Министерство науки и высшего образования Российской Федерации
Санкт-Петербургский политехнический университет Петра Великого
Институт машиностроения, материалов и транспорта
Высшая школа автоматизации и робототехники

ОТЧЁТ О НАУЧНО-ИССЛЕДОВАТЕЛЬСКОЙ РАБОТЕ

Программирование робота OmegaBot на движение до препятствия

по направлению подготовки 15.03.06 «Мехатроника и робототехника»
направленность (профиль) 15.03.06_01 «Проектирование и конструирование мехатронных модулей и механизмов роботов»

Выполнил
студент гр. 3331506/10101 Непомнящий И.А.

Научный руководитель
доцент ВШАиР К.Ф.-М.Н. Ананьевский М.С.

Санкт-Петербург
2024

Содержание

ВВЕДЕНИЕ

OmegaBot – робототехническая платформа, разработанная специально для обучения робототехнике и программированию в визуальной среде, C++ и Python в образовательных учреждениях.

Основу конструктора составляют универсальный программируемый контроллер Omegabot и колесная платформа со встроенным аккумуляторным блоком, имеющая гнезда крепления для навесных модулей. Программируемый контроллер разработан на архитектуре Arduino, что обеспечивает возможность использования огромной открытой базы библиотек для решения самых разнообразных учебных и технических задач, а также совместимость с физическими модулями разных производителей.

Модульная конструкция OmegaBot’a позволяет использовать множество различных компонентов для дальнейшего программирования и работы. Одним из таких является ультразвуковой сенсор, который позволяет измерять расстояние до объекта. Использование этого датчика может быть использовано для предотвращения столкновения OmegaBot'a с препятствием.

Цель работы – программирование OmegaBot’a на движение до препятствия и поворот при его обнаружении, с продолжением движения.

Задачи:

1) выбор необходимого модуля
2) написание программы

1.Выполнение задачи

Устройство робота

Внешний вид колёсной платформы(рис 1).

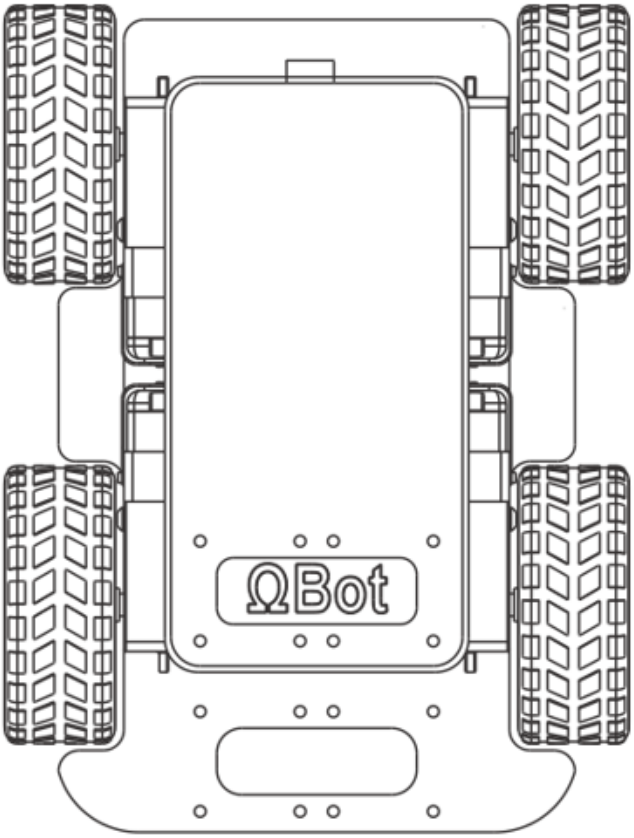

Рис. 1. Колесная платформа

Колесная платформа является несущей конструктивной частью Конструктора и обеспечивает механическую устойчивость создаваемых робототехнических устройств при эксплуатации. Колесная платформа состоит из жесткой металлической рамы и съёмной основы. Также на раме крепится блок аккумуляторов и мотор-колёса. Внешними габаритами колесной платформы являются края пластиковой пластины, достаточно гибкой, чтобы демпфировать удары при падении конструкции с высоты учебного стола. Также демпфирующее свойство пластины, в сочетании с применением магнитных креплений внешних модулей, приводит к свободному отсоединению последних и снижает риск их повреждения.

На колесной платформе с базовой основой предусмотрены четыре гнезда для магнитного крепления внешних модулей, что позволяет устанавливать на платформу до четырех датчиков или исполнительных устройств дополнительно к интегрированным на плату контроллера.

Основные технические характеристики колесной платформы с базовой основой

Тип основы колесная

Количество мотор-колёс 4

Ток потребления мотора, не более 4 А

Номинальное напряжение, электропитания мотора 7,4 В

Количество универсальных креплений для модулей 5

Выбор модуля.

Для того, чтобы определять препятствие, необходим сенсор. Был выбран ультразвуковой дальномер HC-SR04 (рис.2).

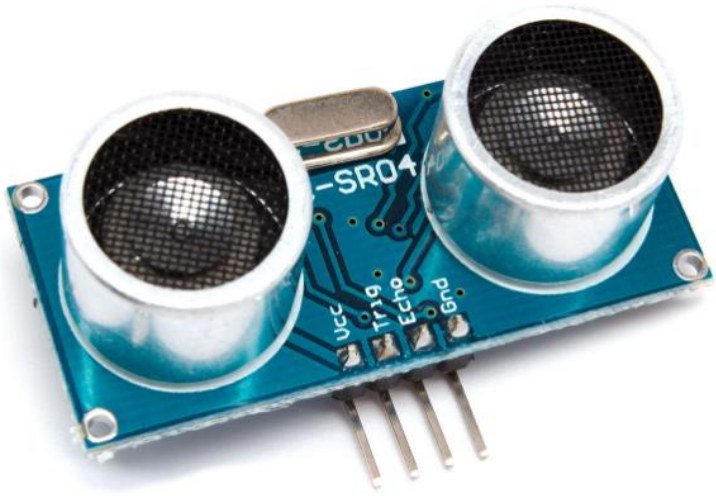

Рис. 2. ультразвуковой дальномер HC-SR04

Описание работы модуля. Модуль УЗ дальномера, используя принцип эхолокации, посылает сигнал и принимает его отражение от объекта. Время между отправкой и получением сигнала используется для определения расстояния. В состав ультразвукового дальномера входят дальномер HC-SR04 и вычислительный модуль.

Принцип работы дальномера HC-SR04 заключается в излучении ультразвукового сигнала и приеме его отражения от объекта наблюдения. При этом запуск излучения осуществляется каким-либо внешним устройством управления подачей импульса на вывод, а при приеме отраженного сигнала дальномер HC-SR04 формирует импульс на выводе. Таким образом может быть измерено время между началом излучения и приемом отраженного сигнала. То есть то время, за которое ультразвук преодолел путь, равный двум расстояниям до объекта наблюдения. Результатом умножения половины полученного значения времени на скорость звука является расстояние до объекта наблюдения.

Все перечисленные функции и вычисления автоматически выполняет вычислительный модуль дальномера, построенный на основе восьмиразрядного микроконтроллера. При обращении по интерфейсу I2C он выдает расстояние до объекта наблюдения в сантиметрах в диапазоне от 2 до 254 см.

Написание программы.

Робот начинает движение после нажатия кнопки на плате, пока расстояние до препятствия будет больше 15 см, робот будет двигаться прямо, после обнаружения препятствия, робот поворачивается на 90 градусов направо и так повторяется 5 раз.

Алгоритм программы в виде блок схем представлен на рисунке 2.

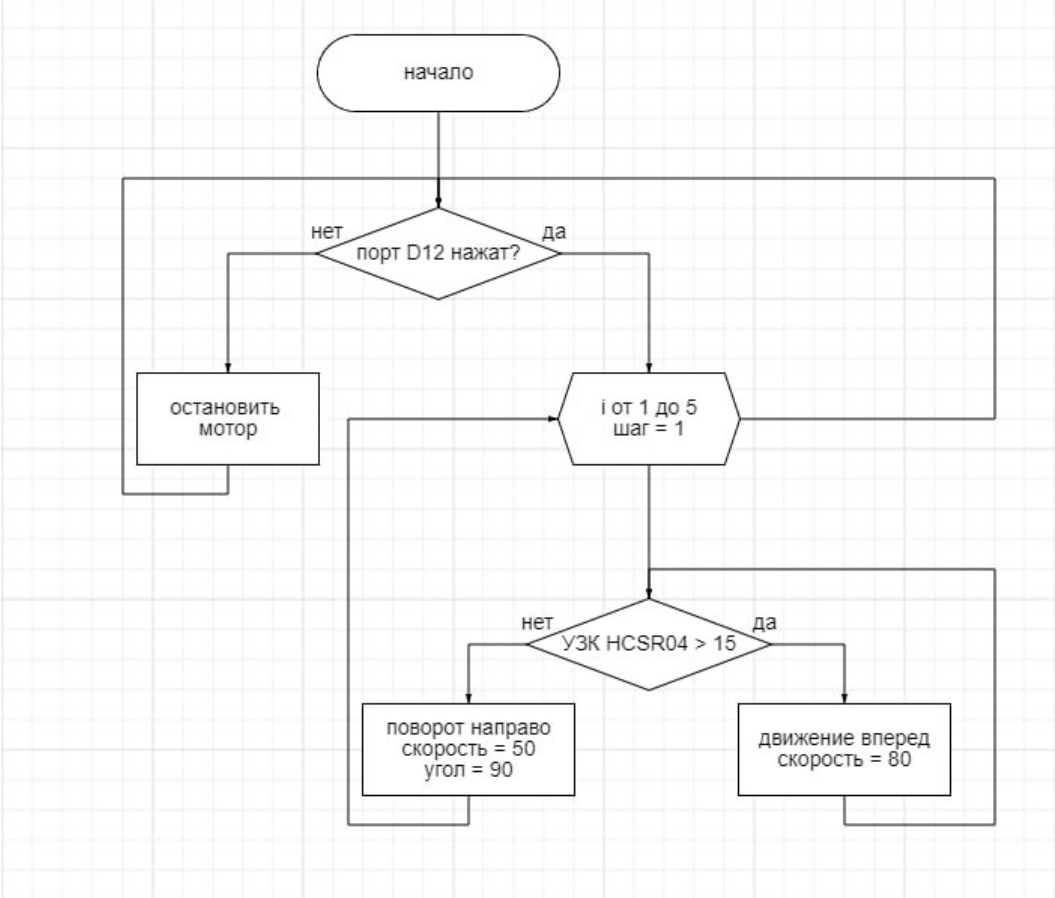

Рис. 3. Алгоритм движения.

Код программы представлен в приложении 1.

2. Результат работы

По результатам работы, OmegaBot, после нажатия кнопки начинает движение вперед. При обнаружении препятствия на расстоянии 15 сантиметров, робот поворачивает направо на 90 градусов и продолжает движение прямо. Работа робота прекращается после 5 поворотов. Робот движущийся к препятствию изображен на риснке 4.

Рис. 4. OmegaBot на пути к препятствию.

Заключение

В ходе выполнения научно-исследовательской работы, предоставилась возможность ознакомиться с робототехнической модульной платформой OmegaBot. Был управляющий код, для автономного движения робота на заданное количество поворотов при обнаружении препятствий.

Были получены практические и теоретические знания, необходимые для работы с данной платформой.

СПИСОК ЛИТЕРАТУРЫ

1. C/C++. Программирование на языке высокого уровня // Т.А. Павловская — СПб.: Питер, 2011 .
2. Margolis M. Arduino Cookbook. // O'Reilly Media, 2011.
3. Blum J. Exploring Arduino: Tools and Techniques for Engineering Wizardry. //- Wiley, 2014.

Приложение 1.

```
#define M1_DIR 4
#define M1_PWM 5
#define M2_DIR 7
#define M2_PWM 6
#define TRIG_PIN A4
#define ECHO_PIN A5
#define ENCODER_K_DIST 1
#define ENCODER_K_ANGLE 2.37
unsigned long EncoderCount1, EncoderCount2;
int ButtonRead(int ButtonPin)
{
  int Data = digitalRead(ButtonPin);
  if(ButtonPin == 12) Data = !Data;
  return Data;
}

int _1_a;
void InitMotors()
{
  pinMode(M1_DIR, OUTPUT);
  pinMode(M1_PWM, OUTPUT);
  pinMode(M2_DIR, OUTPUT);
  pinMode(M2_PWM, OUTPUT);
}

void Motors(int Speed1, int Speed2)
{
  if(Speed1 > 255) Speed1 = 255;
  if(Speed1 < -255) Speed1 = -255;
  if(Speed2 > 255) Speed2 = 255;
  if(Speed2 < -255) Speed2 = -255;
```

```
  if(Speed1 > 0)
  {
    digitalWrite(M1_DIR, 1);
    analogWrite(M1_PWM, Speed1);
  }
  else
  {
    digitalWrite(M1_DIR, 0);
    analogWrite(M1_PWM, -Speed1);
  }

  if(Speed2 > 0)
  {
    digitalWrite(M2_DIR, 1);
    analogWrite(M2_PWM, Speed2);
  }
  else
  {
    digitalWrite(M2_DIR, 0);
    analogWrite(M2_PWM, -Speed2);
  }
}

//Ехать вперед
void MoveForward(int Speed)
{
  Motors(Speed, Speed);
}

void InitUltrasonicHCSR04()
{
```

```
  pinMode(TRIG_PIN, OUTPUT);
}

int UltrasonikHCSR04Read()
{
  digitalWrite(TRIG_PIN, 1);
  delayMicroseconds(10);
  digitalWrite(TRIG_PIN, 0);

  int Data = pulseIn(ECHO_PIN, 1) / 55;

  return Data;
}

void InitEnc()
{
  pinMode(2, INPUT_PULLUP);
  pinMode(3, INPUT_PULLUP);
  attachInterrupt(0, nEncoder1, CHANGE);
  attachInterrupt(1, nEncoder2, CHANGE);
}

void nEncoder1()
{
  EncoderCount1++;
}

void nEncoder2()
{
  EncoderCount2++;
}
```

```
void Encoder1ToNull()
{
  EncoderCount1 = 0;
}

void Encoder2ToNull()
{
  EncoderCount2 = 0;
}

void EncodersToNull()
{
  EncoderCount1 = 0;
  EncoderCount2 = 0;
}

unsigned long GetAngleFromEnc1()
{
  return (double)EncoderCount1 * (double)ENCODER_K_ANGLE;
}

unsigned long GetAngleFromEnc2()
{
  return (double)EncoderCount2 * (double)ENCODER_K_ANGLE;
}

void MoveRight(int Speed)
{
  Motors(-Speed, Speed);
}

void Stop()
```

```
{
  Motors(0, 0);
}

void MoveRightByEncoder(int Speed, int Angle)
{
  EncodersToNull();
  MoveRight(Speed);

  while(1)
  {
    if(GetAngleFromEnc1() > Angle) break;
    if(GetAngleFromEnc2() > Angle) break;
    Serial.print("1");
  }

  Stop();
}


void setup()
{
  pinMode(12, INPUT_PULLUP);
  InitMotors();
  InitUltrasonicHCSR04();
  InitEnc();
}

void loop()
{
  while ( ButtonRead(12) )
  {
```

```
  for (_1_a=1; _1_a<= ( 5 ); ++_1_a )
  {
   do
   {
    MoveForward(80);
   }
   while(( ( UltrasonikHCSR04Read() ) > ( 15 ) ));
   MoveRightByEncoder(50, 90);
  }
 }

 Stop();
}
```