Санкт-Петербургский политехнический университет Петра Великого

Институт машиностроения материалов и транспорта

Высшая школа автоматизации и робототехники

Кафедра «Автономные роботы» (при ЦНИИ РТК)

Курсовая работа

Дисциплина: объектно-ориентированное программирование

Тема: разработка алгоритма автономного движения по линии

Студент гр. 3331506/00401 Бойко А. В.

Преподаватель Ананьевский М. С.

Санкт-Петербург

2023

Оглавление

Цель и задачи

Задачей курсовой работы является написание программы автономного движения по черной линии на заданном поле для микроконтроллера на платформе ОМЕГАБОТ. Алгоритм движения по линии должен выполнять следующие функции: поиск линии, движение по линии, возврат на линию в случае потери.

Введение

Платформа ОМЕГАБОТ предназначена для образовательных целей и участия в профильных соревнованиях по робототехнике. Она модульная, LEGO-совместимая, не требует пайки и инструментов для сборки. Программировать платформу можно как в Arduino IDE, так и в специальной визуальной среде [1].

ОМЕГАБОТ не единственная подобная платформа, есть и другие LEGO-совместимые программируемые наборы, но большинство из них слишком примитивные и предназначены для обучающихся начальных и средних классов [2]. Есть похожий аналог от компании Keyestudio, это три модификации робота в одной и различными вариантами контроллеров на борту, в том числе Arduino Nano [3].

На платформе ОМЕГАБОТ установлен контроллер Arduino UNO на базе микроконтроллера ATmega328 [4]. На этот контроллер загружалась написанная программа. Контроллер Arduino выбран из-за высокого уровня программирования, готовых библиотек и средств работы с периферией. Вследствие этого можно сосредоточиться на написании основного алгоритма, не отвлекаясь на разработку средств работы с другими устройствами.

Алгоритм движения по линии является распространенной задачей в робототехнике как соревновательной, так и промышленной. Он применяется в системах передвижений роботов Amazon [5], в заданиях для многих конкурсов и соревнований по робототехнике. Существуют различные методы реализации

алгоритма, в этой работе был использован метод конечного автомата с пропорциональным регулятором в основной части цикла движения.

Ход работы

Изучение платформы

Работа началась с изучения характеристик самой мобильной платформы (далее «тележки»). Были получены данные о минимальном сигнале, необходимом для запуска моторов, максимальном угле поворота, средние значения датчиков линии на белом покрытии, черном покрытии и на пересечении цветов. Были протестированы режимы торможения моторов: с замедлением, с остановкой, с реверсом. После было изучено само поле и особенности черной линии на нем

Для выполнения поставленной задачи была собрана следующая конфигурация модулей:

1) Мобильная платформа
2) Контроллер Arduino UNO
3) Два датчика линии
4) Кнопка

Подробно конфигурация представлена на рисунке 1.

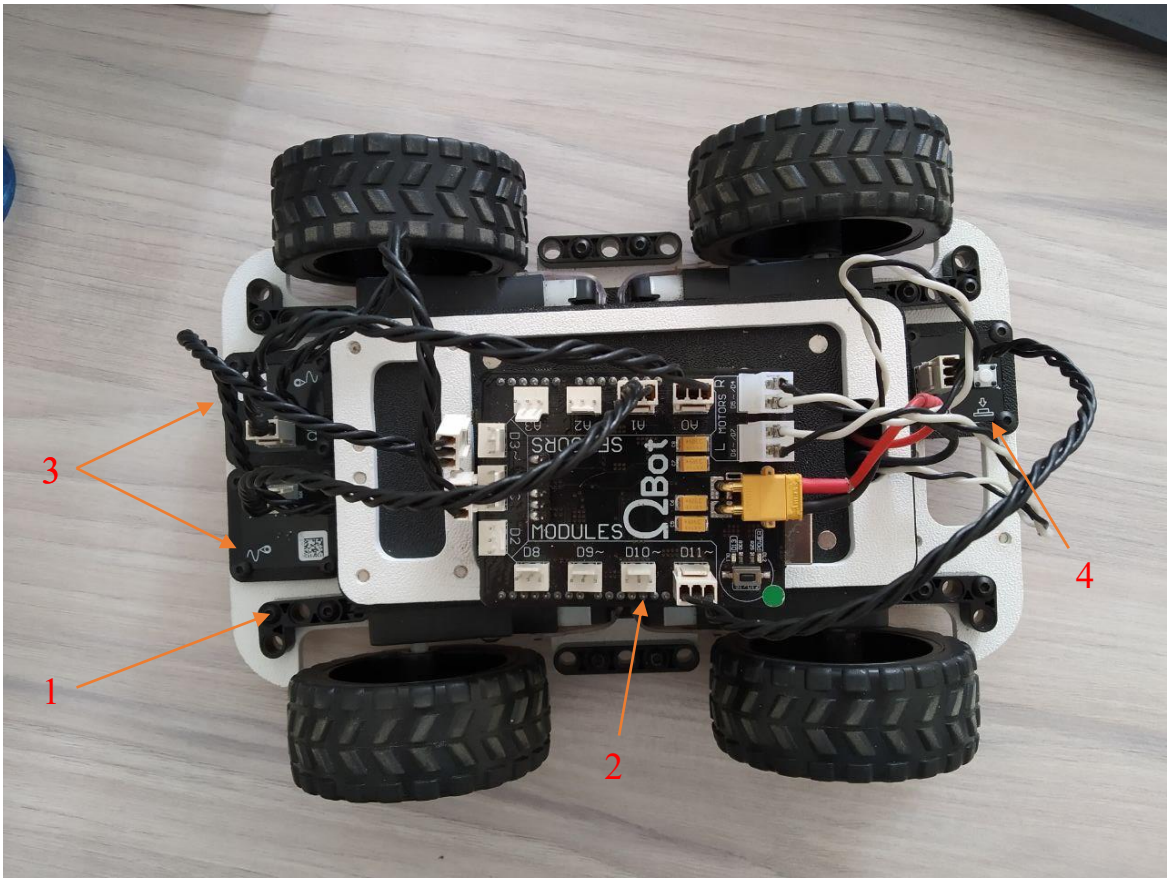
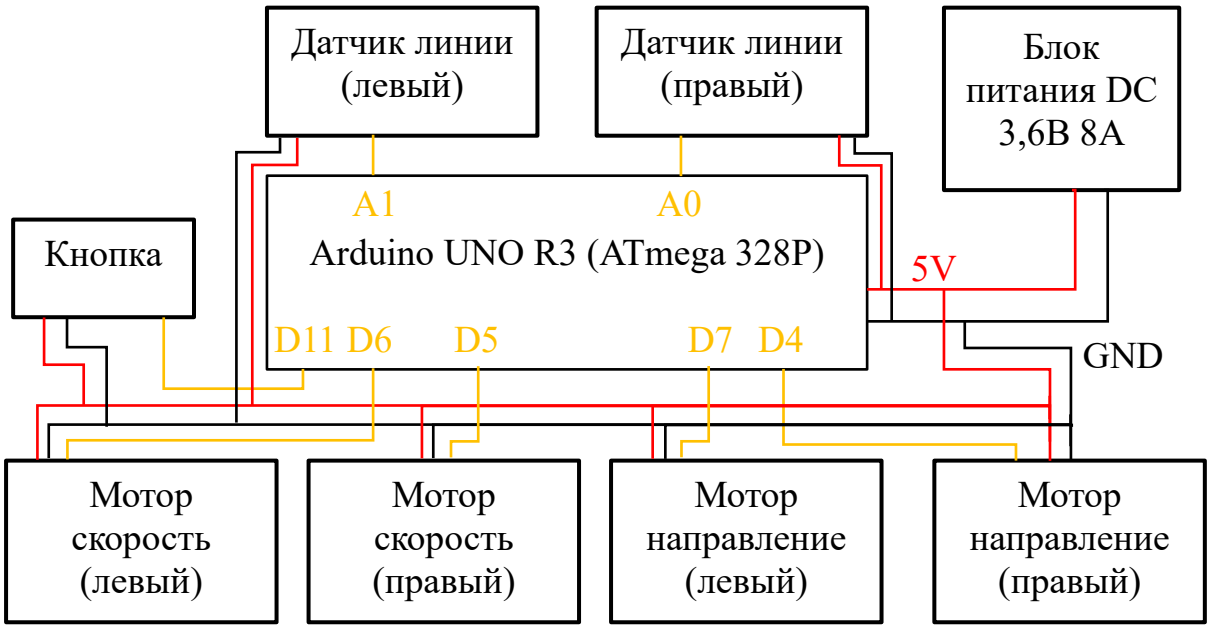


Рисунок 1 – конфигурация платформы



Алгоритм движения по прямой линии

Сначала был реализован алгоритм движения по прямой линии. Для этого применялся пропорциональный регулятор. Берется разность между показаниями левого датчика линии и правого (если разность больше нуля,

значит левый на черном, а правый нет и наоборот). Эта разность используется в качестве отрицательного коэффициента для скорости на левый мотор и положительного коэффициента на правый мотор. Далее эти скорости распределяются между 0 и максимальной заданной скоростью (если разность больше нуля, левый крутится медленнее, правый – быстрее и наоборот).

Чтобы разность между датчиками меньше зависела от особенностей каждого датчика и окружающего освещения, в начале программы производится балансировка датчиков: вычисляется разность датчиков на одинаковой поверхности. Поэтому при первом запуске робота нужно ставить его двумя датчиками на одинаковый цвет.

Алгоритм поворотов

Повороты «тележки» вызывают определенные трудности. Моторы начинают свое движение с определенного значения, передающегося на них через аналоговый выход (порядка 90 при заряженном аккумуляторе). Поэтому нельзя добиться любой кривизны поворота. Между состояниями «одно колесо стоит / другое крутится» и «одно колесо крутится быстрее / другое медленнее», есть большой пробел. Все повороты на заданном поле вписывались в этот пробел, поэтому для поворота по дуге использовалось состояние «одно колесо стоит / другое крутится».

Чтобы робот не сбивался с пути, если «тележка» не впишется в поворот, алгоритм отслеживает какой сенсор последним был на черном. Отталкиваясь от этого, алгоритм решает, какое колесо остановить, а какое крутить.

Алгоритм поиска линии

Для поиска линии было решено применить движение по спирали, чтобы охватывать все больший радиус. В программе это реализовано так: одно колесо вращается на максимальной скорости, другое начинает с небольшой скорости. Во время равное периоду скорость второго колеса увеличивается, а период увеличивается в два раза.

Реализация основных алгоритмов на этом окончена. Для плавности работы и устойчивости на линии были разработаны еще несколько алгоритмов.

Алгоритм выхода на линию

Этот режим нужен, чтобы робот плавно выходил на линию при большом угле входа (есть и слишком большие углы, когда робот пролетает линию). Этот алгоритм представляет собой обратный алгоритму движения по прямой линии, но с коэффициентом, возведенным в некоторую степень, для более резких реакций на изменение линии.

Алгоритм должен «отталкиваться» от линии, а не «притягиваться» к ней, как делает алгоритм прямой линии. Таким образом робот должен развернуть себя по расположению линии, после чего алгоритм будет закончен.

После испытаний оказалось, что этот алгоритм справляется в небольшом диапазоне углов вхождения и почти сразу заканчивается, так как выход на линию был завершен.

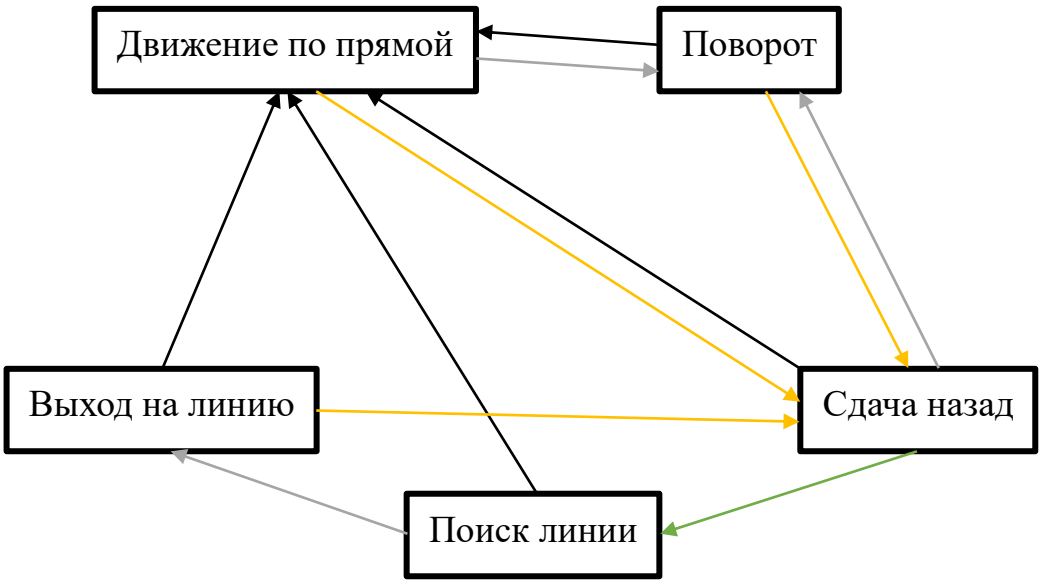
Режим сдавания назад

Это нельзя назвать алгоритмом, поэтому режим. Робот сдает назад, пока алгоритм не закончит свою работу. Нужен, чтобы возвращаться на линию, если робот с нее вылетел.

Реализация конечного автомата

Чтобы переключаться между несколькими режимами (в том числе и алгоритмами) необходима была система для отслеживания режима, текущих условий перехода и входного сигнала. Такой системой была выбрана система конечного автомата. В качестве входного сигнала служило текущее состояние на линии: оба датчика на черном, один на черном, оба датчика на белом. А в каждом режиме были прописаны режимы, в которые он может перейти, и условия перехода в каждый режим, в зависимости от входа.

Визуализация конечного автомата выглядит так (черный – оба на черном, серый – один на черном, желтый – оба на белом, зеленый – прошло 50 итераций):



Для определения состояния написана отдельная функция, которая сравнивает данные датчиков с некоторым константным значением, разделяющим белый и черный цвет.

Основное тело программы

В основном теле программы прописано считывание нажатия кнопки, после чего происходит запуск функции, соответствующей текущему режиму. В процессы выполнения функции значение глобальной переменной, несущей текущий режим работы, меняется и меняется запускаемая функция.

При повторном нажатии выполнение программы останавливается, а режим работы сбрасывается в поиск линии.

Дополнительные функции

В качестве дополнительных функций для облегчения работы с платформой были написаны: функция работы моторов, на вход подается любое целочисленное значение, как положительное, так и отрицательное, функция сама ограничивает значение рабочим диапазоном, определяет по знаку направление вращения и записывает соответствующие значения в пины

моторов; функция остановки, записывает в моторы скорость, соответствующую режиму остановки (0, если просто глушить моторы, -10, если с противозапуском); функция проверки черного, которая сравнивает значение датчика с граничным для черного, возвращает 1 если датчик на черном и записывает его в «последний на черном»; функция проверки состояния, которая проверяет каждый датчик на черный и возвращает одно из трех состояний; функция обработки режима, она просто запускает функцию из массива функций, соответствующую текущему режиму.

Код программы

Весь код алгоритма движения по линии представлен ниже

```
// Обозначение пинов
#define PIN_BUTTON 11

#define PIN_SENS_R A0
#define PIN_SENS_L A1

#define PIN_MOTOR_SPEED_R 5
#define PIN_MOTOR_SPEED_L 6
#define PIN_MOTOR_DIR_R 4
#define PIN_MOTOR_DIR_L 7

enum STATE {both_w, one_b, both_b};
enum MODE {search_mode, take_pos_mode, straight_mode, turn_mode, back_mode};
enum SENS {left, right};

void search();
void take_pos();
void straight();
void turn();
void back();

void (*(actions[5]))() = {search, take_pos, straight, turn, back};

void stop();


// Глобальные переменные состояний
int LAST_ON_BLACK = 0;
int CURRENT_MODE = search_mode;

// Глобальные константы для настройки работы алгоритмов
```

```
const int BLACK_VAL = 750;
const int STOP_SPEED = -50;
const int IDLE_SPEED = 110;
const int SPEED = 225;
int MAX_DIF = 300;

bool IS_ON = false;
float SENSORS_BALANCING;

void setup() {
  Serial.begin(9600);
  SENSORS_BALANCING =(analogRead(PIN_SENS_L) - analogRead(PIN_SENS_R));
}


int check_state(){
  int sen_state = check_black(left) + check_black(right);
  delay(10);
  return sen_state;
}


bool check_black(int sensor){
  int val = 0.0;
  if (sensor == left){
    val = analogRead(PIN_SENS_L);
  }
  if (sensor == right){
    val = analogRead(PIN_SENS_R);
  }

  if (val >= BLACK_VAL){
    LAST_ON_BLACK = sensor;
    return true;
  }
  else{
    return false;
  }
}


void stop(){
  int val_l = STOP_SPEED;
  int val_r = STOP_SPEED;

  run_motor(val_l,val_r);
  delay(100);
}


void back(){
```

```
  int val_l;
  int val_r;
  static int iterations = 0;
  if(LAST_ON_BLACK == left){
    val_l = -IDLE_SPEED;
    val_r = -IDLE_SPEED * 0.8;
  }
  else{
    val_l = -IDLE_SPEED * 0.8;
    val_r = -IDLE_SPEED;
  }
  run_motor(val_l,val_r);

  iterations++;

  if(iterations >= 50){
    CURRENT_MODE = search_mode;
    iterations = 0;
    return;
  }

  switch(check_state()){
    case both_b:
      CURRENT_MODE = straight_mode;
      iterations = 0;
      break;
    case one_b:
      CURRENT_MODE = turn_mode;
      iterations = 0;
      break;
  }
}


void search(){
  static float motor_koeff = 0.0;
  static float time = 0.0;
  static int period = 20;

  run_motor(SPEED, IDLE_SPEED * motor_koeff);

  time += 1.0;
  if(time > period){
    time = 0.0;
    motor_koeff = constrain(motor_koeff + 0.25 * (1 - motor_koeff), 0.0, 1.0);
    period *= 2;
  }


  switch(check_state()){
    case one_b:
```

```
      stop();
      CURRENT_MODE = take_pos_mode;
      motor_koeff = 0.0;
      period = 20;
      break;
    case both_b:
      stop();
      CURRENT_MODE = straight_mode;
      motor_koeff = 0.0;
      period = 20;
      break;
  }
  delay(100);
}


void take_pos(){
  float diff = analogRead(PIN_SENS_L) - analogRead(PIN_SENS_R) -
SENSORS_BALANCING;
  diff = float(constrain(diff, -MAX_DIF, MAX_DIF) / MAX_DIF);
  diff = pow(diff, 1.4);

  int val_l = map(int(255 * (diff)), -255, 255, 0, 2*IDLE_SPEED);
  int val_r = map(int(255 * (-diff)), -255, 255, 0, 2*IDLE_SPEED);


  run_motor(val_l, val_r);

  switch(check_state()){
    case both_b:
      CURRENT_MODE = straight_mode;
      break;
    case both_w:
      CURRENT_MODE = back_mode;
  }
}


void straight(){
  float diff = analogRead(PIN_SENS_L) - analogRead(PIN_SENS_R) -
SENSORS_BALANCING;
  diff = float(constrain(diff, -MAX_DIF, MAX_DIF) / MAX_DIF);

  int val_l = map(int(255 * (-diff)), -255, 255, 0, SPEED);
  int val_r = map(int(255 * (diff)), -255, 255, 0, SPEED);

  run_motor(val_l,val_r);

  switch(check_state()){
    case one_b:
      stop();
```

```
      CURRENT_MODE = turn_mode;
      break;
    case both_w:
      CURRENT_MODE = back_mode;
      break;
  }
}


void turn(){
  int val_l;
  int val_r;
  if (LAST_ON_BLACK == left){
    val_l = STOP_SPEED;
    val_r = SPEED;
  }
  else{
    val_l = SPEED;
    val_r = STOP_SPEED;
  }


  run_motor(val_l, val_r);

  switch(check_state()){
    case both_b:
      CURRENT_MODE = straight_mode;
      break;
    case both_w:
      CURRENT_MODE = back_mode;
      break;
  }
}


void run_motor(int val_l, int val_r){
  int dir_l;
  int dir_r;

  val_l = constrain(val_l, -250, 250);
  val_r = constrain(val_r, -250, 250);

  if (val_l > 0){
    dir_l = HIGH;
  }
  else{
    dir_l = LOW;
  }

  if (val_r > 0){
    dir_r = HIGH;
```

```
  }
  else{
    dir_r = LOW;
  }

  digitalWrite(PIN_MOTOR_DIR_L,dir_l);
  digitalWrite(PIN_MOTOR_DIR_R,dir_r);

  analogWrite(PIN_MOTOR_SPEED_L, abs(val_l));
  analogWrite(PIN_MOTOR_SPEED_R, abs(val_r));
}


void proceed(int mode){
  (*actions[mode])();
}


void loop() {
  // Считывание кнопки
  if (digitalRead(PIN_BUTTON) == HIGH){
    IS_ON = not IS_ON;
    CURRENT_MODE = 0;
    delay(1000);
  }

  // Обработка режима
  if (IS_ON){
    proceed(CURRENT_MODE);
  }
  else{
    // Выключение моторов
    digitalWrite(6,LOW);
    digitalWrite(5,LOW);
  }
}
```

Список литературы

1. ОМЕГАБОТ робототехническая платформа с программируемыми модулями - https://omegabot.ru
2. Китайские LEGO-совместимые робототехнические конструкторы - https://habr.com/ru/companies/robouniver/articles/695444/
3. Keyestudio Beetlebot 3 in 1 Robot for Arduino STEM Education - https://www.keyestudio.com/products/keyestudio-beetlebot-3-in-1-robot-for-arduino-stem-education
4. Amazon показала роботов, которых разработала для собственных складов - https://habr.com/ru/news/672788/