Методические рекомендации по решению задач
практического этапа
Московского конкурса межпредметных навыков и умений
Интеллектуальный мегаполис. Потенциал
в номинации
ИТ класс
по направлению
Робототехника

Методические рекомендации к выполнению конкурсных заданий

Материалы практического этапа Московского конкурса меж предметных навыков и знаний «Интеллектуальный мегаполис. Потенциал» (далее – Конкурс) предназначены для оценки уровня практической подготовки участников Конкурса.

1. **Общая информация о знаниях и навыках**

Для выполнения задания уровня сложности «базовый» участнику необходимо обладать следующими знаниями и навыками.

Знания:

- Базовых принципов работы с инструментом и крепежом.
- Принципов составления и электрических схем и базовых электронных компонентов.
- Принципов составления и стандартов изображения алгоритмов.
- Синтаксиса языка С и С++, принципов программирования Arduino, основ управления движением робота.

Умения:

- Следовать инструкции, навыки использования базовых инструментов и крепежа.
- Читать блок-схемы, принципиальные электрические схемы, соединять электронные модули в соответствии со схемами.
- Разрабатывать и изображать алгоритмы в соответствии с заданием, используя специализированное программное обеспечение.
- Разрабатывать и отлаживать программы, обеспечивающие движение робота в соответствии с алгоритмом и заданием.
- Использовать датчики и программировать действие учебного робота в зависимости от задач проекта.

Для выполнения задания уровня сложности «повышенный» участнику необходимо знать и уметь:

- Составлять алгоритмы и реализовывать программы считывающие и обрабатывающие несколько видов датчиков.

- Уметь изображать алгоритмы повышенной сложности в соответствии с заданием, используя специализированное программное обеспечение.
- Знать синтаксис языка С и принципов программирования Arduino, на уровне достаточном для реализации одновременного управления движением робота и считывания двух видов датчиков.
- Знать принципы работы ИК и УЗ датчиков препятствий и особенности их подключения к плате Arduino.
- Владеть универсальными умениями: постановка задачи, формулирование проблемы; поиск, выделение и структурирование необходимой информации; выбор наиболее эффективных методов решения задачи в зависимости от конкретных условий.

2. Сборка робототехнического набора.

Даны детали для сборки робота из робототехнического набора и инструкция по сборке, электронные компоненты, схема подключения.

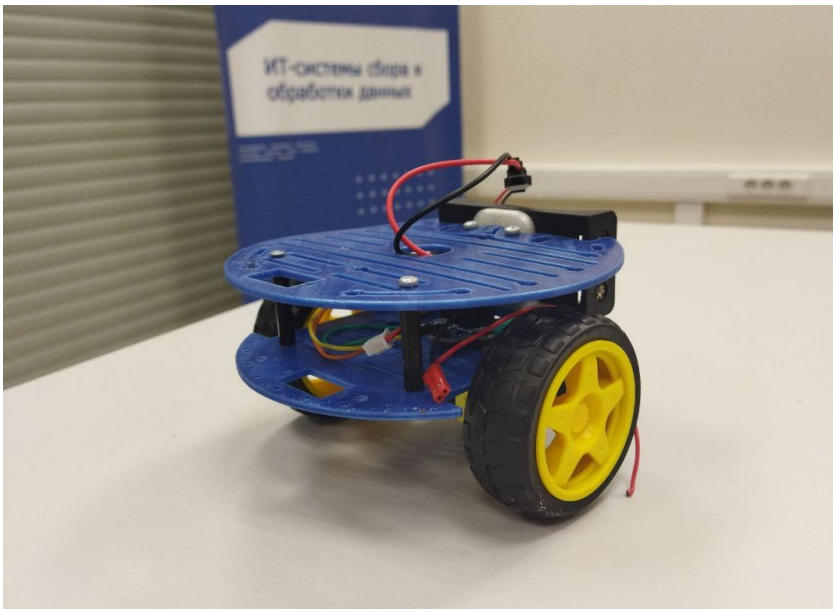

Рисунок 1: Пример робототехнического набора без отладочной платы и дополнительных модулей

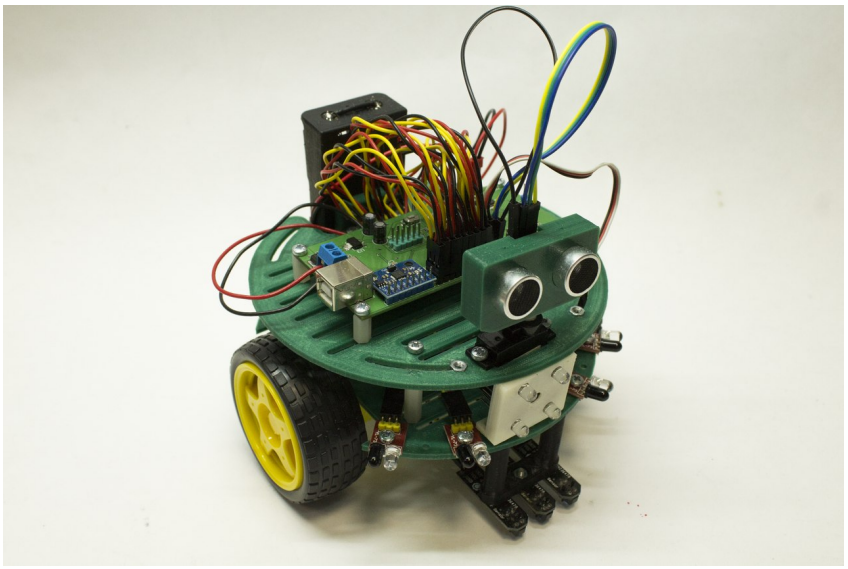

Рисунок 2: Пример собранного робототехнического набора с УЗ датчиком

Для того что бы собрать робота разработана пошаговая инструкция, описывающая последовательность шагов необходимых для сборки.

Дополнительно разработан анимационный ролик, упрощающий процесс сборки. Ролик доступен по ссылке: https://www.youtube.com/watch?v=8tdou6kaWII&t=2s

3. Подключение платы Arduino и программирование робототехнического набора.

Управление миниатюрным электродвигателем, возможно при подключении его к выходу Arduino, однако, дискретный выход не потянет двигатели, потребляющие больше 40 мА. Наиболее правильным будет управление любыми нагрузками мощнее светодиода при помощи транзисторов или специальных микросхем – драйверов. Это нужно для того, чтобы защитить ножку микроконтроллера от возможного повреждения из-за чрезмерно мощной нагрузки. Выход заключается в использовании простого усиливающего устройства, транзистора, чтобы иметь возможность управлять электродвигателями постоянного тока любой мощности. Наиболее удобный

вариант для работы с электродвигателями – использование специализированной микросхемы – драйвера состоящего из специальной сборки транзисторов. На картинке далее приведены варианты драйверов с микросхемами tb6612, l298 и l293D.

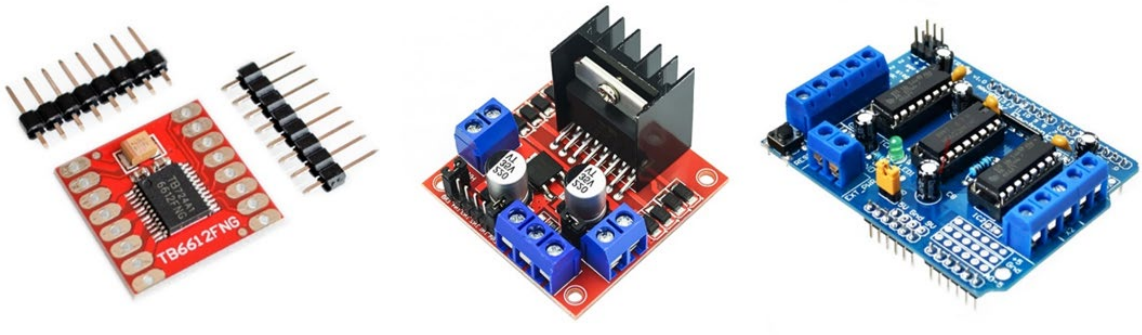

Рисунок 3: Пример собранного робототехнического набора с УЗ датчиком

Настройка и подключение драйвера

Моторы робота управляются контроллером не напрямую, а через микросхему-драйвер TB6612, для управления одним мотором необходимо 3 управляющих вывода микроконтроллера. (Для большинства микросхем – драйверов для управления мотором нужно 3 вывода микроконтроллера)

Проверьте подключение моторов! Левый мотор должен быть подключен к разъему M2, а правый - к M1.

Таблица 1. Соответствие ножек микроконтроллера на плате Roboard ножкам микросхемы TB6612

ШИМ мотора 1	PWMA	D5
ШИМ мотора 2	PWMB	D13
Мотор 1	AIN1	D12
Мотор 1	AIN2	D4
Мотор 2	BIN1	D8
Мотор 2	BIN2	D10

Примечание: Для различных вариантов робота или разных версий плат возможно свое сочетание ножек МК и выводов микросхемы драйвера. Для различных драйверов соблюдается логика – два вывода микроконтроллера для задания направления вращения и один вывод для регулирования скорости вращения при помощи ШИМ сигнала.

Первым делом необходимо заставить робота ездить по прямой, для этого нам понадобится открыть Arduino IDE, подключить плату RoBoard к компьютеру и написать следующий код:

```
//Для удобства создаем переменные, в которые записываем номера портов
#define LP 5 //Скорость левого мотора
#define L1 12 //Направление вращения левого мотора
#define L2 4 //Направление вращения левого мотора
#define RP 13 //Скорость вращения правого мотора
#define R1 8 //Направление вращения правого мотора
#define R2 10 //Направление вращения правого мотора
void setup() {
pinMode(LP,OUTPUT);
pinMode(L1,OUTPUT);
pinMode(L2,OUTPUT);
pinMode(RP,OUTPUT);
pinMode(R1,OUTPUT);
pinMode(R2,OUTPUT);
}
void loop() {
//Левое колесо вращается с ШИМом 100
analogWrite(LP,100);
digitalWrite(L1,LOW);
digitalWrite(L2,HIGH);
//Правое колесо вращается с ШИМом 100
analogWrite(RP,100);
digitalWrite(R1,LOW);
digitalWrite(R2,HIGH);
}
```

Листинг 1: Пример кода для движения вперед.

Вы можете заметить, что двигатели вращаются с разной скоростью и робот не может двигаться по прямой. Наша задача сделать замеры и заставить двигатели крутиться с одинаковой скоростью.

Напишем функцию движения:

```
void loop() {
  moveForward();

}

int speedRobot = 70; // Создаём переменную скорости

void moveForward(){ //Создаём функцию движения вперёд

  digitalWrite(L1, LOW);
  digitalWrite(L2, HIGH);
  digitalWrite(R1, LOW);
  digitalWrite(R2, HIGH);

  analogWrite(LP, speedRobot+5); // +5 Прибавляем скорости левому двигателю
                                 // чтобы правый и левый двигатели имели одинаковые скорости
  analogWrite(RP, speedRobot);  // Скорость правого двигателя
}
```

Листинг 2: Пример кода с коррекцией скорости.

Теперь изменим скорость моторов speedRobot и увидим, что моторы опять вращаются с разной скоростью. Зависимость сигнала ШИМ от скорости вращения моторов нелинейная и мы не можем точно предсказать, с какой скоростью вращается наш двигатель, поэтому мы сделаем замеры вручную, чтобы добиться максимально одинаковых скоростей вращения двигателей.

Изменим значение скорости speedRobot несколько раз и для каждого замера заставим робота двигаться прямо.

Таблица 2. Соотношение ШИМ сигнала для левого и правого мотора.

speedRobot	LP	RP
50	50 '+ 0	50
70	70 '+ 5	70
90	90 '+ 20	90
100	100 '+ 10	100
120	120 '+ 14	120

На графике видно, что для одинаковых скоростей моторов на них нужно подавать разный уровень ШИМ.

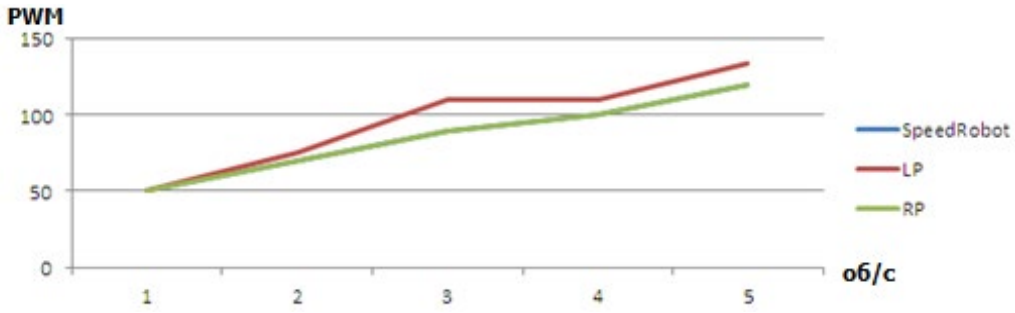

Рисунок 4. График соотношения скоростей вращения моторов.

Перепишем код с учётом выравнивания скоростей:

```
void moveForward(){ //Создаём функцию движения вперёд
  int delta = 0;
  digitalWrite(L1, LOW);
  digitalWrite(L2, HIGH);
  digitalWrite(R1, LOW);
  digitalWrite(R2, HIGH);

  if (speedRobot < 70){                    // Если скорость в пределах от 0 до 70
    delta = 0;
  } else if (70 <= speedRobot < 90 ){      // Если скорость в пределах от 70 до 90
    delta = 5;                             // то левый мотор крутиться на 5 ед ШИМ быстрее
  } else if (90 <= speedRobot < 100 ){     // Если скорость в пределах от 90 до 100
    delta = 20;
  } else if (100 <= speedRobot < 120 ){    // Если скорость в пределах от 100 до 120
    delta = 10;
  } else {
    delta = 0;                             // Если скорость в пределах от 120 до 255 ничего не меняем
  }

  analogWrite(LP, speedRobot + delta); // +5 Прибавляем скорости левому двигателю
  analogWrite(RP, speedRobot);
}

void loop() {
  for (int i=50; i<140; i++){ // Постепенно увеличиваем скорость от 50 до 140
    speedRobot = i;           // смотрим, чтобы робот двигался прямо
    moveForward();
    delay(50);
  }
}
```

Листинг 3: Пример кода для движения прямо с ускорением.

Для того, чтобы робот развернулся нам нужно, чтобы одно колесо вращалось в одну сторону, а другое в противоположную. Установив стандартную скорость поворота, например, равную 70, нам остаётся только

замерить время, за которое робот совершит разворот. Теперь напишем функцию поворота и поэкспериментируем со временем.

```
void turnRobot(){
  digitalWrite(L1, HIGH); // Меням местами HIGH и LOW
  digitalWrite(L2, LOW);  // Меняется направление вращения левого мотора
  digitalWrite(R1, LOW);
  digitalWrite(R2, HIGH);

  analogWrite(LP, 70); // Задаём стандартную скорость поворота
  analogWrite(RP, 70); // Робот всегда будет поворачиваться с такой скоростью

  delay(980);          // Меняем время и смотрим, чтобы робот совершил полный оборот
}

void loop() {
    moveForward();    // Движемся прямо пол секунды
    delay(500);       // Задержка
    turnRobot();      // Разворачиваемся
}
```

Листинг 4: Пример кода для поворота.

Меняя время задержки delay(980) мы смотрим, чтобы робот совершил разворот на 180^0. Точно таким же образом можно написать функцию поворота на 90^0.

Для выполнения заданий потребуется, подключить модули и написать программу для того, чтобы робот мог двигаться по линии. Задача состоит из нескольких этапов, первым из которых будет подключение платы с микросхемами для управления моторами. Пример подключения и написания программы для управления двигателями приводится на специальном ролике:

https://www.youtube.com/watch?v=m6a-ofcWlzw&t=32s

После подключения двигателей необходимо добавить к роботу ИК датчики.

4. **Подключение ИК датчиков к плате Arduino и программирование робототехнического набора для движения по линии.**

Принцип работы датчика - светодиод излучает в инфракрасном диапазоне на длине волны 950 нм. Свет отражается от поверхности и попадает на фототранзистор. Сигнальный светодиод загорается, когда датчик находится над светлой (по его мнению) поверхностью, на выходе датчика устанавливается высокий логический уровень – «1» 3.3 или 5В в зависимости от напряжения питания датчика.

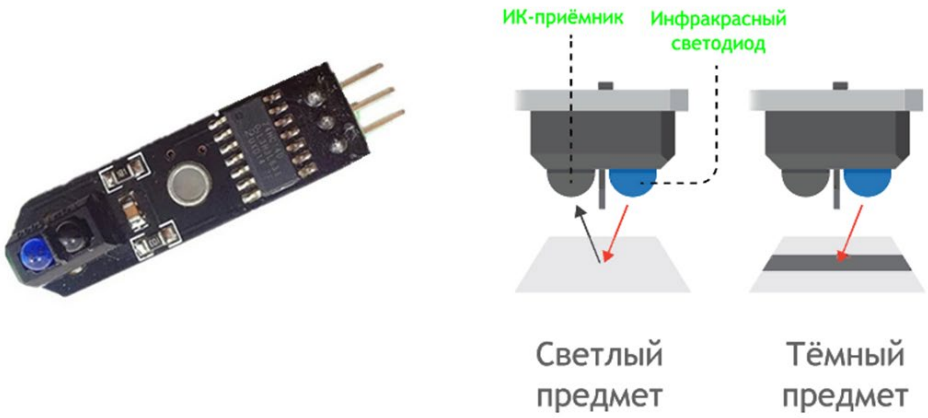

Рисунок 5: Общий вид ИК датчика и принцип его работы

Для запуска программы примера вам понадобится собранный робот и два ИК датчика поверхности. На плате управления робота подключите левый датчик к «11» порту, а правый - к «7».

При наезде датчика на черную линию, он возвращает плате логический «0», и наоборот, при наезде на белое поле – «1».

Подключение датчиков и работа с данными

- Питание (V) — красный провод. На него должно подаваться напряжение 5 В (или 3,3 В).
- Земля (G) — чёрный провод. Должен быть соединён с землёй микроконтроллера.
- Сигнальный (OUT) — жёлтый провод. Подключается к цифровому входу микроконтроллера. Через него датчик передает микроконтроллеру бинарное значение, ноль или единицу.

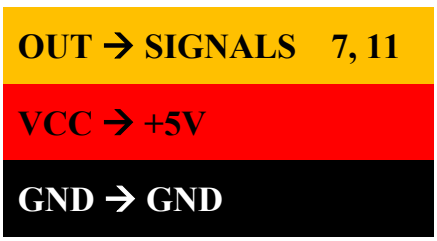

Рисунок. 6. Схема подключения ИК датчика при помощи цветных проводов.

Снимем данные с датчиков и отправим их в serial port

```
#define LEFTSENSOR 11 // Подключаем левый датчик к 11 порту на плате
#define RIGHTSENSOR 7 // Подключаем правый датчик к 7 порту на плате

void setup() {
  pinMode(LEFTSENSOR, INPUT);  // Настраиваем порты на вход
  pinMode(RIGHTSENSOR, INPUT);
  Serial.begin(9600);
}

void loop() {
  // Считываем значения с датчиков
  byte left = digitalRead(LEFTSENSOR);       // 0 - черный
  byte right = digitalRead(RIGHTSENSOR);     // 1 - белый

  Serial.print(left);
  Serial.println(right);

}
```

Листинг 5: Пример листинга кода для проверки ИК датчика

Движение по линии при помощи двух ИК датчиков.

В основном цикле:

1) Считываем показания с датчиков, определяем чёрный 0 или белый 1 цвет зафиксировали датчики.
2) Приписываем условия. Какие функции будут выполняться при комбинации данных с датчиков.

Таблица 3. Соотношение сигналов от датчиков и вызываемых функций для движения.

Левый датчик	Правый датчик	Выполняемая функция
		go();
		stp();
		rght();
		lft();

Функция go()

```
void go() // Движемся вперёд
{
  analogWrite(LP, speedRobot);
  digitalWrite(L1,LOW);
  digitalWrite(L2,HIGH);
  analogWrite(RP, speedRobot);
  digitalWrite(R1,LOW);
  digitalWrite(R2,HIGH);
}
```

Функция stp()

```
void stp() // Останавливаемся
{
  analogWrite(LP, 0);
  digitalWrite(L1,LOW);
  digitalWrite(L2,HIGH);
  analogWrite(RP, 0);
  digitalWrite(R1,LOW);
  digitalWrite(R2,HIGH);
}
```

Функция rght() и lft()

Прохождение плавных поворотов

```
void rght() // Поворачиваем направо
{
  analogWrite(LP, turnSpeed);  //Левое колесо движется
  digitalWrite(L1,LOW);
  digitalWrite(L2,HIGH);
  analogWrite(RP, 0);           //Правое колесо стоит на месте
  digitalWrite(R1,LOW);
  digitalWrite(R2,HIGH);
}

void lft() // Поворачиваем налево
{
  analogWrite(LP, 0); //Левое колесо стоит на месте
  digitalWrite(L1,LOW);
  digitalWrite(L2,HIGH);
  analogWrite(RP, turnSpeed);  //Правое колесо движется
  digitalWrite(R1,LOW);
  digitalWrite(R2,HIGH);
}
```

Основная программа:

```
#define L1 12
#define L2 4
#define RP 13
#define R1 8
#define R2 10

#define LEFTSENSOR 11 // Подключаем левый датчик к 11 порту на плате
#define RIGHTSENSOR 7 // Подключаем правый датчик к 7 порту на плате

void setup() {
  pinMode(LP, OUTPUT);
  pinMode(L1, OUTPUT);
  pinMode(L2, OUTPUT);
  pinMode(RP, OUTPUT);
  pinMode(R1, OUTPUT);
  pinMode(R2, OUTPUT);

  pinMode(LEFTSENSOR, INPUT);  // Настраиваем порты на вход
  pinMode(RIGHTSENSOR, INPUT);
}


 int speedRobot = 40;  // Создаём переменную скорости
 int turnSpeed = 20;   // Скорость поворотов

void loop() {
  // Считываем значения с датчиков
  byte left = digitalRead(LEFTSENSOR);       // 0 - черный
  byte right = digitalRead(RIGHTSENSOR);     // 1 - белый

  if ((left==1)&&(right==1))      //Если левый и правый датчики попадают на белое поле
  {                               //мы двигаемся вперёд
   go();
  }
  else if ((left==0)&&(right==0))  //Если левый и правый датчики попадают на чёрное поле поле
  {                                //мы останавливаемся
   stp();
  }
  else if ((left==1)&&(right==0))  //Если правый датчик попадает на чёрное поле поле
  {                                //мы поварачиваем направо
   rght();
  }
   else if ((left==0)&&(right==1))  //Если левый датчик попадает на чёрное поле поле
  {                                 //мы поварачиваем налево
   lft();
  }

}
```

Листинг 6: Пример листинга кода для движения по линии с использованием ИК датчиков

5. **Подключение ИК датчиков препятствий к плате Arduino и программирование робототехнического набора для движения по линии с объездом препятствий.**

Рассмотрим один из самых распространенных датчиков препятствия, который работает по принципу отражения. Устроен он очень просто. Датчик содержит направленный источник света и детектор света. Источником часто служит инфракрасный светодиод с линзой, а детектором — фотодиод или фототранзистор.

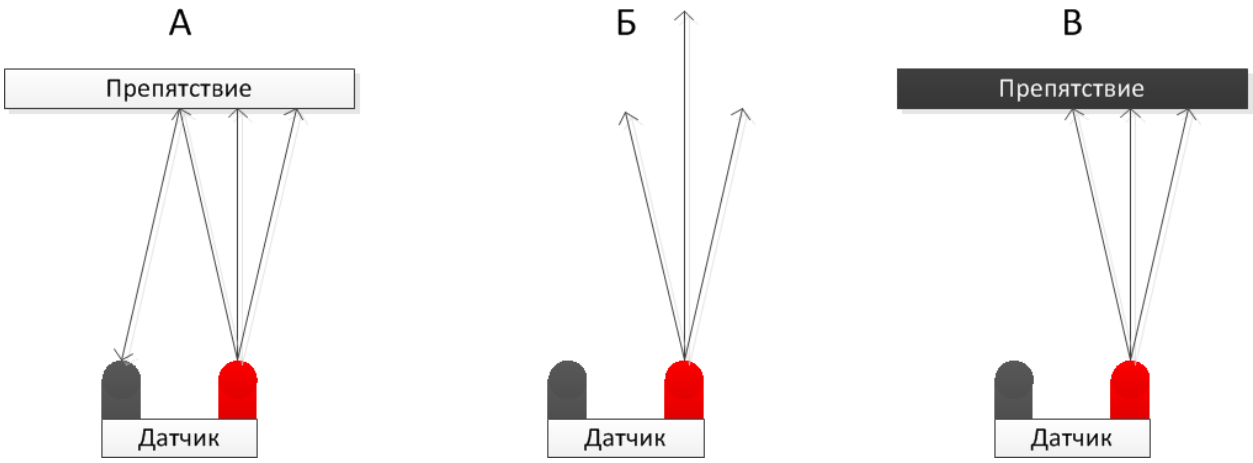

Рисунок 7 – Принцип работы ИК датчика препятствия.

- Если перед датчиком есть препятствие, то на детектор попадает отраженное ИК излучение от источника, и на выходе датчика появляется высокий логический уровень.
- Если препятствия нет, то ИК излучение на датчик не попадает и логический уровень на выходе низкий.
- Есть и третий вариант, когда препятствие есть, но свет от него не отражается! Получается, матовую черную поверхность робот не увидит.

При встрече с препятствием датчик возвращает «1» контроллеру, если же препятствия нет, то «0».

Рисунок.8 – Общий вид ИК датчиков препятствия.

Подключение датчика препятствия

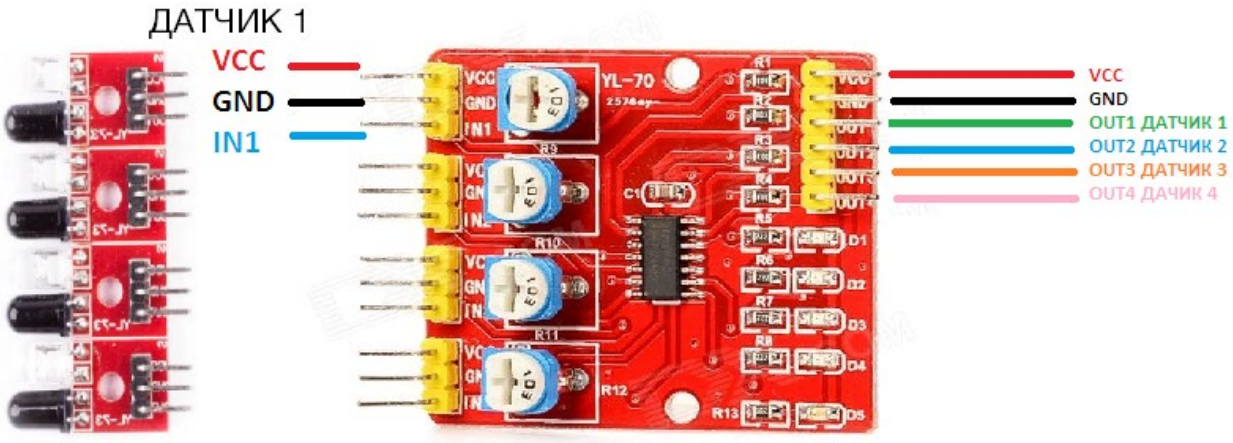

Рисунок 9 – Схема подключения ИК датчика препятствия.

Подключаем датчик, считываем сигнал в Serial и смотрим, на каком расстоянии срабатывает датчик. С помощью резистора на плате управления можно регулировать порог срабатывания датчика, тем самым изменяя расстояние срабатывания.

Движение по линии с объездом препятствия.

Проехать по чёрной линии, если на пути робота появляется банка, её нужно объехать и продолжить движение по треку.

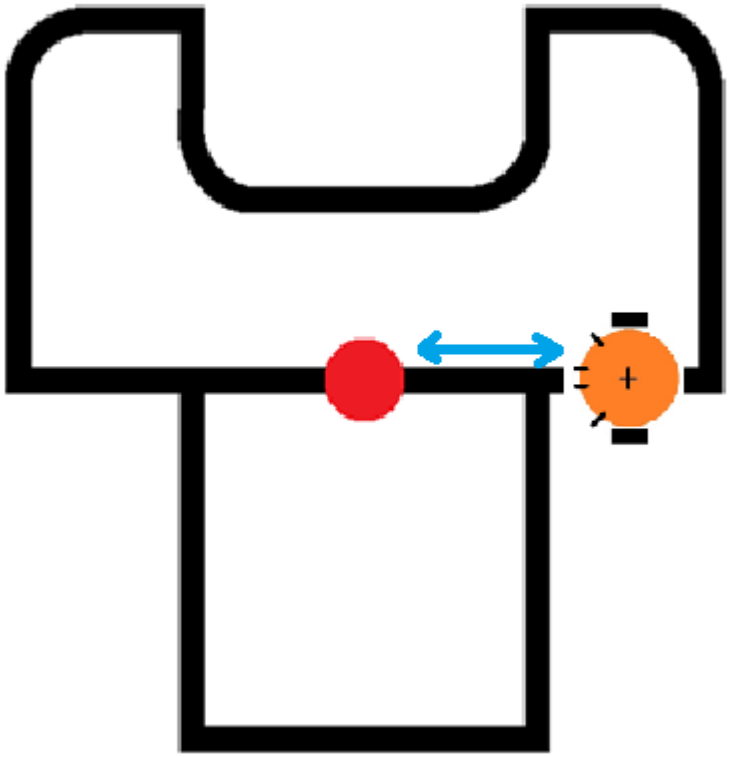

Рисунок 10 – Схема проезда трассы с препятствием.

Структурно алгоритм может быть построен следующим образом:

1. «РОБОТ УВИДЕЛ ПРЕПЯТСТВИЕ»
2. «РОБОТ ПРОДОЛЖАЕ ДВИЖЕНИЕ ПО ТРЕКУ ДО ПЕРЕКРЕСТКА»
3. «РОБОТ НАЧАЛ ОБЪЕЗД ПРЕПЯТСТВИЯ» (Повернул на перекрестке)
4. «РОБОТ ПРОДОЛЖАЕТ ДВИЖЕНИЕ ПО ТРЕКУ» (фиксирует центральным датчиком чёрную линию, до следующего перекрестка)

Альтернативный вариант алгоритма связан с использованием датчиков с меньшей дальностью обнаружения или предназначен для других комбинаций секций трека.

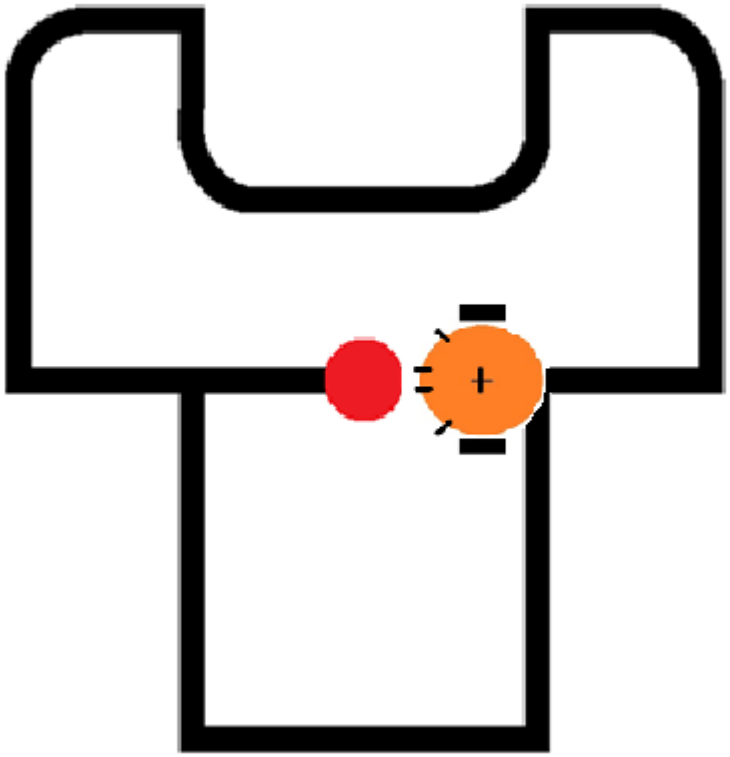

Рисунок 11 – Схема проезда трассы с препятствием вариант 2.

Тогда структурно алгоритм может быть построен следующим образом:

1. «РОБОТ УВИДЕЛ ПРЕПЯТСТВИЕ»
2. «РОБОТ СОВЕРШАЕТ РАЗВОРОТ»
3. «РОБОТ ПРОДОЛЖАЕ ДВИЖЕНИЕ ПО ТРЕКУ ДО ПЕРЕКРЕСТКА»
4. «РОБОТ НАЧАЛ ОБЪЕЗД ПРЕПЯТСТВИЯ» (Повернул на перекрестке)
5. «РОБОТ ПРОДОЛЖАЕТ ДВИЖЕНИЕ ПО ТРЕКУ» (фиксирует центральным датчиком чёрную линию, до следующего перекрестка)

Имеющуюся программу необходимо дополнить, так что бы к коду с передвижением по чёрной линии, добавить алгоритм объезда.

```
#define OBST_SEN 1
```
// ПОДКЛЮЧИМ ДАТЧИК НА ПОРТ 1

```
pinMode(OBST_SEN, INPUT);
```
// НЕ ЗАБУДЕМ ОБЪЯВИТЬ ПОРТ НА ВХОД В setup()

Добавляем новую функцию объезда препятствия

```
void obstSens(){
  if (digitalRead(OBST_SEN)==0){
    fastright(); // Если сработал датчик
    delay(1000); // То поворачиваем направо 1000 мк(100 градусов)

    while(1){ //Движемся по окружности, объезжая препятствие
      if(digitalRead(CENTERSENSOR)==0) // Если срабатывает центральный датчик черной линии
        break;      // то мы заканчиваем объезжать препятствие и переходим к основному алгоритму
      rndleft();
    }

    while(1){ // Доворачиваем направо, чтобы выровнить робота относительно линии
      if((digitalRead(LEFTSENSOR)==1) && (digitalRead(CENTERSENSOR)==0) && (digitalRead(RIGHTSENSOR)==1))
        break;
      fastright();
    }
  }
}
```

Листинг 7: Пример листинга кода для разворота перед препятствием.

Функция rndLeft – робот едет по окружности некоторого радиуса (участок 2 на рисунке)

```
void rndleft(){
  analogWrite(LP, 2*speedRobot);
  digitalWrite(L1,LOW);
  digitalWrite(L2,HIGH);
  analogWrite(RP, (speedRobot+diff));
  digitalWrite(R1,LOW);
  digitalWrite(R2,HIGH);
}
```

Эту функцию можно доработать для разворота на месте. Установив противоположные направления вращения для колес и одинаковую скорость. Итоговым результатом будет разворот на 180 градусов через левую или правую сторону.

Добавим функцию obstSens в основной цикл программы. Таким образом, перед тем как начать следовать линии, робот сначала всегда будет проверять есть ли препятствие

```
void loop() {
  obstSens();
```

Остальная часть кода содержит проверку состояния датчиков для движения по линии.

```
  if ((left==1)&&(center==0)&&(right==1)) //Если левый и правый датчики попадают на белое поле,
                                          //а средний на чёрное
  {                                       //Мы двигаемся вперёд
   go();
  }
  else if ((left==1)&&(center==1)&&(right==1))  //Если все датчики попадают на белое поле
  {                                             //Мы двигаемся вперёд
   go();
  }
  else if ((left==1)&&(center==1)&&(right==0))  //Если левый и средний датчики попадают на белое поле
  {                                             //Мы двигаемся направо
   rght();
  }
  else if ((left==0)&&(center==1)&&(right==1))  //Если правый и средний датчики попадают на белое поле
  {                                             //Мы двигаемся налево
   lft();
  }
  else if ((left==0)&&(center==0)&&(right==0))  //Если все датчики попадают на чёррное поле
  {                                             //Мы останавливаемся
   stp();
  }
  else if ((left==0)&&(center==0)&&(right==1))  //Если левый и средний датчики попадают на чёрное поле
  {                                             //РЕЗКИЙ ПОВОРТ НАЛЕВО
   turnlft();
  }
  else if ((left==1)&&(center==0)&&(right==0))  //Если правый и средний датчики попадают на чёрное поле
  {                                             //РЕЗКИЙ ПОВОРТ НАПРАВО
   turnrght();
  }
  else {                                        //В любом другом случае
    go();                                       //Мы двигаемся вперёд
  }
}
```

Листинг 8: Пример листинга кода с проверками состояний датчиков для движения по линии.

После отработки алгоритма движения по линии с тремя датчиками рекомендуется добавить плату, имеющую 5 датчиков линии в ряд. Подобное оснащения робота позволяет стабильно определять прохождение перекрестка за счет добавления дополнительных двух датчиков.

Для наглядности и простоты обучения в методических материалах присутствуют примеры кода, позволяющие проверить базовые функции и изучить основы программирования. Рекомендуется ознакомиться с материалами в процессе подготовки к выполнению заданий.

Приведенные в качестве примера программы специально упрощены для простоты понимания и наглядности работы. Для увеличения скорости движения по линии и плавности хода робота необходимо реализовывать регуляторы и более сложные алгоритмы обработки как датчиков линии, так и

датчиков препятствий. Указанных алгоритмов и программ достаточно, для того, чтобы робот начал двигаться, но для того, чтобы добиться успеха необходимо их серьезно усовершенствовать.