ДОДАТОК А

Графічний матеріал кваліфікаційної роботи

Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

факультет комп'ютерної інженерії та управління

КВАЛІФІКАЦІЙНА РОБОТА

на тему: Система на платформі Arduino програмного управління переміщеннями колісного роботу на підставі даних від датчику одометру

Студента:
групи СПм-20-1
Мар'янов С.О.

Керівник:
проф. Каргін А.О

2021 рік

МЕТА ТА ЦІЛІ РОБОТИ

Метою атестаційної роботи є є розробка системи програмного управління переміщеннями колісного роботу за допомогою одометрчиних даних.

Для досягнення поставленої мети необхідно вирішити такі завдання:

- проаналізувати предметну область;
- розробити систему керування роботом;
- спроектувати апаратну частину системи.

АКТУАЛЬНІСТЬ РОБОТИ

Зараз у світі спостерігається дуже активний розвиток роботизованих систем і робототехніки загалом. Споживачі таких роботів — логістика та виробництво, а також всі інші майданчики, на яких необхідна автоматизація переміщень (наприклад, торгові та розважальні центри, сільське господарство, банки). Виробництво має бути універсальним, продуктивним та ефективним.

Однією з фундаментальних проблем робототехніки є автономне переміщення. Для ефективного функціонування роботи забезпечені системами сприйняття довкілля, засобами аналізу ситуацій та здійснюють планування руху. При цьому виключно важливою є теоретико-механічна складова задачі, що включає дослідження загальних глобальних властивостей керованої системи.

АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

Менеджер паролів повинен відповідати наступним критеріям:

- відкритий вихідний код;
- достатня точність рухів;
- повна повторюваність маршруту при повторному запуску;
- можливість задання нового маршруту;

5

АНАЛІЗ ІСНУЮЧИХ МЕТОДІВ КЕРУВАННЯ

На даний момент існує декілька основних методів керування колісним роботом:

- за допомогою часових відліків;
- напівавтоматичне керування;
- інтелектуального управління.

Основними недоліками яких є:

- відсутність відкритого вихідного коду;
- недостатня точність рухів ;
- обмежений функціонал безкоштовних версій;
- залежність від попереднього упорядкування середовища;
- необхідний контроль оператора.

6

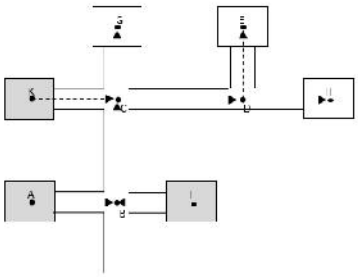
ВИКОРИСТАНІ ТЕХНОЛОГІЇ ТА БІБЛІОТЕКИ

- Arduino UNO
- Motor shield L293D
- Optical encoder H206
- Arduino C

МЕТОД КЕРУВАННЯ

7

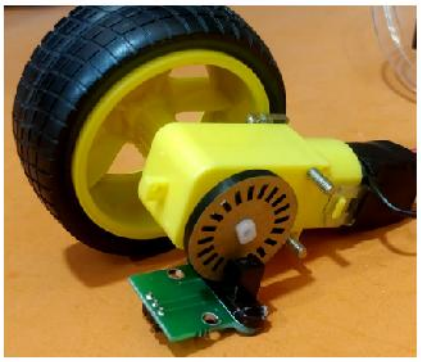
Метод керування базується на заздалегідь побудованому маршруті



АПАРАТНА РЕАЛІЗАЦІЯ

8

В роботі одометр представлений у вигляді інфрачервоного датчику H206 та диском з 20 отворами, що закріплений на осі двигуна. Принцип його роботи заключаєтьмя в тому, що датчик реагує на зміну сигналу при кожному проходженні отворів на диску між зубцями



ВИСНОВКИ

В ході виконання кваліфікаційної роботи були отримані наступні результати:

- проаналізована предметна область;
- проведено огляд і аналіз існуючих методів керування;
- розроблений метод керування;
- реалізований метод керування.

ДОДАТОК Б Вихідний код

Б.1 sketch.ino

```
#define LM_pos 6
#define LM_neg 7
#define RM_pos 8
#define RM_neg 9
#define CS_PIN 10

#include <SD.h>
#include <SPI.h>
#include <EEPROM.h>

File printFile;
String buffer;
boolean SDfound;
int steps_number = 0;
int new_step = 0;
struct Step{
  String direction;
  int distance;
};

Step* route = 0;
String step_direction;
int step_distance;
byte n=4;
float w_length=0.87;
float w_length_n=(w_length/n);
float robot_lenght;
float robot_width;
float turning_length = ((3.14*robot_width*90)/180);

void setup() {
  attachInterrupt(1,gap,RISING);
  Serial.begin(9600);
  Serial.print("Initializing SD card... ");

  if (!SD.begin(CS_PIN)) {
    Serial.println("Card initialization failed!");
    while (true);
  }

  Serial.println("initialization done.");
  printFile = SD.open("route.txt ");

  if (!printFile) {
```

```
    Serial.print("The text file cannot be opened");
    while(1);
  }

  while (printFile.available()) {
    printFile.readStringUntil('\n');
    steps_number++;
  }

  printFile.close();
  printFile = SD.open("route.txt");
  route = new Step [steps_number];
  String direction_temp;
  String distance_temp;
  int index = 0;
  while (printFile.available()) {
    direction_temp = printFile.readStringUntil(',');
    distance_temp = printFile.readStringUntil('\n');
    route[index].direction = direction_temp;
    route[index].distance = distance_temp.toInt();
    index++;
  }

  Serial.begin(9600);
  pinMode(LM_pos, OUTPUT);
  pinMode(LM_neg, OUTPUT);
  pinMode(RM_pos, OUTPUT);
  pinMode(RM_neg, OUTPUT);
  digitalWrite(LM_neg, LOW);
  digitalWrite(RM_neg, LOW);

  attachInterrupt(digitalPinToInterrupt(2), Left_ISR, CHANGE);
  attachInterrupt(digitalPinToInterrupt(3), Right_ISR, CHANGE);
}

void gap() {
  step_distance=step_distance-w_length_n;
}

void loop() {
int i=0;
for (int i=0; i <= steps_number-1; i++)
{
  Serial.println(route[i].direction);
  Serial.println(route[i].distance);
  String step_direction = route[i].direction;
  int step_distance = route[i].distance;

    switch (step_direction) {
    case 'Forward':
      step_distance = route[i].distance - robot_lenght;
      while ( step_distance != 0 )
      {
```

```
    analogWrite(LM_pos, acceleration);
    analogWrite(RM_pos, acceleration);
  }
  break;
case 'Left':
  step_distance=turning_length;
  while ( step_distance != 0 )
  {
    analogWrite(RM_pos, acceleration);
  }
  step_distance = route[i].distance;
  while ( step_distance != 0 )
  {
    analogWrite(LM_pos, acceleration);
    analogWrite(RM_pos, acceleration);
  }
  break;
case 'Right':
  step_distance = robot_width;
  while ( step_distance != 0 )
  {
    analogWrite(LM_pos, acceleration);
    analogWrite(RM_pos, acceleration);
  }
  step_distance=(turning_length*3);
  while ( step_distance != 0 )
  {
    analogWrite(LM_pos, acceleration);
    analogWrite(RM_pos, acceleration);
  }
  step_distance = route[i].distance;
  while ( step_distance != 0 )
  {
    analogWrite(LM_pos, acceleration);
    analogWrite(RM_pos, acceleration);
  }
  break;
case 'Backward':
  step_distance = robot_width;
  while ( step_distance != 0 )
  {
    analogWrite(LM_pos, acceleration);
    analogWrite(RM_pos, acceleration);
  }
  step_distance=(turning_length*3);
  while ( step_distance != 0 )
  {
    analogWrite(RM_pos, acceleration);
  }
  step_distance = robot_width;
  while ( step_distance != 0 )
  {
    analogWrite(LM_pos, acceleration);
```

```
        analogWrite(RM_pos, acceleration);
      }
      break;
      step_distance=(turning_length*3);
      while ( step_distance != 0 )
      {
        analogWrite(RM_pos, acceleration);
      }
      while ( step_distance != 0 )
      {
        analogWrite(LM_pos, acceleration);
        analogWrite(RM_pos, acceleration);
      }
      break;
    default:
      break;
    }
  }
}
```