Санкт-Петербургский политехнический университет Петра Великого
Институт машиностроения, материалов и транспорта
Высшая школа автоматизации и робототехники

Курсовая работа

по дисциплине «Объектно-ориентированное программирование»

Тема: Разработка алгоритма движения робототехнической платформы на основе показаний ультразвукового датчика расстояния

Выполнил студент гр. 3331506/00401 С. А. Поздняков

Преподаватель М. С. Ананьевский

Санкт-Петербург
2023

Оглавление

Цель и задачи

Написание программного кода для микроконтроллера, позволяющая определять препятствия и расстояния до них с помощью ультразвукового датчика для реализации операции парковки робототехнической платформы Omegabot.

Введение

Описание робототехнической платформы, используемой для реализации задания: четырехколесный робот с дополнительной платформой и закрепленными на ней энкодерами, модулем схвата (захватной системой) и техническим зрением на основе Arduino и Raspberry Pi, возможна реализация таких задач, как движение по трассе, движение по лабиринту, парковка робота.

Данная робототехническая платформа имеет 4 двигателя и 4 присоединенных к ним колеса без возможности поворота относительно оси симметрии платформы. Следовательно, поворот может быть осуществлен только путем замедления одной пары колес и ускорения другой. Для движения робота применяется колесная платформа Omegabot. Колесная платформа является несущей конструктивной частью робототехнических устройств и позволяет размещать на ней необходимые компоненты (схваты, датчики и т. д.).

В данном случае для реализации операции парковки будут задействованы сама платформа с двигателями и энкодерами, модуль захвата для удержания ультразвукового дальномера, ультразвуковой дальномер HC-SR04, модуль с кнопкой без фиксации, модуль пьезоизлучателя.

Написанная программа для реализации задания загружена на программируемый микроконтроллер Arduino Uno. Arduino Uno – плата микроконтроллера на базе ATmega328P, имеющая 14 цифровых портов ввода/вывода, 6 аналоговых портов. Питание возможно от аккумулятора или

от источника постоянного тока. Использование платформы Arduino позволяет использовать различные библиотеки для упрощения кода и возможности использования аналогичных алгоритмов для решения разных задач.

Блоки программы

- Активация и деактивация выполнения программы по кнопке. В данном случае используется модуль с кнопкой от Omegabot без фиксации. Возможно использовать иные методы включения/выключения (кнопка с фиксацией, геркон, таймер), но это потребует внесения изменений в код программы.
- Получение данных с ультразвукового датчика расстояния.
- Звуковое оповещение при близком приближении робота к препятствию
- Управление моторами на основании полученных данных (поворот платформы, ускорение, замедление, остановка перед препятствием).

Вид платформы Omegabot с используемыми компонентами представлен на рисунке 1.

Рисунок 1 — Общий вид платформы с используемым навесным оборудованием

Ход работы

Фотография используемого оборудования с указанными позициями элементов представлена на рисунке 2.

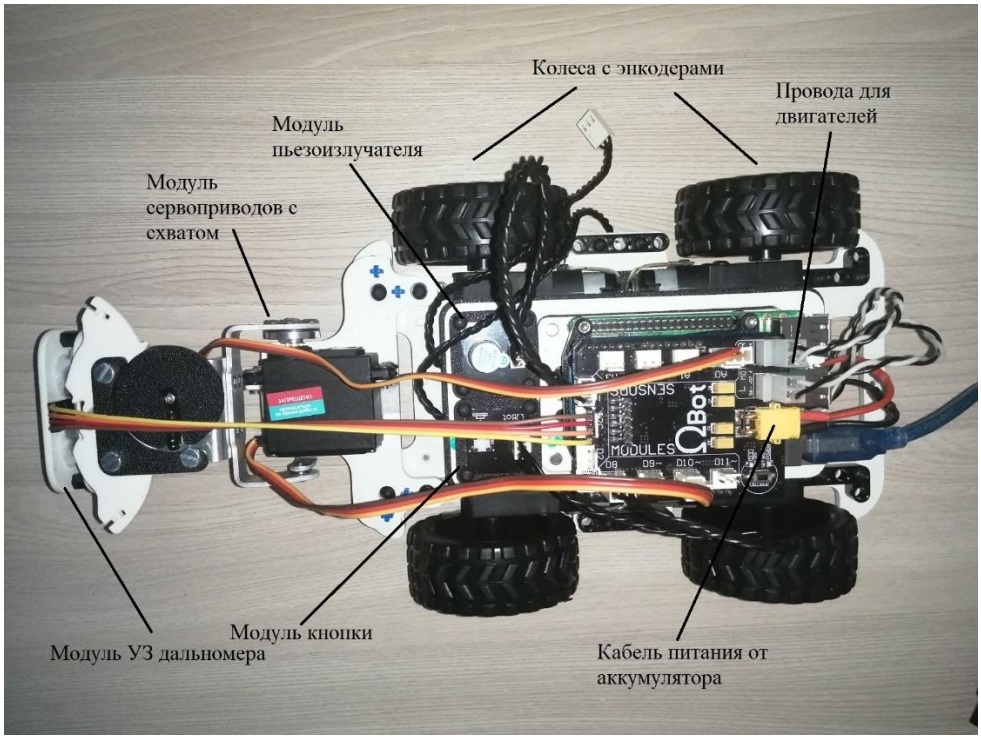


Рисунок 2 — Используемые компоненты

Схема используемых компонентов робототехнической платформы представлена на рисунке 3.

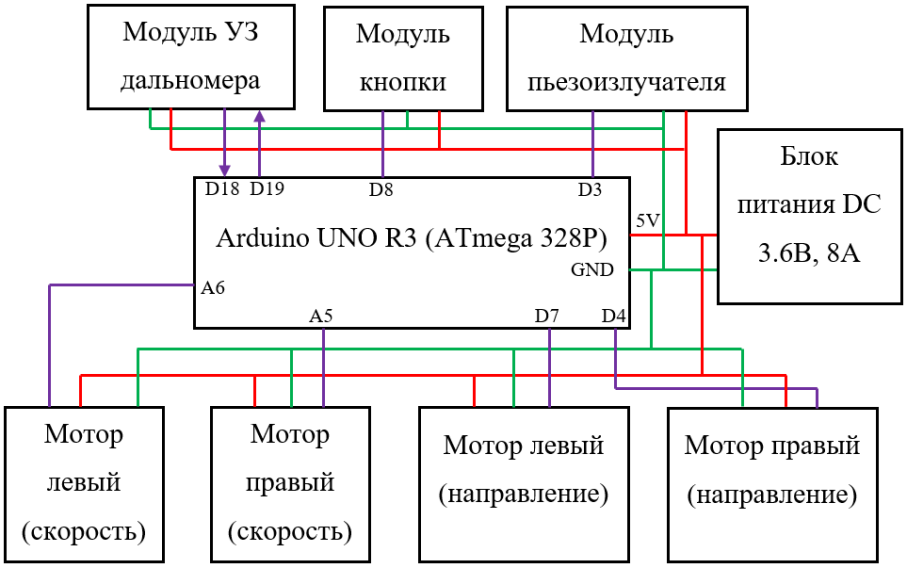


Рисунок 3 — Схема

Изучение схемы питания

Робототехническая платформа Omegabot питается от аккумулятора Li-ion 18650 с номинальной емкостью 3350 мАч и номинальным напряжением 3.6 В. Заряжается аккумулятор от сети 220В с помощью блока питания на 12В и 8А.

Реализация активации и деактивации выполнения программы

Активация и деактивация выполнения программы реализуется с помощью кнопки без фиксации, то есть подается сигнал высокого уровня только в момент замыкания контактов на кнопке (нажатия) и при размыкании контактов сигнал становится обратно низкоуровневым. Схематичное изображение используемого модуля кнопки представлено на рисунке 4.

Рисунок 4 — Модуль кнопки

Ниже представлена часть кода программы, реализующая выполнение данного блока.

```
#define BUTTONPIN 8  // кнопка включения и отключения
bool is_on = false;  // для кнопки включения
bool check_button() {
  if (digitalRead (BUTTONPIN) == HIGH) {
    button_state = not button_state;
    current_mode = 0;
    delay(100);
  }
  return button_state;
}
```

Изначально булева переменной is_on задано значение false, что означает остановку всех моторов. Во время выполнения основной части программы

происходит проверка, если было нажатие кнопки, и переменная is_on стала иметь значение true, то начинает выполняться функция для работы моторов. При следующем нажатии кнопки переменная is_on станет иметь значение false, а скорость моторов будет сброшена до нуля (робот остановится и прекратится выполнение программы). Результат работы данной функции вызывается в основном цикле программы void loop.

Получение и обработка данных с датчика

Для реализации задания используется ультразвуковой дальномер HC-SR04. Модуль ультразвукового дальномера использует принцип эхолокации: посылает сигнал и принимает его отражение от объекта. При этом запуск излучения осуществляется каким-либо внешним устройством управления подачей импульса на вывод, а при приеме отраженного сигнала дальномер формирует импульс на выводе. Таким образом может быть измерено время между началом излучения и приемом отраженного сигнала, а путем простейших математических преобразований полученное время может быть переведено в единицы длины (сантиметры и тд.) путем умножения половины полученного значения на скорость звука. Данный датчик обращается по интерфейсу I2C и выдает значения в сантиметрах от 2 до 254 см. В том случае, если предполагается работа робота с данным датчиком не при постоянной температуре, а например на улице с постоянными перепадами, то следует использовать датчик температуры для корректировки скорости звука. Скорость звука зависит от температуры: при +20°C это 343 м/с, а при -20°C – 318 м/с. Это 318/343=7%, что на расстоянии в 1 метр даст погрешность 7 сантиметров. Схематическое изображение модуля ультразвукового дальномера представлено на рисунке 5.

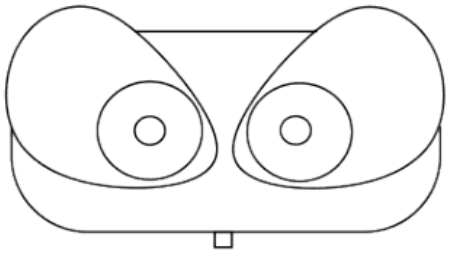


Рисунок 5 — Модуль УЗ дальномера

Ниже представлена часть кода программы, реализующая выполнение данного блока.

```
#define ECHOPIN 19  // приемник
#define TRIGPIN 18  // источник
float filt_param = 0.2;  // параметр для фильтрации
float dur_param = 29.1;

int get_distance() {
  long dur, sm;
  digitalWrite(TRIGPIN, LOW);
  delayMicroseconds(2);
  digitalWrite(TRIGPIN, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIGPIN, LOW);
  dur = pulseIn(ECHOPIN, HIGH);
  sm = (dur / 2) / dur_param;  // перевод показаний датчика в сантиметры
  delay(100);
  //sm = map(sm, 0, 350, 0, 255);
  return (sm);
}

int dist_filtered() {
  int dist = get_distance();
  float dist_filt = 0;  // отфильтрованный выход
  dist_filt += (dist - dist_filt) * filt_param;
  return (dist_filt);
  Serial.print("Distance is ");
  Serial.print(dist_filt);
  Serial.print(" sm");
}
```

Примеры получаемого сигнала до использования функции фильтрации и после представлены на рисунках 6 и 7.

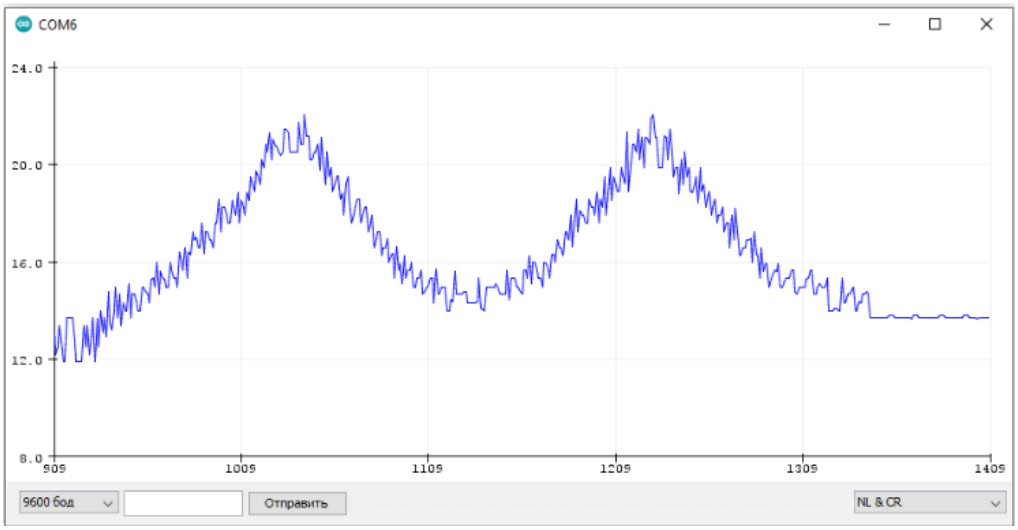


Рисунок 6 — Сигнал до фильтрации

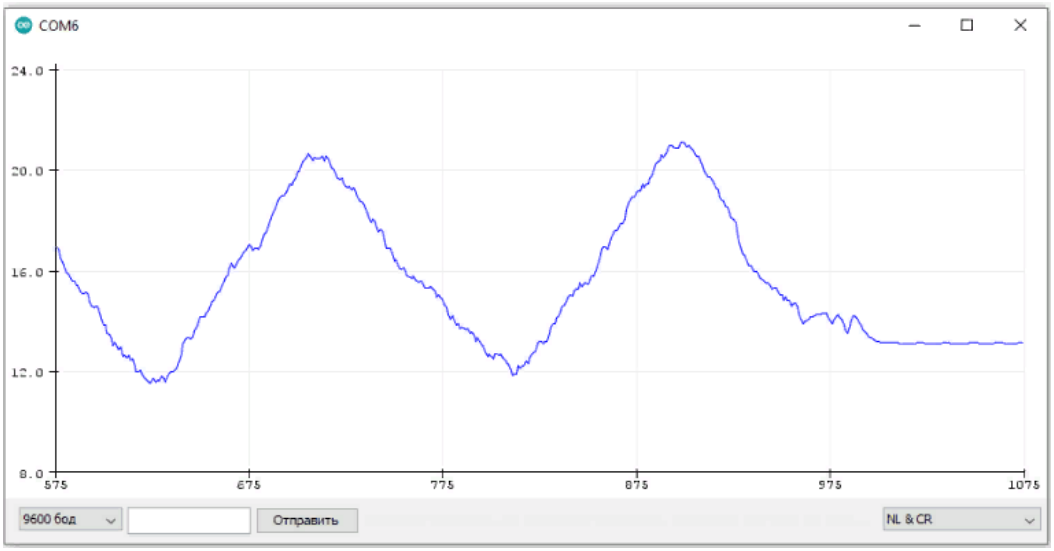


Рисунок 7 — Сигнал после фильтрации

Звуковое оповещение при близком приближении робота к препятствию

Для оповещения потенциального пользователя о близком приближении робота к препятствию добавлена функция звукового сигнала, громкость которого пропорциональна расстоянию до препятствия. Для реализации данной функции используется модуль пьезоизлучателя Omegabot. При наличии управляющего сигнала воспроизводится звуковой сигнал. Схематическое изображение модуля пьезоизлучателя представлено на рисунке 8.

Рисунок 8 — Модуль пьезоизлучателя

Ниже представлена часть кода программы, реализующая выполнение данного блока.

```
int peep_time() {
  int sound = dist_filtered();
  int sound_time = map(sound, 0, 255, 700, 100);
  //tone(NOTEPIN, 1000, sound_time);
  return (sound_time);
}
```

Управление моторами

Данный этап разработки программного кода заключается в построении алгоритма управления моторами (скорость вращения и направление). В качестве исходных данных используются показания ультразвукового дальномера.

Итоговый код программы представлен ниже:

```
#include <Servo.h>
//#include <Ultrasonik.h>
#define ECHOPIN 19  // приемник
#define TRIGPIN 18  // источник
#define BUTTONPIN 8  // кнопка включения и отключения
#define NOTEPIN 3  // оповещение о приближении
#define SPEED_R 5  // скорость правых моторов
#define SPEED_L 6  // скорость левых моторов
#define DIR_R 4  // направление правых моторов
#define DIR_L 7  // направление левых моторов
#define PERIOD 100;

Servo sensor_servo;
enum MODE {move_towards, detour_object};

void move();
void detour();
```

```
void stop();

void (*(actions[3]))() = {move, detour, stop};

int current_mode = move_towards;
bool is_on = false;  // для кнопки включения
bool button_state = false;
float time = 0.0;
int sound_time = 0;  // время оповещения
float filt_param = 0.2;  // параметр для фильтрации
float dur_param = 29.1;


void setup() {
  Serial.begin(9600);
  pinMode(TRIGPIN, OUTPUT);
  pinMode(ECHOPIN, INPUT);
  pinMode(BUTTONPIN, INPUT);

  sensor_servo.attach(10);
 // sensor_servo.write(45);
  delay(100);

  tone(NOTEPIN, 1000, 500);
}

int period(int mil) {
  uint32_t tmr;
  if (millis() - tmr >= mil) {
    tmr = millis();
  }
}

void sens_privod(int degree) {
  sensor_servo.write(degree);
  delay(500);
}

void set_mode (int new_mode) {
  switch(new_mode) {
    case move_towards:
      move();
      break;

    case detour_object:
      detour();
      break;

    default:
      stop();
```

```
      break;

    }
    current_mode = new_mode;
    Serial.print(new_mode);
}


void move() {
  int value_right = 1;
  int value_left = 1;

  start_motors(value_right, value_left);
}

void detour() {
  int value_right = 1;
  int value_left = 1;

  start_motors(value_right, value_left);
}

void stop() {
  int value_right = 0;
  int value_left = 0;
  start_motors(value_right, value_left);
}

void start_motors (int value_right, int value_left) {
  int direction_left;
  int direction_right;
  if (value_right > 0) {
    direction_right = 1;
  }
  else {
    direction_right = 0;
  }

  if (value_left > 0) {
    direction_left = 1;
  }
  else {
    direction_left = 0;
  }
  digitalWrite(DIR_R, direction_right);
  digitalWrite(DIR_L, direction_left);
  analogWrite(SPEED_R, value_right);
  analogWrite(SPEED_L, value_left);
}
```

```
void program() {
  int mode = 0;
  mode = find_object;
  set_mode(mode);
  delay(300);
}

void find_object() {

}

void ride() {
  int p_time = peep_time();

  digitalWrite(DIR_R, 1);
  digitalWrite(DIR_L, 0);
  analogWrite(SPEED_R, 80);
  analogWrite(SPEED_L, 80);

  int sm = dist_filtered();
  if (sm < 20) {
      tone(NOTEPIN, 1000, p_time);
  }
}

int get_distance() {
  long dur, sm;
  digitalWrite(TRIGPIN, LOW);
  delayMicroseconds(2);
  digitalWrite(TRIGPIN, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIGPIN, LOW);
  dur = pulseIn(ECHOPIN, HIGH);
  sm = (dur / 2) / dur_param;  // перевод показаний датчика в сантиметры
  delay(100);
  //sm = map(sm, 0, 350, 0, 255);
  return (sm);
}


int dist_filtered() {
  int dist = get_distance();
  float dist_filt = 0;  // отфильтрованный выход
  dist_filt += (dist - dist_filt) * filt_param;
  return (dist_filt);
  Serial.print("Distance is ");
  Serial.print(dist_filt);
  Serial.print(" sm");
}
```

```
int peep_time() {
  int sound = dist_filtered();
  int sound_time = map(sound, 0, 255, 700, 100);
  //tone(NOTEPIN, 1000, sound_time);
  return (sound_time);
}

bool check_button() {
  if (digitalRead (BUTTONPIN) == HIGH) {
    button_state = not button_state;
    current_mode = 0;
    delay(100);
  }
  return button_state;
}

void loop() {
  bool is_on = check_button();

  if (is_on) {
    sens_privod(19);
    ride();
    //program();
  }


  else {
    stop();
  }
}
```

Возможные изменения:

Вследствие использования конструкции корпуса с жестко соединенными осями была выявлена проблема проскальзывания колес при повороте из-за различного типа покрытия, что не позволяет в должной мере реализовывать управление роботом. Уменьшить такую проблему возможно путем замены жестко соединенных осей на две отдельный платформы, с одной осью на каждой, соединенными между собой сервоприводом. Поворот будет осуществляться не изменением скорости или направления вращения колес, а путем поворота одной оси относительно другой (аналог техники-переломки).

В газах (воздухе) скорость распространения звуковой волны увеличивается по мере увеличения температуры. Так, при температуре 10ºС скорость звука равна 337 м/с, а при 40ºС достигает уже 354 м/с. Данное явления объясняется увеличением упругости газов, потому что чем более сильные силы упругости возникают в деформируемой среде, тем выше подвижность частиц и больше степень взаимодействия. Уменьшить погрешность измерения расстояния из-за изменения температуры окружающей среды с помощью ультразвукового дальномера позволить использование дополнительного датчика – термометра. Данные, полученные с термометра, следует использовать в расчете скорости звука при изменяющейся температура по следующей формуле:

$$c = 331.45\sqrt{\frac{T}{273}}, \text{где } T - \text{температура воздуха в Кельвинах}$$

На данном этапе следует обратиться к рисунку 6, на котором изображен график сигнала, полученного с ультразвукового дальномера, до применения фильтрации. Как можно видеть по представленному графику, точность ультразвукового дальномера до фильтрации составляет 5–7%. Если учитывать диапазон рабочих температур для данной робототехнической платформы 10–40 °C, то разность скоростей звука в воздухе составляет

$(354 - 337)/337 = 5.6\%$. Из полученных значений можно сделать вывод о том, что использование датчика температуры не позволит повысить точность получаемого сигнала, но потребует больших вычислительных возможностей, что может сказаться на эффективности работы всей программы.

Данная робототехническая платформа Omegabot оснащена одноплатным компьютером Raspberry Pi 3+, что позволяет использовать техническое зрение для обнаружения объекта или построения пути, а также управлять роботом в режиме прямого телеуправления. Ниже представлен код на языке Python, реализующий получение видео изображения с камеры. Использование технического зрения позволяет более точно определять препятствия и выбирать оптимальные маршруты, чем ультразвуковой датчик.

```
import cv2
import numpy as np

cap = cv2.VideoCapture(0)
fourcc = cv2.VideoWriter_fourcc(*'XVID')
out = cv2.VideoWriter('output.avi', fourcc, 60.0, (300, 240))

while True:
    ret, frame = cap.read()
    frame = cv2.flip(frame, -1)
    #gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    out.write(frame)

    cv2.imshow('video feed', frame)
    #cv2.imshow('gray feed', gray)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

cap.release()
out.release()
cv2.destroyAllWindows()
```

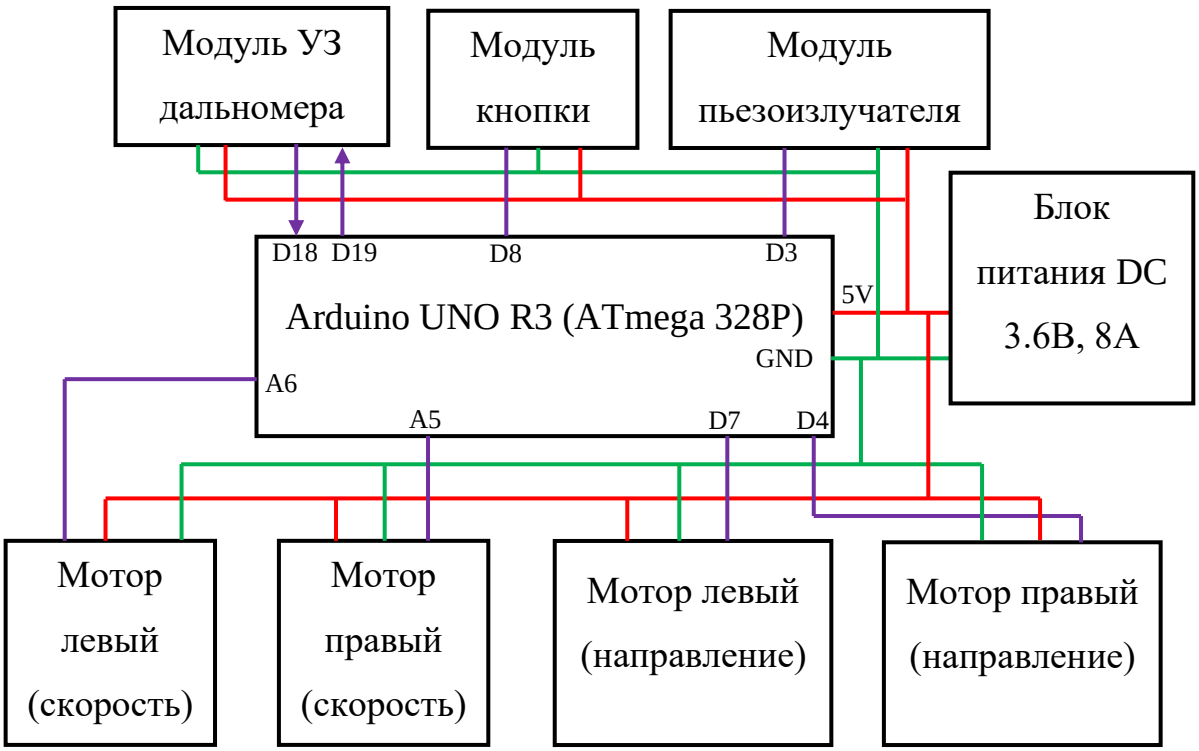
Код, реализующий режим прямого телеуправления представлен ниже:

```
from pynput import keyboard

import time
import smbus
#import keyboard

class SimpleMover:
    def __init__(self, address):
        self.address = address
        self.bus = smbus.SMBus(1)
        time.sleep(1)
```

```
    def writeData(self, value1, value2, valueMode):
        byte3 = int(valueMode).to_bytes(1, byteorder='big')
        byte1 = int(value1).to_bytes(1, byteorder='big')
        byte2 = int(value2).to_bytes(1, byteorder='big')
        self.bus.write_i2c_block_data(self.address, 0, [byte1[0], byte2[0],
byte3[0]])

SM = SimpleMover(0x22)
SM.writeData(128, 128, 0)

def on_release(key):
    if key.char == 'w':
        SM.writeData(128, 128, 0)
    if key.char == 's':
        SM.writeData(128, 128,0)
    if key.char == 'a':
        SM.writeData( 128, 128, 0)
    if key.char == 'd':
        SM.writeData(128, 128, 0)
#
    if key == keyboard.Key.esc:
        return False

with keyboard.Listener(
      on_press = on_press,
      on_release = on_release) as listener:
        listener.join()

listener = keyboard.Listener(
    on_press = on_press,
    on_release = on_release)
listener.start()
```

Список литературы

1. Omegabot. Официальный сайт Omegabot [Электронный ресурс]. URL: https://omegabot.ru/
2. ArduinoMaster [Электронный ресурс]. URL: https://arduinomaster.ru/datchiki-arduino/ultrazvukovoj-dalnomer-hc-sr04/
3. Omegabot: Education. Интернет-магазин Omegabot [Электронный ресурс]. URL: https://omegabot.ru/product/44


Модуль УЗ дальномера
Модуль кнопки
Модуль пьезоизлучателя
Блок питания DC 3.6В, 8А
D18
D19
D8
D3
Arduino UNO R3 (ATmega 328P)
5V
GND
A6
A5
D7
D4
Мотор левый (скорость)
Мотор правый (скорость)
Мотор левый (направление)
Мотор правый (направление)