)^2)
D_cd=sqrt((0.06-2.36)^2+(3.11-2.35)^2)
S=D_ab+D_bc+D_cd
V=S/34
A_ab=15.4/2*(0.85*pi-td1)
B_bc=15.4/2*(td1+0.725*pi-td2)
C_cd=15.4/2*(td2+1.4*pi-td3)
ena1=A_ab*360/pi/5.6
ena2=B_bc*360/pi/5.6
ena3=C_cd*360/pi/5.6
en1=D_ab*360/pi/5.6
en2=D_bc*360/pi/5.6
en3=D_cd*360/pi/5.6

Задание № 2. Блок-схема алгоритма



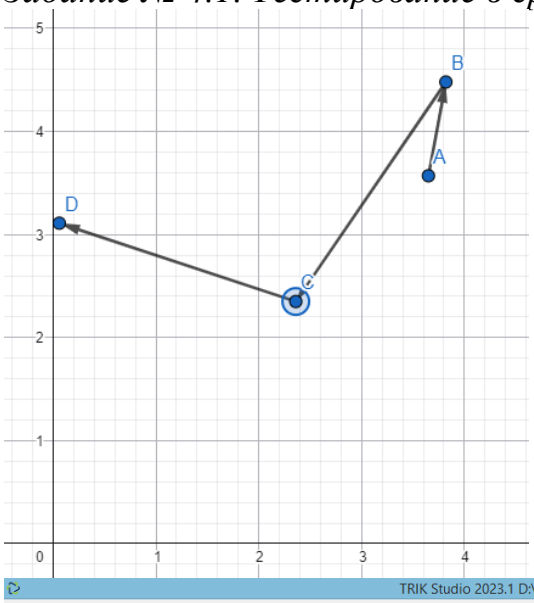
Задание № 3.1. Управляющая программа (в базовом варианте – управление движением по времени)
Задание № 3.2. Управляющая программа (в усложненном варианте – управление движением по показаниям энкодеров)

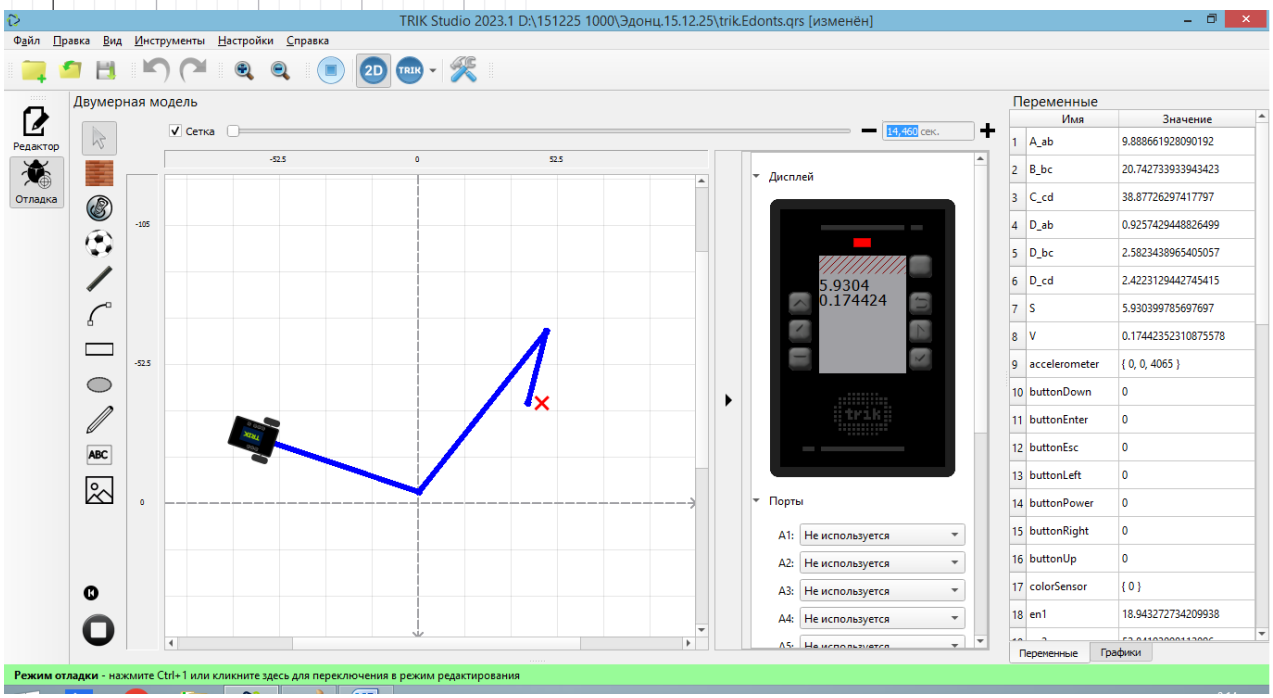
```
import sys
import time
import random
import math

class Program():
  __interpretation_started_timestamp__ = time.time() * 1000

  pi = 3.141592653589793
```

```
A_ab = None
B_bc = None
C_cd = None
D_ab = None
D_bc = None
D_cd = None
S = None
V = None
en1 = None
en2 = None
en3 = None
ena1 = None
ena2 = None
ena3 = None
td1 = None
td2 = None
td3 = None

def execMain(self):


  self.td1 = math.atan((4.48 - 3.57) / (3.82 - 3.65))
  self.td2 = math.atan((4.48 - 2.35) / (3.82 - 2.36))
  self.td3 = math.atan((3.11 - 2.35) / (2.36 - 0.06))
  self.D_ab = math.sqrt(math.pow((3.82 - 3.65), 2) + math.pow((4.48 - 3.57), 2))
  self.D_bc = math.sqrt(math.pow((2.36 - 3.82), 2) + math.pow((2.35 - 4.48), 2))
  self.D_cd = math.sqrt(math.pow((0.06 - 2.36), 2) + math.pow((3.11 - 2.35), 2))
  self.S = self.D_ab + self.D_bc + self.D_cd
  self.V = self.S / 34
  self.A_ab = 15.4 / 2 * (0.85 * self.pi - self.td1)
  self.B_bc = 15.4 / 2 * (self.td1 + 0.725 * self.pi - self.td2)
  self.C_cd = 15.4 / 2 * (self.td2 + 1.4 * self.pi - self.td3)
  self.ena1 = self.A_ab * 360 / self.pi / 5.6
  self.ena2 = self.B_bc * 360 / self.pi / 5.6
  self.ena3 = self.C_cd * 360 / self.pi / 5.6
  self.en1 = self.D_ab * 360 / self.pi / 5.6
  self.en2 = self.D_bc * 360 / self.pi / 5.6
  self.en3 = self.D_cd * 360 / self.pi / 5.6
  brick.marker().down("blue")

  brick.encoder("E1").reset()
  brick.encoder("E2").reset()
  brick.encoder("E3").reset()
  brick.encoder("E4").reset()

  brick.motor("M3").setPower(100)

  brick.motor("M4").setPower(-(100))

  while not (brick.encoder("E3").read() > int(self.ena1)):
    script.wait(10)

  brick.encoder("E1").reset()
  brick.encoder("E2").reset()
  brick.encoder("E3").reset()
  brick.encoder("E4").reset()

  brick.motor("M3").setPower(100)
  brick.motor("M4").setPower(100)

  while not (brick.encoder("E3").read() > int(self.en1 * 30)):
    script.wait(10)

  brick.encoder("E1").reset()
  brick.encoder("E2").reset()
  brick.encoder("E3").reset()
```

```
    brick.encoder("E4").reset()

    brick.motor("M3").setPower(100)

    brick.motor("M4").setPower(-(100))

    while not (brick.encoder("E3").read() > int(self.ena2)):
      script.wait(10)

    brick.encoder("E1").reset()
    brick.encoder("E2").reset()
    brick.encoder("E3").reset()
    brick.encoder("E4").reset()

    brick.motor("M3").setPower(100)
    brick.motor("M4").setPower(100)

    while not (brick.encoder("E3").read() > int(self.en2 * 30)):
      script.wait(10)

    brick.encoder("E1").reset()
    brick.encoder("E2").reset()
    brick.encoder("E3").reset()
    brick.encoder("E4").reset()

    brick.motor("M3").setPower(100)

    brick.motor("M4").setPower(-(100))

    while not (brick.encoder("E3").read() > int(self.ena3)):
      script.wait(10)

    brick.encoder("E1").reset()
    brick.encoder("E2").reset()
    brick.encoder("E3").reset()
    brick.encoder("E4").reset()

    brick.motor("M3").setPower(100)
    brick.motor("M4").setPower(100)

    while not (brick.encoder("E3").read() > int(self.en3 * 25)):
      script.wait(10)

    brick.motor("M1").powerOff()
    brick.motor("M2").powerOff()
    brick.motor("M3").powerOff()
    brick.motor("M4").powerOff()

    brick.display().addLabel(str(self.S), 1, 1, 45)
    brick.display().redraw()

    brick.display().addLabel(str(self.V), 1, 45, 45)
    brick.display().redraw()

    script.wait(30000)

    brick.stop()
    return

def main():
  program = Program()
  program.execMain()

if __name__ == '__main__':
  main()
```

Задание № 4.1. Тестирование в среде ТРИК





Задание № 4.2. Тестирование на роботе

Робот двиггается по траектории из п. 1. На дисплее вводится две строки: первая - длинна всего маршрута в метрах, вторая - скорость робота в м/с. На диспле выводится:

5.93
0.17

Кейс 2. Вариант 12.

Условие

В среде программирования Trick Studio в программе на языке Python задаются данные для перемещения мобильного трехколесного робота (или блоками). Программа должна по полученным данным вычислить среднюю скорость перемещения робота за заданное время при равномерном перемещении робота по траектории из четырех точек. Необходимо написать программу, в которую включить заданные точки маршрута для того, чтобы программа рассчитала и вывела значения длинны маршрута и средней скорости робота. Задание реализуется либо в Trick Studio, либо на реальном роботе.

Формат ввода

При написании программы необходимо учесть следующие заданные координаты точек маршрута:

x0,y0 – координаты нулевой точки (A) маршрута, м.;
x1,y1 – координаты первой точки (B) маршрута, м.;
x2,y2 – координаты второй точки (C) маршрута, м.;
x3,y3 – координаты третьей точки (D) маршрута, м.;
Робот движется равномерно по траектории. Параметры координат точек маршрута и времени его прохождения задаются в основном коде программы в виде не целых чисел. В каждой строке с координатами два числа, первое x, второе y. Числа в строке разделены одинарным пробелом. В пятой строке указывается одно число – время. Размерность координат – метры, времени – секунды и миллисекунды (ограничение в два знака после точки). Границы координат не выходят за пределы от 0 до 5 метров по x и y.
Пример задания входных данных (x0,y0), (x1,y1), (x2,y2), (x3,y3), t для программы:
2.61 2.1
4.0 3.17
0.74 0.58
1.97 2.55
11.0
Участники самостоятельно разрабатывают программный код, содержащий разделы считывания заданных координат, промежуточных расчетов и вывода данных на дисплей робота вычисленных расстояния между точками и скорости в м/с. Необходимо использовать следующие формулы для расчета параметров скорости и расстояния между всеми точками маршрута:
$V = \frac{L}{t}$, где V - скорость (м/с), L - длинна пути маршрута (м), t - время (сек),
|AB|= $\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$, где √ - корень квадратный, |AB| - модуль расстояния между соседними точками A и B с координатами (x1,y1), (x2,y2) соответственно. Остальные расстояния и формулы для участков BC, CD и DE задаются самостоятельно.
Формулы необходимо самостоятельно преобразовать к программному виду и включить в программу. Обозначения параметров функций и переменных участники могут выбирать самостоятельно.

Формат вывода

Необходимо чтобы робот проехал по заданным точкам в среде симуляции (или в реальности). Необходимо приложить к ответу видеозапись симуляции перемещения робота из Trick Studio. Код предоставлять в трех форматах: 1) скриншотами в отчете; 2) блоками в Trick Studio сохранен проект с работающей симуляцией; 3) в среде Trick Studio, но на языке Python.
Программа должна вывести две строки. В первой строке должно вывестись вычисленная программой длинна всего маршрута в виде вещественного числа с двумя знаками после запятой, единица измерения метры. Во второй строке должно вывестись вычисленная программой скорость робота в виде вещественного числа с двумя знаками после запятой, единица измерения метры в секунду (м/с). Необходимо предусмотреть только вывод чисел. В качестве разделителя целой и вещественной части – точка. Текст комментариев и единицы измерения выводить не нужно.
Пример того, что должна вывести программа (числа не связанны с данными, указанными в разделе "Формат ввода"):
9.44
3.03

Исходные точки траектории движения.

Задание входных данных (x0,y0), (x1,y1), (x2,y2), (x3,y3), t для программы:
2.61 2.1
4.0 3.17
0.74 0.58
1.97 2.55
11.0

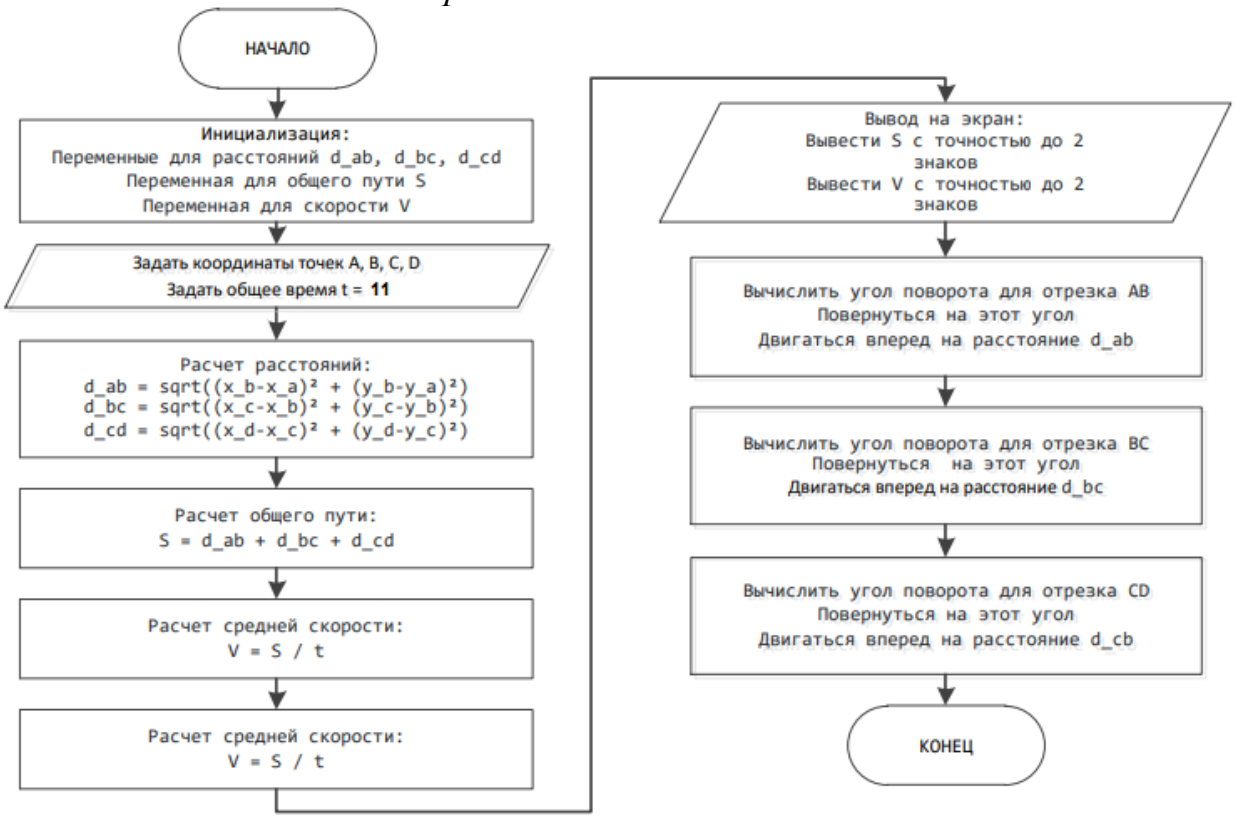
Задание № 1. Вычисление параметров движения

Расчтеные данные движения:

Участки пути.

td1=atan((3.17-2.1)/(4.0-2.61))
td2=atan((3.17-0.58)/(4.0-0.74))
td3=atan((2.55-0.58)/(1.97-0.74))
D_ab=sqrt((4.0-2.61)^2+(3.17-2.1)^2)
D_bc=sqrt((0.74-4.0)^2+(0.58-3.17)^2)
D_cd=sqrt((1.97-0.74)^2+(2.55-0.58)^2)
S=D_ab+D_bc+D_cd
V=S/11
A_ab=15.4/2*(0.4*pi-td1)
B_bc=15.4/2*(td1+pi-td2)
C_cd=15.4/2*(td2+1.25*pi-td3)
ena1=A_ab*360/pi/5.6
ena2=B_bc*360/pi/5.6
ena3=C_cd*360/pi/5.6
en1=D_ab*360/pi/5.6
en2=D_bc*360/pi/5.6
en3=D_cd*360/pi/5.6

Задание № 2. Блок-схема алгоритма



Задание № 3.1. Управляющая программа (в базовом варианте – управление движением по времени)

Задание № 3.2. Управляющая программа (в усложненном варианте – управление движением по показаниям энкодеров)

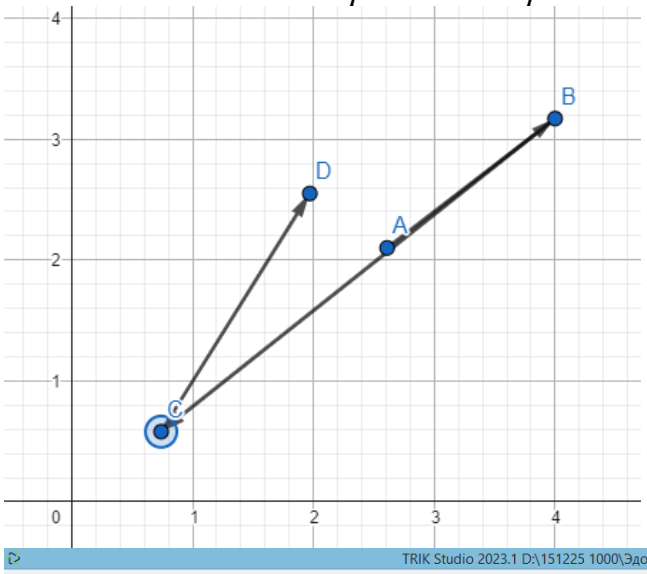
```
import sys
import time
import random
import math

class Program():
  __interpretation_started_timestamp__ = time.time() * 1000
```

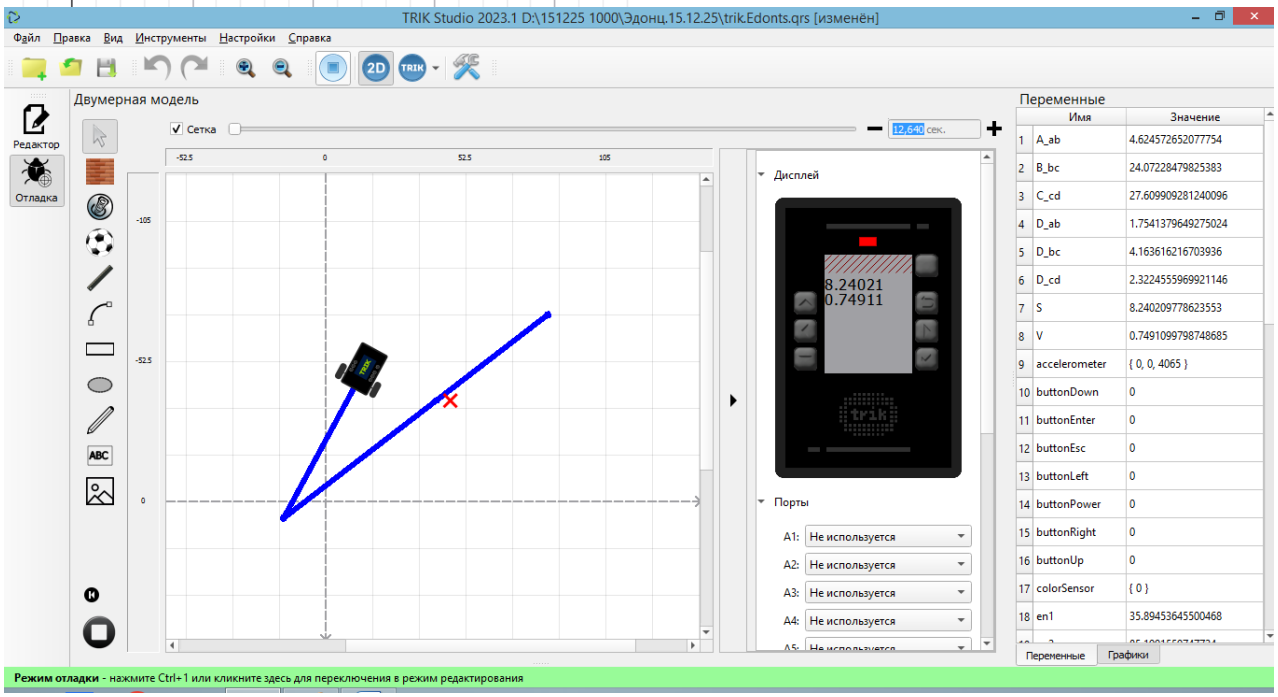
```
pi = 3.141592653589793
A_ab = None
B_bc = None
C_cd = None
D_ab = None
D_bc = None
D_cd = None
S = None
V = None
en1 = None
en2 = None
en3 = None
ena1 = None
ena2 = None
ena3 = None
td1 = None
td2 = None
td3 = None

def execMain(self):


  self.td1 = math.atan((3.17 - 2.1) / (4.0 - 2.61))
  self.td2 = math.atan((3.17 - 0.58) / (4.0 - 0.74))
  self.td3 = math.atan((2.55 - 0.58) / (1.97 - 0.74))
  self.D_ab = math.sqrt(math.pow((4.0 - 2.61), 2) + math.pow((3.17 - 2.1), 2))
  self.D_bc = math.sqrt(math.pow((0.74 - 4.0), 2) + math.pow((0.58 - 3.17), 2))
  self.D_cd = math.sqrt(math.pow((1.97 - 0.74), 2) + math.pow((2.55 - 0.58), 2))
  self.S = self.D_ab + self.D_bc + self.D_cd
  self.V = self.S / 11
  self.A_ab = 15.4 / 2 * (0.4 * self.pi - self.td1)
  self.B_bc = 15.4 / 2 * (self.td1 + self.pi - self.td2)
  self.C_cd = 15.4 / 2 * (self.td2 + 1.25 * self.pi - self.td3)
  self.ena1 = self.A_ab * 360 / self.pi / 5.6
  self.ena2 = self.B_bc * 360 / self.pi / 5.6
  self.ena3 = self.C_cd * 360 / self.pi / 5.6
  self.en1 = self.D_ab * 360 / self.pi / 5.6
  self.en2 = self.D_bc * 360 / self.pi / 5.6
  self.en3 = self.D_cd * 360 / self.pi / 5.6
  brick.marker().down("blue")

  brick.encoder("E1").reset()
  brick.encoder("E2").reset()
  brick.encoder("E3").reset()
  brick.encoder("E4").reset()

  brick.motor("M3").setPower(100)

  brick.motor("M4").setPower(-(100))

  while not (brick.encoder("E3").read() > int(self.ena1)):
    script.wait(10)

  brick.encoder("E1").reset()
  brick.encoder("E2").reset()
  brick.encoder("E3").reset()
  brick.encoder("E4").reset()

  brick.motor("M3").setPower(100)
  brick.motor("M4").setPower(100)

  while not (brick.encoder("E3").read() > int(self.en1 * 30)):
    script.wait(10)

  brick.encoder("E1").reset()
```

```
    brick.encoder("E2").reset()
    brick.encoder("E3").reset()
    brick.encoder("E4").reset()

    brick.motor("M3").setPower(100)

    brick.motor("M4").setPower(-(100))

    while not (brick.encoder("E3").read() > int(self.ena2)):
      script.wait(10)

    brick.encoder("E1").reset()
    brick.encoder("E2").reset()
    brick.encoder("E3").reset()
    brick.encoder("E4").reset()

    brick.motor("M3").setPower(100)
    brick.motor("M4").setPower(100)

    while not (brick.encoder("E3").read() > int(self.en2 * 30)):
      script.wait(10)

    brick.encoder("E1").reset()
    brick.encoder("E2").reset()
    brick.encoder("E3").reset()
    brick.encoder("E4").reset()

    brick.motor("M3").setPower(100)

    brick.motor("M4").setPower(-(100))

    while not (brick.encoder("E3").read() > int(self.ena3)):
      script.wait(10)

    brick.encoder("E1").reset()
    brick.encoder("E2").reset()
    brick.encoder("E3").reset()
    brick.encoder("E4").reset()

    brick.motor("M3").setPower(100)
    brick.motor("M4").setPower(100)

    while not (brick.encoder("E3").read() > int(self.en3 * 25)):
      script.wait(10)

    brick.motor("M1").powerOff()
    brick.motor("M2").powerOff()
    brick.motor("M3").powerOff()
    brick.motor("M4").powerOff()

    brick.display().addLabel(str(self.S), 1, 1, 45)
    brick.display().redraw()

    brick.display().addLabel(str(self.V), 1, 45, 45)
    brick.display().redraw()

    script.wait(30000)

    brick.stop()
    return

def main():
  program = Program()
  program.execMain()

if __name__ == '__main__':
```

main()

Задание № 4.1. Тестирование в среде ТРИК





Задание № 4.2. Тестирование на роботе

Робот двиггается по траектории из п. 1. На дисплее вводится две строки: первая - длинна всего маршрута в метрах, вторая - скорость робота в м/с. На диспле выводится:

8.23
0.75

Кейс 2. Вариант 13.

Условие

В среде программирования Trick Studio в программе на языке Python задаются данные для перемещения мобильного трехколесного робота (или блоками). Программа должна по полученным данным вычислить среднюю скорость перемещения робота за заданное время при равномерном перемещении робота по траектории из четырех точек. Необходимо написать программу, в которую включить заданные точки маршрута для того, чтобы программа рассчитала и вывела значения длинны маршрута и средней скорости робота. Задание реализуется либо в Trick Studio, либо на реальном роботе.

Формат ввода

При написании программы необходимо учесть следующие заданные координаты точек маршрута:
x0,y0 – координаты нулевой точки (A) маршрута, м.;
x1,y1 – координаты первой точки (B) маршрута, м.;
x2,y2 – координаты второй точки (C) маршрута, м.;
x3,y3 – координаты третьей точки (D) маршрута, м.;
Робот движется равномерно по траектории. Параметры координат точек маршрута и времени его прохождения задаются в основном коде программы в виде не целых чисел. В каждой строке с координатами два числа, первое x, второе y. Числа в строке разделены одинарным пробелом. В пятой строке указывается одно число – время. Размерность координат – метры, времени – секунды и миллисекунды (ограничение в два знака после точки). Границы координат не выходят за пределы от 0 до 5 метров по x и y.
Пример задания входных данных (x0,y0), (x1,y1), (x2,y2), (x3,y3), t для программы:
4.6 4.18
3.24 1.11
1.1 3.44
4.04 1.23
39.0
Участники самостоятельно разрабатывают программный код, содержащий разделы считывания данных из файла input.txt, промежуточных расчетов и вывода на дисплей робота вычисленных расстояния между точками и скорости в м/с. Необходимо использовать следующие формулы для расчета параметров скорости и расстояния между всеми точками маршрута:
$V = \frac{L}{t}$, где V - скорость (м/с), L - длинна пути маршрута (м), t - время (сек),
|AB|= $\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$, где √ - корень квадратный, |AB| - модуль расстояния между соседними точками A и B с координатами (x1,y1), (x2,y2) соответственно. Остальные расстояния и формулы для участков BC, CD и DE задаются самостоятельно.
Формулы необходимо самостоятельно преобразовать к программному виду и включить в программу. Обозначения параметров функций и переменных участники могут выбирать самостоятельно.

Формат вывода

Необходимо чтобы робот проехал по заданным точкам в среде симуляции (или в реальности). Необходимо приложить к ответу видеозапись симулиции перемещения робота из Trick Studio. Код предоставлять в трех форматах: 1) скриншотами в отчете; 2) блоками в Trick Studio сохранен проект с работающей симуляцией; 3) в среде Trick Studio, но на языке Python.
Программа должна вывести две строки. В первой строке должно вывестись вычисленная программой длинна всего маршрута в виде вещественного числа с двумя знаками после запятой, единица измерения метры. Во второй строке должно вывестись вычисленная программой скорость робота в виде вещественного числа с двумя знаками после запятой, единица измерения метры в секунду (м/с). Необходимо предусмотреть только вывод чисел. В качестве разделителя целой и вещественной части – точка. Текст комментариев и единицы измерения выводить не нужно.
Пример того, что должна вывести программа (числа не связанны с данными указанными в разделе "Формат ввода"):
9.44
3.03

Исходные точки траектории движения.

Задание входных данных (x0,y0), (x1,y1), (x2,y2), (x3,y3), t для программы:
4.6 4.18
3.24 1.11
1.1 3.44
4.04 1.23

39.0

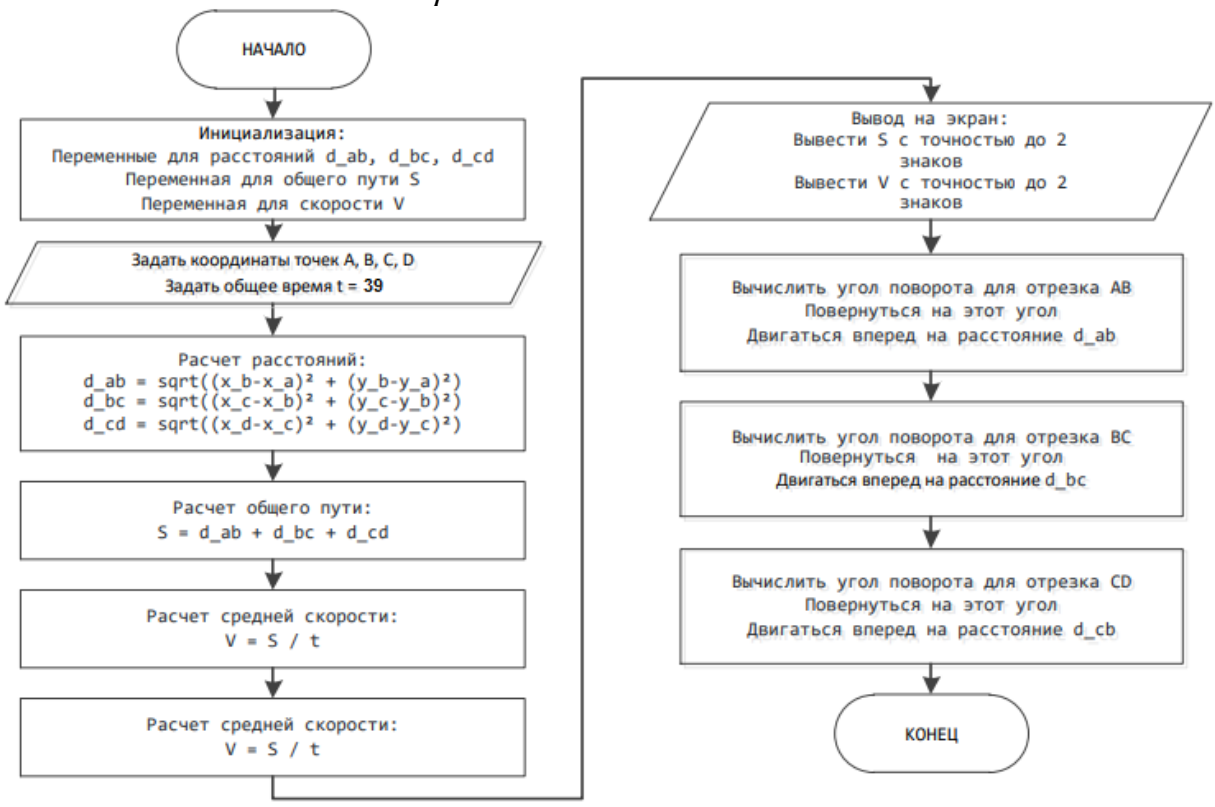
Задание № 1. Вычисление параметров движения

Расчтеные данные движения:

Участки пути.

td1=atan((4.18-1.11)/(4.6-3.24))
td2=atan((3.44-1.11)/(3.24-1.1))
td3=atan((3.44-1.23)/(4.04-1.1))
D_ab=sqrt((3.24-4.6)^2+(1.11-4.18)^2)
D_bc=sqrt((1.1-3.24)^2+(3.44-1.11)^2)
D_cd=sqrt((4.04-1.1)^2+(1.23-3.44)^2)
S=D_ab+D_bc+D_cd
V=S/39
A_ab=15.4/2*(1.7*pi-td1)
B_bc=15.4/2*(td1+1.25*pi-td2)
C_cd=15.4/2*(td2+pi-td3)
ena1=A_ab*360/pi/5.6
ena2=B_bc*360/pi/5.6
ena3=C_cd*360/pi/5.6
en1=D_ab*360/pi/5.6
en2=D_bc*360/pi/5.6
en3=D_cd*360/pi/5.6

Задание № 2. Блок-схема алгоритма



Задание № 3.1. Управляющая программа (в базовом варианте – управление движением по времени)

Задание № 3.2. Управляющая программа (в усложненном варианте – управление движением по показаниям энкодеров)

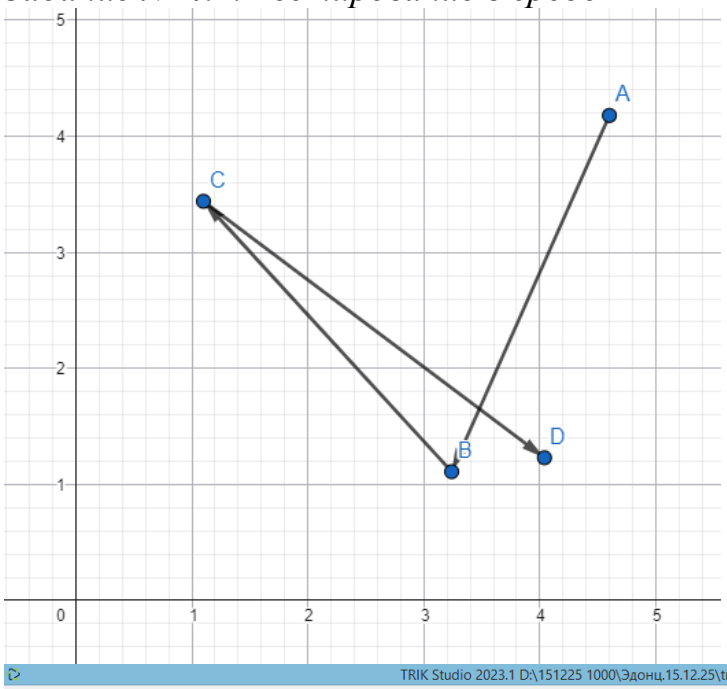
```
import sys
import time
import random
import math

class Program():
```

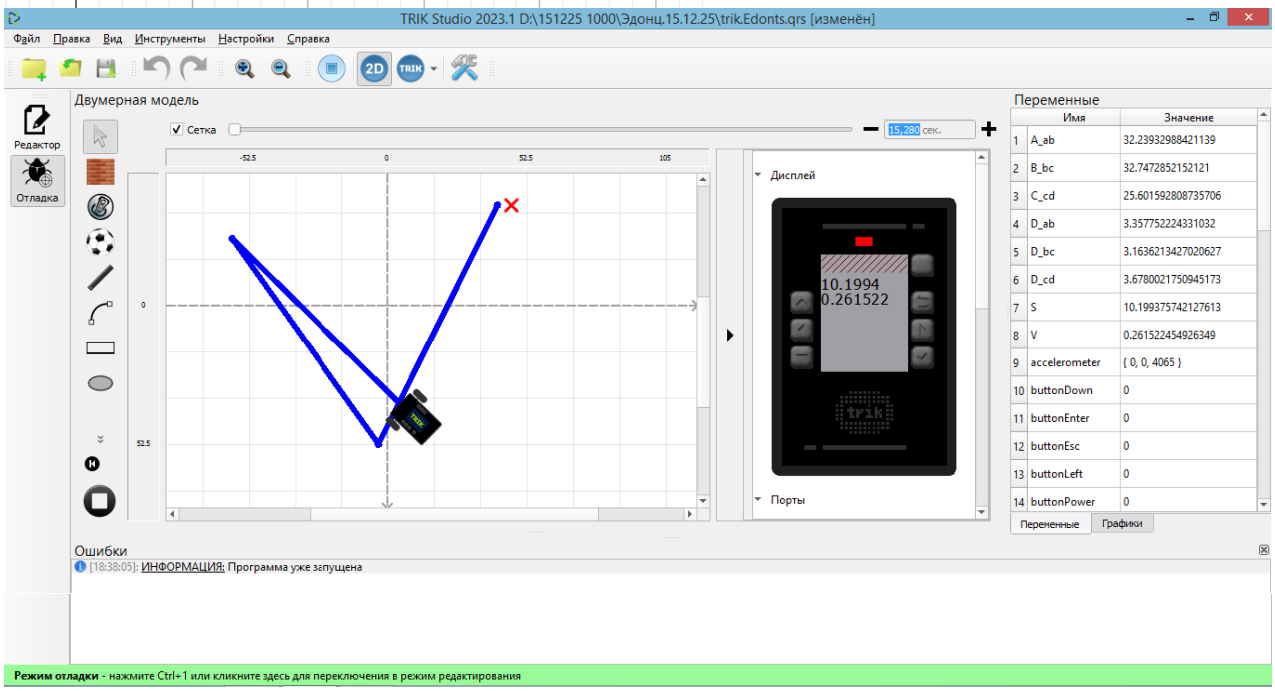
```
__interpretation_started_timestamp__ = time.time() * 1000

pi = 3.141592653589793
A_ab = None
B_bc = None
C_cd = None
D_ab = None
D_bc = None
D_cd = None
S = None
V = None
en1 = None
en2 = None
en3 = None
ena1 = None
ena2 = None
ena3 = None
td1 = None
td2 = None
td3 = None

def execMain(self):


  self.td1 = math.atan((4.18 - 1.11) / (4.6 - 3.24))
  self.td2 = math.atan((3.44 - 1.11) / (3.24 - 1.1))
  self.td3 = math.atan((3.44 - 1.23) / (4.04 - 1.1))
  self.D_ab = math.sqrt(math.pow((3.24 - 4.6), 2) + math.pow((1.11 - 4.18), 2))
  self.D_bc = math.sqrt(math.pow((1.1 - 3.24), 2) + math.pow((3.44 - 1.11), 2))
  self.D_cd = math.sqrt(math.pow((4.04 - 1.1), 2) + math.pow((1.23 - 3.44), 2))
  self.S = self.D_ab + self.D_bc + self.D_cd
  self.V = self.S / 39
  self.A_ab = 15.4 / 2 * (1.7 * self.pi - self.td1)
  self.B_bc = 15.4 / 2 * (self.td1 + 1.25 * self.pi - self.td2)
  self.C_cd = 15.4 / 2 * (self.td2 + self.pi - self.td3)
  self.ena1 = self.A_ab * 360 / self.pi / 5.6
  self.ena2 = self.B_bc * 360 / self.pi / 5.6
  self.ena3 = self.C_cd * 360 / self.pi / 5.6
  self.en1 = self.D_ab * 360 / self.pi / 5.6
  self.en2 = self.D_bc * 360 / self.pi / 5.6
  self.en3 = self.D_cd * 360 / self.pi / 5.6
  brick.marker().down("blue")

  brick.encoder("E1").reset()
  brick.encoder("E2").reset()
  brick.encoder("E3").reset()
  brick.encoder("E4").reset()

  brick.motor("M3").setPower(100)

  brick.motor("M4").setPower(-(100))

  while not (brick.encoder("E3").read() > int(self.ena1)):
    script.wait(10)

  brick.encoder("E1").reset()
  brick.encoder("E2").reset()
  brick.encoder("E3").reset()
  brick.encoder("E4").reset()

  brick.motor("M3").setPower(100)
  brick.motor("M4").setPower(100)

  while not (brick.encoder("E3").read() > int(self.en1 * 30)):
    script.wait(10)
```

```
    brick.encoder("E1").reset()
    brick.encoder("E2").reset()
    brick.encoder("E3").reset()
    brick.encoder("E4").reset()

    brick.motor("M3").setPower(100)

    brick.motor("M4").setPower(-(100))

    while not (brick.encoder("E3").read() > int(self.ena2)):
      script.wait(10)

    brick.encoder("E1").reset()
    brick.encoder("E2").reset()
    brick.encoder("E3").reset()
    brick.encoder("E4").reset()

    brick.motor("M3").setPower(100)
    brick.motor("M4").setPower(100)

    while not (brick.encoder("E3").read() > int(self.en2 * 30)):
      script.wait(10)

    brick.encoder("E1").reset()
    brick.encoder("E2").reset()
    brick.encoder("E3").reset()
    brick.encoder("E4").reset()

    brick.motor("M3").setPower(100)

    brick.motor("M4").setPower(-(100))

    while not (brick.encoder("E3").read() > int(self.ena3)):
      script.wait(10)

    brick.encoder("E1").reset()
    brick.encoder("E2").reset()
    brick.encoder("E3").reset()
    brick.encoder("E4").reset()

    brick.motor("M3").setPower(100)
    brick.motor("M4").setPower(100)

    while not (brick.encoder("E3").read() > int(self.en3 * 25)):
      script.wait(10)

    brick.motor("M1").powerOff()
    brick.motor("M2").powerOff()
    brick.motor("M3").powerOff()
    brick.motor("M4").powerOff()

    brick.display().addLabel(str(self.S), 1, 1, 45)
    brick.display().redraw()

    brick.display().addLabel(str(self.V), 1, 45, 45)
    brick.display().redraw()

    script.wait(30000)

    brick.stop()
    return

def main():
  program = Program()
  program.execMain()
```

```
if __name__ == '__main__':
  main()
```

Задание № 4.1. Тестирование в среде ТРИК





Задание № 4.2. Тестирование на роботе

Робот двиггается по траектории из п. 1. На дисплее вводится две строки: первая - длинна всего маршрута в метрах, вторая - скорость робота в м/с. На диспле выводится:

10.2
0.26

Кейс 2. Вариант 14.

Условие

В среде программирования Trick Studio в программе на языке Python задаются данные для перемещения мобильного трехколесного робота (или блоками). Программа должна по полученным данным вычислить среднюю скорость перемещения робота за заданное время при равномерном перемещении робота по траектории из четырех точек. Необходимо написать программу, в которую включить заданные точки маршрута для того, чтобы

программа рассчитала и вывела значения длинны маршрута и средней скорости робота. Задание реализуется либо в Trick Studio, либо на реальном роботе.

Формат ввода

При написании программы необходимо учесть следующие заданные координаты точек маршрута:

x0,y0 – координаты нулевой точки (A) маршрута, м.;
x1,y1 – координаты первой точки (B) маршрута, м.;
x2,y2 – координаты второй точки (C) маршрута, м.;
x3,y3 – координаты третьей точки (D) маршрута, м.;

Робот движется равномерно по траектории. Параметры координат точек маршрута и времени его прохождения задаются в основном коде программы в виде не целых чисел. В каждой строке с координатами два числа, первое x, второе y. Числа в строке разделены одинарным пробелом. В пятой строке указывается одно число – время. Размерность координат – метры, времени – секунды и миллисекунды (ограничение в два знака после точки). Границы координат не выходят за пределы от 0 до 5 метров по x и y.

Пример задания входных данных (x0,y0), (x1,y1), (x2,y2), (x3,y3), t для программы:

3.24 0.45
4.81 2.2
2.12 3.69
3.56 3.08
40.0

Участники самостоятельно разрабатывают программный код, содержащий разделы считывания заданных координат, промежуточных расчетов и вывода данных на дисплей робота вычисленных расстояния между точками и скорости в м/с. Необходимо использовать следующие формулы для расчета параметров скорости и расстояния между всеми точками маршрута:

$V = \frac{L}{t}$, где V - скорость (м/с), L - длинна пути маршрута (м), t - время (сек),

|AB|= $\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$, где √ - корень квадратный, |AB| - модуль расстояния между соседними точками A и B с координатами (x1,y1), (x2,y2) соответственно. Остальные расстояния и формулы для участков BC, CD и DE задаются самостоятельно.

Формулы необходимо самостоятельно преобразовать к программному виду и включить в программу. Обозначения параметров функций и переменных участники могут выбирать самостоятельно.

Формат вывода

Необходимо чтобы робот проехал по заданным точкам в среде симуляции (или в реальности). Необходимо приложить к ответу видеозапись симуляции перемещения робота из Trick Studio. Код предоставлять в трех форматах: 1) скриншотами в отчете; 2) блоками в Trick Studio сохранен проект с работающей симуляцией; 3) в среде Trick Studio, но на языке Python.

Программа должна вывести две строки. В первой строке должно вывестись вычисленная программой длинна всего маршрута в виде вещественного числа с двумя знаками после запятой, единица измерения метры. Во второй строке должно вывестись вычисленная программой скорость робота в виде вещественного числа с двумя знаками после запятой, единица измерения метры в секунду (м/с). Необходимо предусмотреть только вывод чисел. В качестве разделителя целой и вещественной части – точка. Текст комментариев и единицы измерения выводить не нужно.

Пример того, что должна вывести программа (числа не связанны с данными, указанными в разделе "Формат ввода"):

9.44
3.03

Исходные точки траектории движения.

Задание входных данных (x0,y0), (x1,y1), (x2,y2), (x3,y3), t для программы:

3.24 0.45

4.81 2.2
2.12 3.69
3.56 3.08
40.0

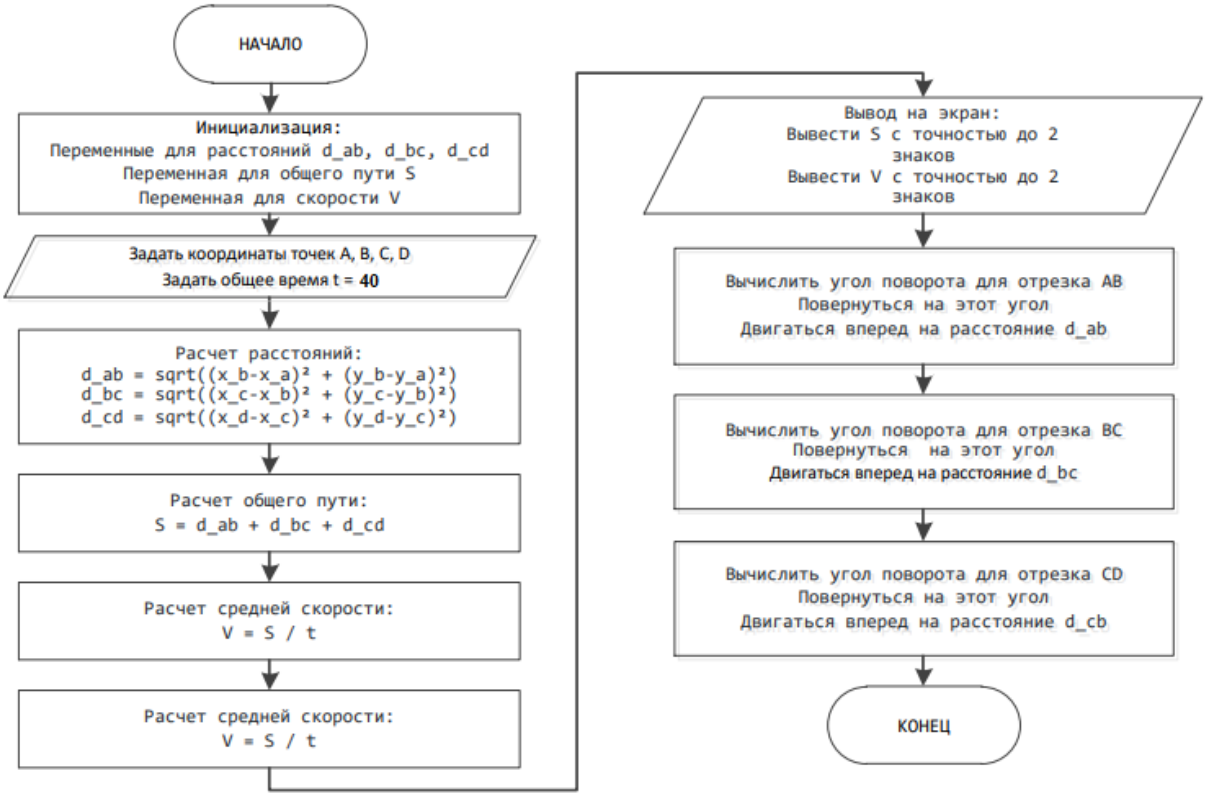
Задание № 1. Вычисление параметров движения

Расчтеные данные движения:

Участки пути.

td1=atan((2.2 - 0.45) / (4.81 - 3.24))
td2=atan((3.69 - 2.2) / (4.81 - 2.12))
td3=atan((3.69 - 3.08) / (3.56 - 2.12))
D_ab=sqrt((4.81 - 3.24)^2+(2.2 - 0.45)^2)
D_bc=sqrt((2.12 - 4.81)^2+(3.69 - 2.2)^2)
D_cd=sqrt((3.56 - 2.12)^2+(3.08 - 3.69)^2)
S=D_ab+D_bc+D_cd
V=S/40
A_ab=15.4/2*(0.45*pi-td1)
B_bc=15.4/2*(td1+0.45*pi-td2)
C_cd=15.4/2*(td2+pi-td3)
ena1=A_ab*360/pi/5.6
ena2=B_bc*360/pi/5.6
ena3=C_cd*360/pi/5.6
en1=D_ab*360/pi/5.6
en2=D_bc*360/pi/5.6
en3=D_cd*360/pi/5.6

Задание № 2. Блок-схема алгоритма



Задание № 3.1. Управляющая программа (в базовом варианте – управление движением по времени)

Задание № 3.2. Управляющая программа (в усложненном варианте – управление движением по показаниям энкодеров)

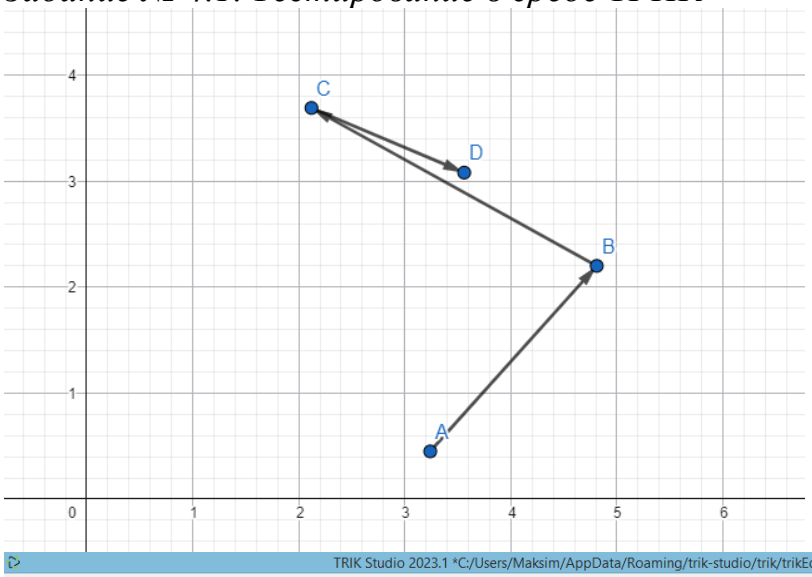
import sys

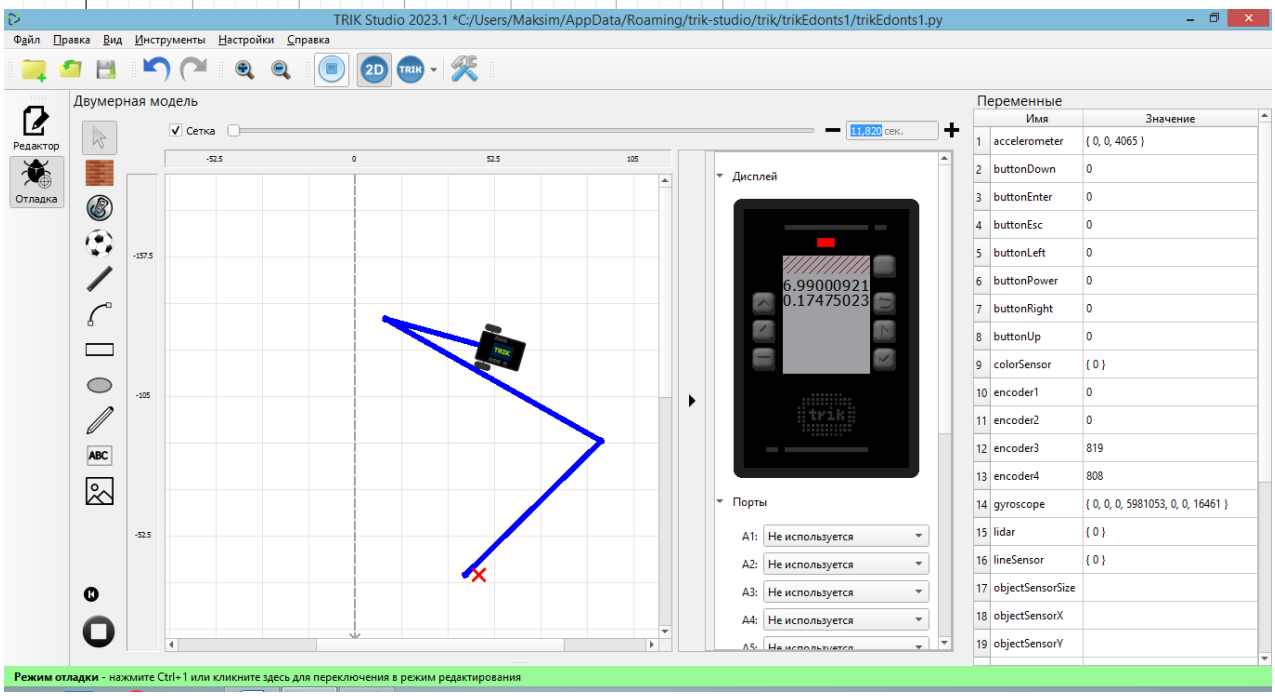
```
import time
import random
import math

class Program():
  __interpretation_started_timestamp__ = time.time() * 1000

  pi = 3.141592653589793
  A_ab = None
  B_bc = None
  C_cd = None
  D_ab = None
  D_bc = None
  D_cd = None
  S = None
  V = None
  en1 = None
  en2 = None
  en3 = None
  ena1 = None
  ena2 = None
  ena3 = None
  td1 = None
  td2 = None
  td3 = None

  def execMain(self):


    self.td1 = math.atan((2.2 - 0.45) / (4.81 - 3.24))
    self.td2 = math.atan((3.69 - 2.2) / (4.81 - 2.12))
    self.td3 = math.atan((3.69 - 3.08) / (3.56 - 2.12))
    self.D_ab = math.sqrt(math.pow((4.81 - 3.24), 2) + math.pow((2.2 - 0.45), 2))
    self.D_bc = math.sqrt(math.pow((2.12 - 4.81), 2) + math.pow((3.69 - 2.2), 2))
    self.D_cd = math.sqrt(math.pow((3.56 - 2.12), 2) + math.pow((3.08 - 3.69), 2))
    self.S = self.D_ab + self.D_bc + self.D_cd
    self.V = self.S / 40
    self.A_ab = 15.4 / 2 * (0.45 * self.pi - self.td1)
    self.B_bc = 15.4 / 2 * (self.td1 + 0.45*self.pi - self.td2)
    self.C_cd = 15.4 / 2 * (self.td2 + self.pi - self.td3)
    self.ena1 = self.A_ab * 360 / self.pi / 5.6
    self.ena2 = self.B_bc * 360 / self.pi / 5.6
    self.ena3 = self.C_cd * 360 / self.pi / 5.6
    self.en1 = self.D_ab * 360 / self.pi / 5.6
    self.en2 = self.D_bc * 360 / self.pi / 5.6
    self.en3 = self.D_cd * 360 / self.pi / 5.6
    brick.marker().down("blue")

    brick.encoder("E1").reset()
    brick.encoder("E2").reset()
    brick.encoder("E3").reset()
    brick.encoder("E4").reset()

    brick.motor("M3").setPower(100)

    brick.motor("M4").setPower(-(100))

    while not (brick.encoder("E3").read() > int(self.ena1)):
      script.wait(10)

    brick.encoder("E1").reset()
    brick.encoder("E2").reset()
    brick.encoder("E3").reset()
    brick.encoder("E4").reset()

    brick.motor("M3").setPower(100)
```

```
brick.motor("M4").setPower(100)

while not (brick.encoder("E3").read() > int(self.en1 * 30)):
  script.wait(10)

brick.encoder("E1").reset()
brick.encoder("E2").reset()
brick.encoder("E3").reset()
brick.encoder("E4").reset()

brick.motor("M3").setPower(100)

brick.motor("M4").setPower(-(100))

while not (brick.encoder("E3").read() > int(self.ena2)):
  script.wait(10)

brick.encoder("E1").reset()
brick.encoder("E2").reset()
brick.encoder("E3").reset()
brick.encoder("E4").reset()

brick.motor("M3").setPower(100)
brick.motor("M4").setPower(100)

while not (brick.encoder("E3").read() > int(self.en2 * 30)):
  script.wait(10)

brick.encoder("E1").reset()
brick.encoder("E2").reset()
brick.encoder("E3").reset()
brick.encoder("E4").reset()

brick.motor("M3").setPower(100)

brick.motor("M4").setPower(-(100))

while not (brick.encoder("E3").read() > int(self.ena3)):
  script.wait(10)

brick.encoder("E1").reset()
brick.encoder("E2").reset()
brick.encoder("E3").reset()
brick.encoder("E4").reset()

brick.motor("M3").setPower(100)
brick.motor("M4").setPower(100)

while not (brick.encoder("E3").read() > int(self.en3 * 25)):
  script.wait(10)

brick.motor("M1").powerOff()
brick.motor("M2").powerOff()
brick.motor("M3").powerOff()
brick.motor("M4").powerOff()

brick.display().addLabel(str(self.S), 1, 1, 45)
brick.display().redraw()

brick.display().addLabel(str(self.V), 1, 45, 45)
brick.display().redraw()

script.wait(30000)

brick.stop()
return
```

```
def main():
  program = Program()
  program.execMain()

if __name__ == '__main__':
  main()
```

Задание № 4.1. Тестирование в среде ТРИК





Задание № 4.2. Тестирование на роботе

Робот двиггается по траектории из п. 1. На дисплее вводится две строки: первая - длинна всего маршрута в метрах, вторая - скорость робота в м/с. На диспле выводится:

6.99

0.17