
Лабораторні роботи 3-4. Розроблення цифрової системи керування на основі платформи Arduino

Мета роботи: побудова алгоритму і програми керування виконавчим пристроєм- двохкоординатним електроприводом з числовим програмним керуванням на основі апаратних, програмних та інструментальних засобів МК платформи Arduino.

Програма роботи

1. Побудова таблиці керуванням переміщеннями.
2. Побудова тимчасових діаграм керуванням переміщенням.
3. Побудова схеми алгоритму керуванням переміщенням.
4. Призначення ліній для вихідних сигналів керування
5. Розробка версії 1 програми (додаток 1) і налагодження з використанням Proteus (модель - 4 channel Relays с лініями керування 8,7,2,4) і віртуальних інструментів. Отримати осцилограми сигналів для $T_w=100.500$ ms). Виконати оцінку необхідних ресурсів.
6. Розробка версії 2 оптимізованої програми (додаток 2) і налагодження аналогічно п.5.
7. Завантаження програми ($T_w=1000$ ms) з п.5 або п.6 в пам'ять керуючого мікроконтролера лабораторного мініплоттера і отримання результату малювання контура.
8. Додатково. Розробка і налагодження версії 3 програми (додаток 3). Оцінка необхідних ресурсів.
9. Висновки по роботі. Оформлення звіту в електронному вигляді з приведенням знімків екрану програм, фотографій і відеофрагментів проведених експериментів.

Теоретичний матеріал

Керування двохкоординатними електроприводами

У сучасному технологічному обладнанні широкого поширення отримали координатні столи і маніпулятори, обладнані різними видами електроприводів (рисунок 2.1).

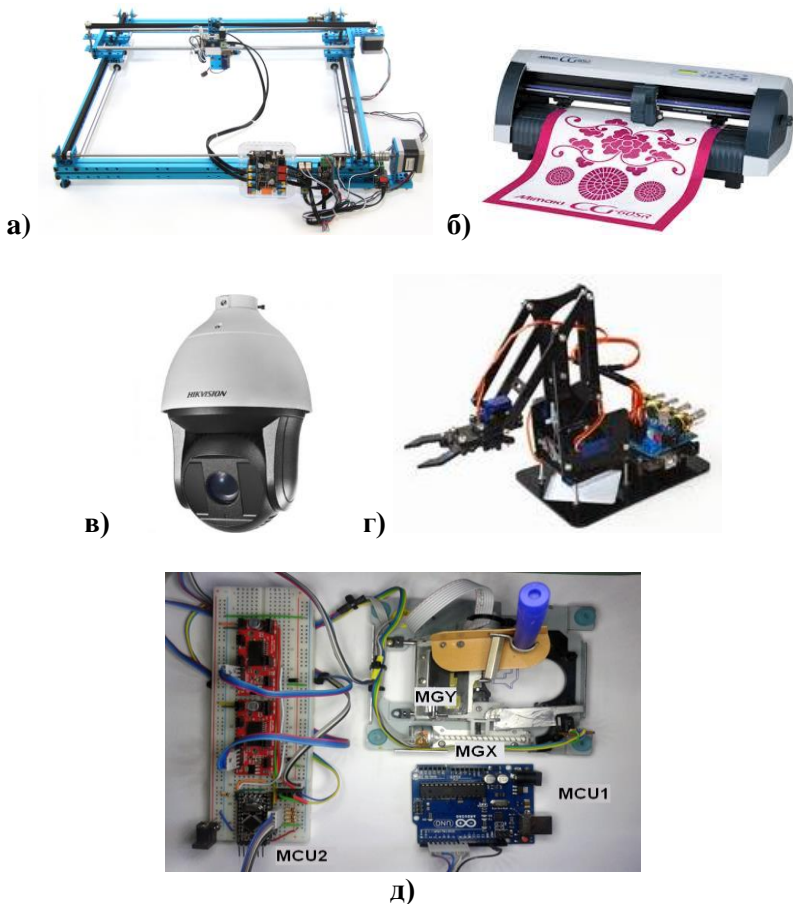


Рисунок 2.1 — Застосування двохкоординатних електроприводів: а) планшетний плоттер XY-Plotter Robot Kit v2.0; б) друкуючий (ріжучий) плоттер; в) маніпулятор; г) роботизована поворотна камера; д) лабораторний мініплоттер

Для керування електроприводами X, Y координатного плоттера (рисунок 2.2г) використовуються силові модулі - драйвери MGX, MGY, що забезпечують необхідні значення струмів і напруги для обертання електродвигунів із заданою швидкістю і в необхідному напрямі. Логічні сигнали керування

даними драйверами формуються керуючим мікроконтролером MCU2.

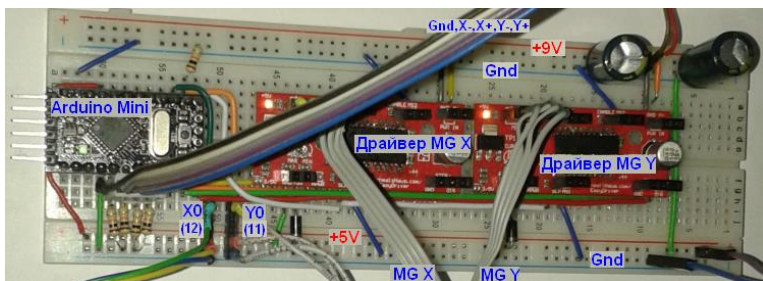


Рисунок 2.2 — Система керування двохкоординатним електроприводом

Мікроконтролер MCU1 керує зміною напрямку переміщення вузла, що пише, по необхідному контуру.

Приклад побудови МК блоку керування двохкоординатним столом

Переміщення двохкоординатного столу здійснюється по координатній сітці з фіксованим кроком. Фрагмент координатної сітки показаний на рисунку 2.3.

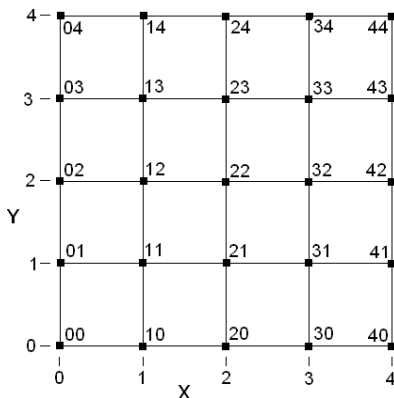


Рисунок 2.3 — Фрагмент координатної сітки

Для виконання технологічних операцій координатний стіл переміщується по деякій траєкторії (рисунок 2.4а), що складається з послідовності кроків з точки 0 через точки 1,2,3,4,5 за напрямками 2,3,1,3,1 з набору на (рисунок 2.4б) ..

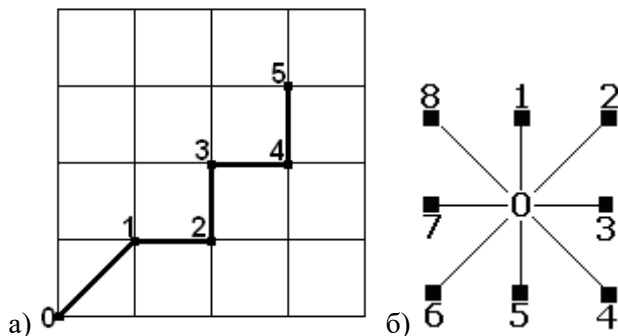


Рисунок 2.4 — Траєкторія руху (а) і вектори напрямів (б).

Сигнали керування (X , X -) блоку керування забезпечують включення/виключення і вибір напрямку обертання двигуна електроприводу X (рисунок 2.4а). Пов'язана з двигуном гвинтова передача забезпечує перетворення обертального руху в поступальний по осі. Аналогічно, сигнали (Y , Y -) забезпечують керування поступальним рухом по осі Y (рисунок 2.5).

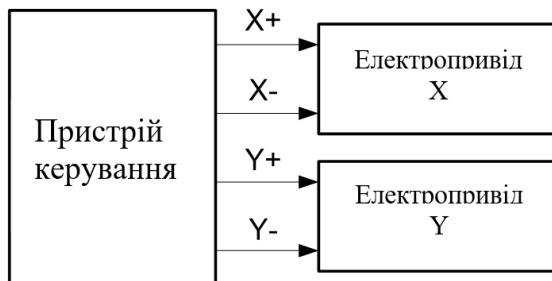


Рисунок 2.5. — Структурна схема системи керування рухом двохкоординатного столу

У таблиці. 2.1 наведені комбінації керуючих сигналів, для усіх напрямів переміщення $Dir = 0,8$.

Таблиця 2.1 - Сигнали керування переміщенням

Dir	X+	X-	Y+	Y-
0	0	0	0	0
1	0	0	1	0
2	1	0	1	0
3	1	0	0	0
4	1	0	0	1
5	0	0	0	1
6	0	1	0	1
7	0	1	0	0
8	0	1	1	0

Тепер можна формалізувати опис роботи пристрою керування координатного столу по вузлах (XY) координатної сітки з урахуванням векторів переміщень (Dir) на кожному кроці. У початковому стані 0 приводи зупинені (комбінація 0000). Для кожного кроку 0..6 в таблиці 2.2 заносимо комбінації сигналів (X, X-, Y, Y-) і тривалість кроку (T). Після завершення переміщень в стані 6 приводи зупиняються (комбінація 0000).

Таблиця 2.2 - Сигнали керування переміщеннями

№	Dir	X+	X-	Y+	Y-	T, сек
0	0	0	0	0	0	1
1	2	1	0	1	0	1
2	3	1	0	0	0	1
3	1	0	0	1	0	1
4	3	1	0	0	0	1
5	1	0	0	1	0	1
6	0	0	0	0	0	1

У таблиці 2.2 виділені значення, які змінюються в поточному кроці переміщення. Інші зберігають попередні значення. З таблиці

отримуємо тимчасові діаграми сигналів керування (рисунок 2.6) для даного прикладу.

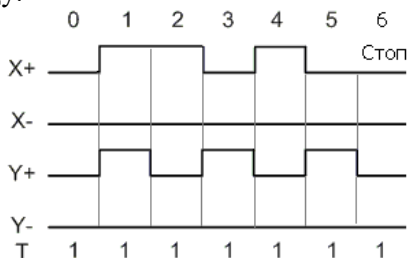


Рисунок 2.6. - Тимчасові діаграми для сигналів керування переміщенням.

Пристрій керування є програмованим і його функціонування окрім тимчасових діаграм описується схемою алгоритму (рисунок 2.7).

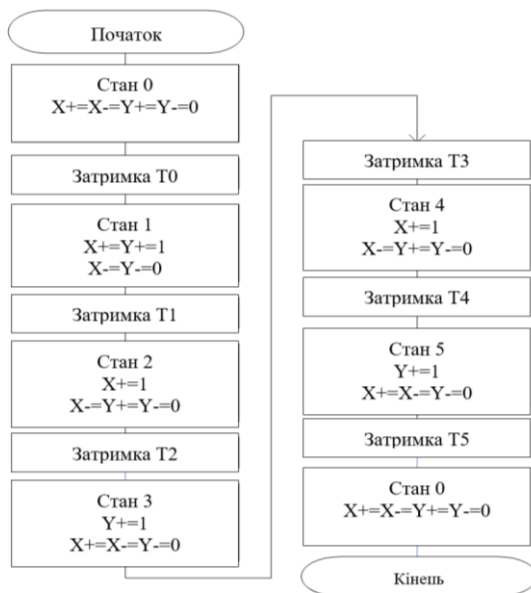


Рисунок 2.7 - Алгоритм керування переміщеннями

Число операцій в алгоритмі можна зменшити, якщо виключити дії, що повторюються. Якщо на черговому кроці стан лінії керування не змінюється, то операція підтвердження стану не потрібна. Таким чином можна отримати оптимізовану схему алгоритму (рисунок 2.8).

Пристрій керування може бути побудований на основі оригінальної МК платформи Arduino UNO (рисунок 2.9а) або одного з клонів Arduino UNO (рисунок 2.9б) на основі МК ATmega328 в різних корпусах. Умовно графічне позначення Arduino UNO наведено на рисунку 2.9в.

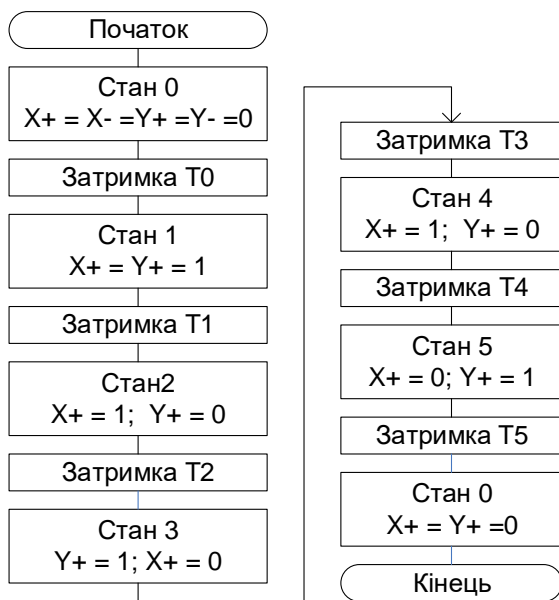
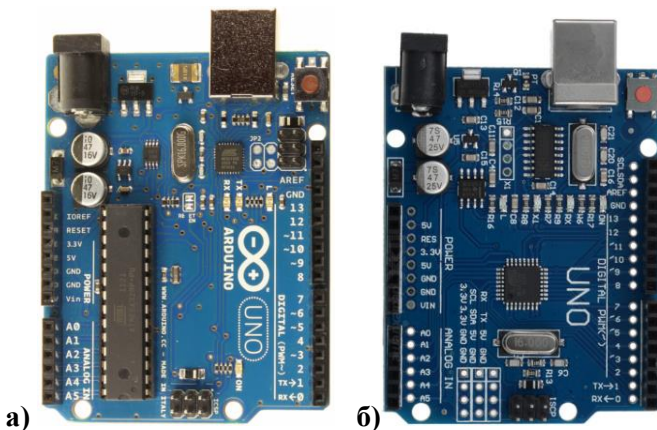


Рисунок 2.8 - Алгоритм керування переміщеннями

Пристрій керування може бути побудований на основі оригінальної МК платформи Arduino UNO (Рисунок 8а) або одного з клонів Arduino UNO (Рисунок 8б) на основі МК ATmega328 в різних корпусах. Умовно графічне позначення Arduino UNO наведено на рисунку 2.9в.



	POWER	MCU	Gnd	
	RESET		D13	
	+3.3V		D12	
	+5V		~D11	
	GND		~D10	
	GND		~D9	
	Vin		D8	
	Analogs		D7	
	A0		~D6	
	A1		~D5	
	A2		D4	
	A3		~D3	
	A4		D2	
	A5		D1	
			D0	

в)

Рисунок 2.9. — Вид Arduino UNO (а, б, в) та умовно-графічне позначення

Структурна схема даного МК пристрою приведена на рисунку 2.10 і містить: 1 - елементи живлення (від USB або зовнішнього джерела); 2 - перетворювач USB - UART для самопрограмування через BOOT- завантажувач і інформаційного обміну МК; 3,4 - роз'єми для підключення зовнішніх пристроїв і сигналів; 5 - МК.

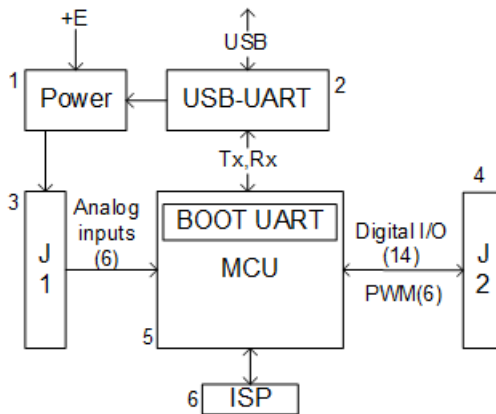


Рисунок 2.10. - Структурна схема МК пристрою Arduino

Детальну інформацію про пристрій можна знайти в літературних джерелах [1] і на численних ресурсах Internet.

Оскільки у розпорядженні користувача є 14 ліній цифрового введення-виведення, які позначаються D0...13 або просто 0...13, необхідно вибрати лінії для вихідних сигналів керування. Можуть бути призначені довільні лінії - 8,7,2,4 (як лінії керування Shield 4-канального реле) або 5,4,3,2 (рисунок 2.11).



Рисунок 2.11. — Призначення ліній для вихідних сигналів керування Arduino UNO

Середовище програмування Arduino

Інтегроване середовище програмування (IDE Arduino) встановлюється з [1], має дуже простий вигляд і містить - вікно редактора коду, елементи керування і вікно повідомлень (рисунок 2.12).

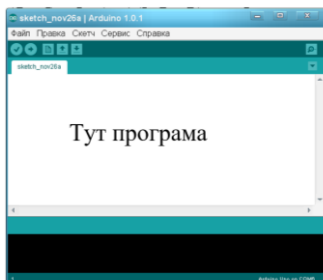


Рисунок 2.12 - Робоче вікно середовища програмування.

За допомогою елементів керування робочого вікна відбувається запуск основних операцій в середовищі програмування Arduino.



- порівняння вмісту Flash- пам'яті з результатом компіляції програми.



- запис програми в програмну пам'ять (Flash) МК.



- початок роботи з новою програмою.



- продовження роботи з існуючою програмою.



збереження

поточної програми.

При створенні програми визначаються змінні і оператори середовища програмування Arduino [1,2,5,6], необхідні для реалізації алгоритму функціонування МК :

```
int xp = 5; // X+
int xm = 4; // X-
int yp = 3; // Y+
int ym = 2; // Y-
```

Лінії, що використовуються для керування налаштовуються як вихідні:

```
pinMode(xp, OUTPUT);
pinMode(xm, OUTPUT);
pinMode(yp, OUTPUT);
pinMode(ym, OUTPUT);
```

Установка різних значень вихідних сигналів виконується за допомогою операцій із таблиці. 2.3.

Таблиця 2.3 - Базові оператори програми

Операція	Оператор IDE Arduino
X+ = 0	digitalWrite(xp, LOW)
X- = 0	digitalWrite(xm, LOW)
Y+ = 0	digitalWrite(yp, LOW)
Y- = 0	digitalWrite(ym, LOW)
X+ = 1	digitalWrite(xp, HIGH)
X- = 1	digitalWrite(xm, HIGH)
Y+ = 1	digitalWrite(yp, HIGH)
Y- = 1	digitalWrite(ym, HIGH)

З наведених раніше описів і операцій відповідно до алгоритму роботи (рисунок 2.7) створюється текст програми (додаток 1). В результаті компіляції програми отримуємо повідомлення про розмір коду для завантаження в пам'ять МК (рисунок 2.13).

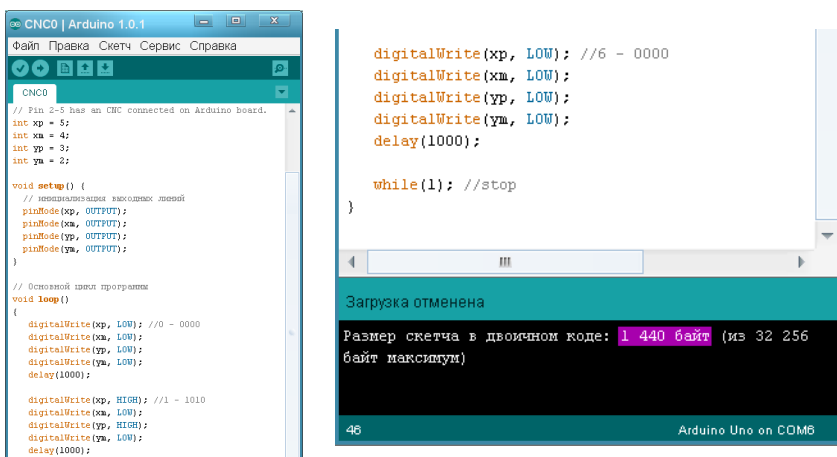


Рисунок 2.13 - Вид програми і результат компіляції

У разі помилок, виводяться повідомлення про її місце і характер. Помилки виправляються і дії повторюються.

Будь-яка програма в процесі створення проходить етап тестування з метою виявлення можливих логічних помилок при написанні програми, або допущених в початкових даних. Ці помилки не виявляються компілятором, але проявляються в програмних моделях МК - симуляторах.

Налагодження програми в симуляторі Arduino

Симулятор дозволяє виконати покрокове налагодження програми [5]. У робочому вікні симулятора відображається текст програми, значення змінних і сигналів МК пристрою (рисунок 2.13).

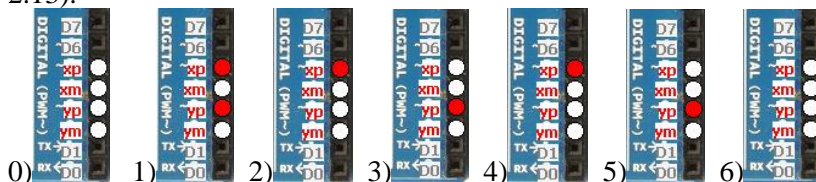
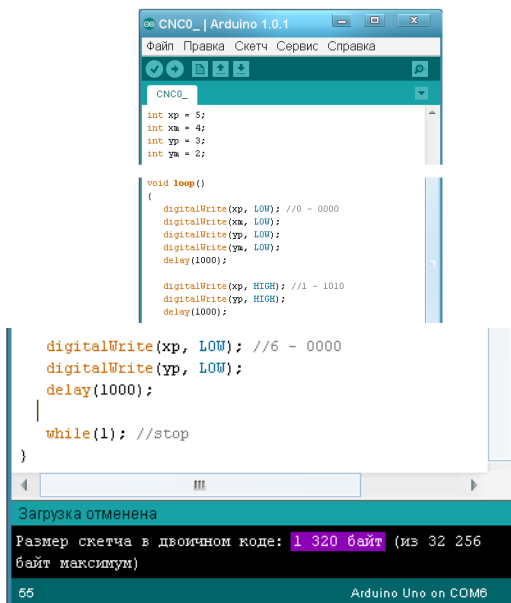


Рисунок 2.13 - Результат покрокового налагодження програми з використанням симулятора Arduino

Наведена програма для алгоритму на рисунку 2.7 надмірна, оскільки включає ряд операцій, що не змінюють стану керуючих сигналів і займає 1440 байт пам'яті програм. Проста оптимізація програми за рахунок виключення надмірних операторів (рисунок 2.8) дає економію 120 байт пам'яті програм (рисунок 2.15).



```
CNC0_ | Arduino 1.0.1
Файл Правка Скетч Сервис Справка

CNC0_
int xp = 5;
int xm = 4;
int yp = 3;
int ym = 2;

void loop()
{
    digitalWrite(xp, LOW); //0 - 0000
    digitalWrite(xm, LOW);
    digitalWrite(yp, LOW);
    digitalWrite(ym, LOW);
    delay(1000);

    digitalWrite(xp, HIGH); //1 - 1010
    digitalWrite(yp, HIGH);
    delay(1000);

    digitalWrite(xp, LOW); //6 - 0000
    digitalWrite(yp, LOW);
    delay(1000);

    while(1); //stop
}

Загрузка отменена
Размер скетча в двоичном коде: 1 320 байт (из 32 256 байт максимум)
55 Arduino Uno on COM6
```

Рисунок 2.15 - Оптимізована програма

За допомогою симулятора можна переконатися, що функціонування пристрою залишилося без змін.

Модифікація програми із спрощеним завданням початкових даних траєкторії руху

Вектори переміщення 0...8 (рисунок 2.46) можна представити у вигляді 36-елементного неупакованого масиву, де кожному вектору відводиться 4 елементи:

dir[36]={0,0,0,0, 0,0,1,0, 1,0,1,0, 1,0,0,0, 1,0,0,1, 0,0,0,1, 0,1,0,1, 0,1,0,0, 0,1,1,0},

Послідовність номерів векторів переміщень з таблиці 2.2 – масивом:

`cmd[7]={0,2,3,1,3,1,0},`

тривалість в мс переміщення на кожному кроці – масивом:

`time[7]={1000,1000,1000,1000,1000,1000,1000};`

Це дозволяє реалізувати $N = 7$ – крокове (включаючи початковий і кінцевий стани 0000) переміщення по заданій траєкторії за наявності функції переміщення на поточному кроці - `void move (byte num)`, в якій встановлюються керуючі сигнали з масиву `dir[36]`, згідно із заданим вектором `num`. Крок $0 \leq i < N$ реалізується таким чином:

`move (cmd[i]);`

`delay (time[i]);`

Для зміни алгоритму функціонування необхідно задати значення N і елементи масивів `cmd[]`, `time[]`. У додатку 3 представлений скетч, де реалізовано керування для початкового прикладу.

Налагодження програми в середовищі Proteus

Середовище Proteus [6] дає можливість побудувати схему МК пристрою (рисунок 2.16), зв'язати МК з hex- файлом вмісту пам'яті програм і виконати моделювання роботи пристрою за заданою програмою.

Як отримати .hex з Arduino?

Запускаємо Arduino IDE, відкриваємо скетч, натискаємо Verify. Відкриваємо провідник, пишемо там `%temp%\` і натискаємо Enter. Знаходимо там папки з іменами `buildXXXXXXXXXXXXXXXXXX.tmp`. Якщо програма- `CNC1.ino`, то в папці слід знайти файл `CNC1.cpp.hex` - це і є результат компіляції `CNC1.ino`.

`CNC1.cpp.hex` можна записати в пам'ять МК плати Arduino, або використовувати при моделюванні в Proteus.

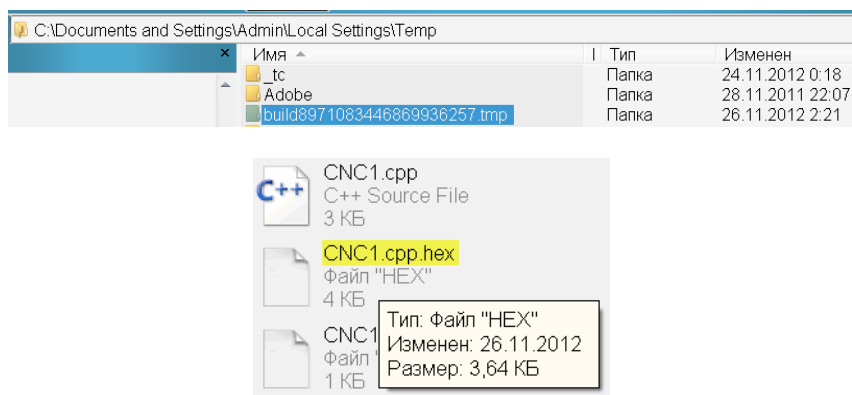


Рисунок 2.16 — Пошук “hex” –файла

У Proteus використовуються віртуальні елементи керування і виконавчі пристрої (рисунок 2.17), а тимчасові діаграми вихідних сигналів можуть бути зареєстровані за допомогою 4-канального осцилографа або логічного аналізатора (рисунок 2.17). Використання демонстраційної версії Proteus вносить обмеження:

- для моделювання можна використати тільки готові приклади з Sample;
- схема може бути змінена, але без можливості збереження;
- до ліній схеми можна підключати осцилограф, пробники і проводити моделювання, але зберегти зміни не можна.

Для моделювання пристрою керування зручно використати приклад керування 4-канальним реле з використанням цифрових виводів 8,7,2,4 (рисунок 2.17). Світлодіоди відображають стани виходів.

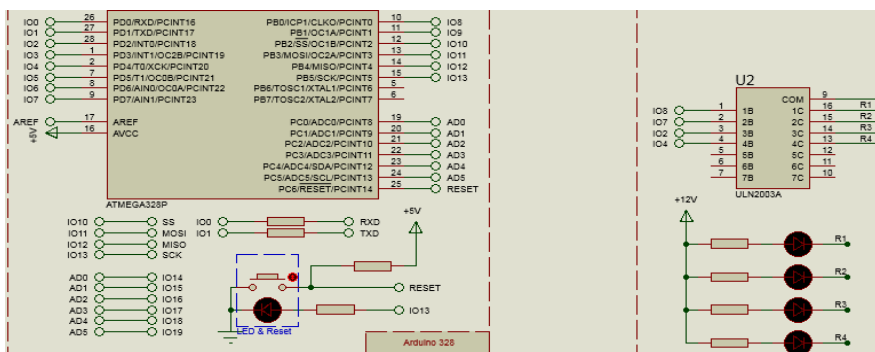


Рисунок 2.17 —. Моделювання пристрою керування електроприводами в Proteus

Для спостереження тимчасових діаграм зміни сигналів керування підключається 4-канальний осцилограф (рисунок 2.18).

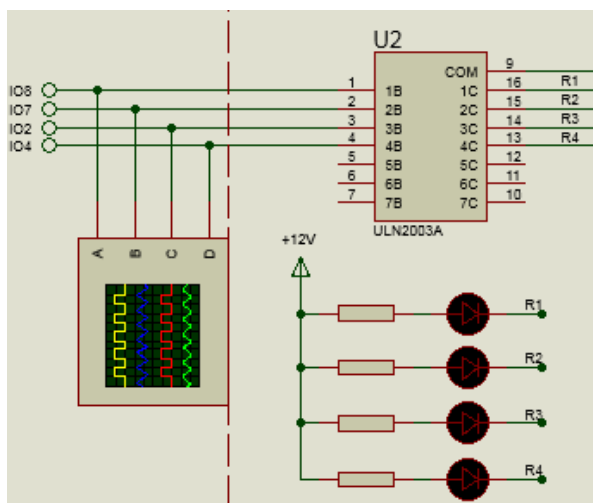


Рисунок 2.18 —. Використання віртуального осцилографа (A – X+, B – X-, C – Y+, D – Y-)

Якщо осцилограми сигналів керування за виглядом співпадають з початковими тимчасовими діаграмами (рисунок 2.19), то це служить підтвердженням правильного формування

керуючих сигналів МК пристрою, працюючого за розробленою програмою в середовищі Arduino.

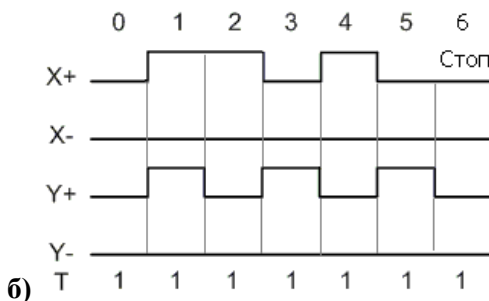
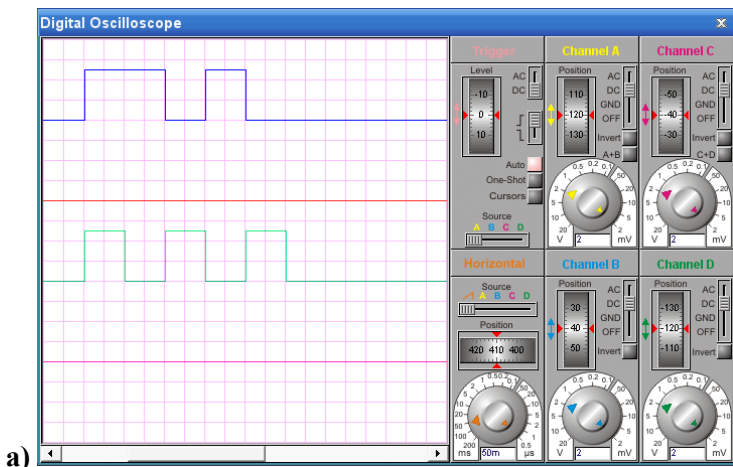


Рисунок 2.19 —. Осцилограми сигналів керування за результатами моделювання (а) і вихідні діаграми (б).

Завантаження програми в пам'ять МК

Плата Arduino, яка виконуватиме керування мініплоттером кабелем USB підключається до вільного порту USB комп'ютера, визначається виділений номер віртуального порту і вказується в налаштуваннях IDE, вибирається тип плати, використовуваний МК і запускається завантаження. Ці дії детально описані в [3].

Після завантаження програми, плата підключається до ліній керування мініплоттером (рисунок 2.1, 2.2). Після скидання мініплоттер під керуванням плати Arduino малює контур який повинен співпадати із заданим (рисунок 2.20).

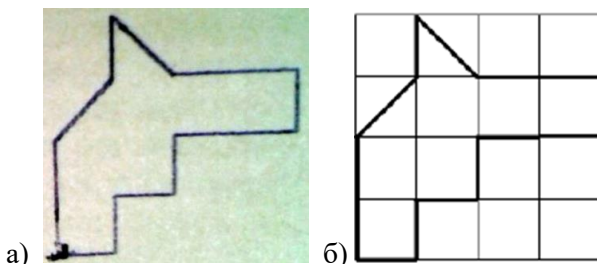


Рисунок 2.20 — Приклад результату роботи мініплоттера (а) під керуванням розробленої програми для завдання (б).

Зміст звіту

1. Схеми апаратних засобів для вирішення завдань.
2. Тексти програм для завдань.
3. Результати моделювання.
4. Результати виконання експериментів.
5. Висновки по роботі.

Література

1. Среда разработки Arduino. [Електронний ресурс]. – Режим доступу: <http://www.arduino.cc/en/Main/Software>
2. Программирование Ардуино. [Електронний ресурс]. – Режим доступу: <http://arduino.ru/Reference>
3. Соммер У. Программирование микроконтроллерных плат Arduino/Freeduino.- СПб.: БХВ – Петербург, 2012.- 256с.
4. Arduino блокнот программиста_Brian W. Evans.- 2007.- 40с.
5. Simulator for Arduino. [Електронний ресурс]. – Режим доступу: <http://virtronics.com.au/Data/SetupFree.zip>.
6. PROTEUS VSM. Среда виртуального моделирования. [Електронний ресурс]. – Режим доступу: <http://proteus123.narod.ru/PROTEUS-d.pdf>

Додаток 1. Версія 1 програми

/* Управління переміщенням за допомогою сигналів xp (X+),
xm (X-), yp (Y+), ym (Y-)
по послідовності N точок. Тривалість кожного кроку-
1000 мс. */

// Pin 2-5 has an CNC connected on Arduino board.

```
int xp = 5;  
int xm = 4;  
int yp = 3;  
int ym = 2;  
int Tw = 1000;  
void setup() {  
    // ініціалізація вихідних ліній  
    pinMode(xp, OUTPUT);  
    pinMode(xm, OUTPUT);  
    pinMode(yp, OUTPUT);  
    pinMode(ym, OUTPUT);  
}
```

```
void loop()      // Основний цикл програми v1  
{  
    digitalWrite(xp, LOW); //0 - 0000  
    digitalWrite(xm, LOW);  
    digitalWrite(yp, LOW);  
    digitalWrite(ym, LOW);  
    delay(Tw);  
  
    digitalWrite(xp, HIGH); //1 - 1010  
    digitalWrite(xm, LOW);  
    digitalWrite(yp, HIGH);  
    digitalWrite(ym, LOW);  
    delay(Tw);  
  
    digitalWrite(xp, HIGH); //2 - 1000  
    digitalWrite(xm, LOW);  
    digitalWrite(yp, LOW);
```

```
digitalWrite(ym, LOW);
delay(Tw);

digitalWrite(xp, LOW); //3 - 0010
digitalWrite(xm, LOW);
digitalWrite(yp, HIGH);
digitalWrite(ym, LOW);
delay(Tw);

digitalWrite(xp, HIGH); //4 - 1000
digitalWrite(xm, LOW);
digitalWrite(yp, LOW);
digitalWrite(ym, LOW);
delay(Tw);

digitalWrite(xp, LOW); //5 - 0010
digitalWrite(xm, LOW);
digitalWrite(yp, HIGH);
digitalWrite(ym, LOW);
delay(Tw);

digitalWrite(xp, LOW); //6 - 0000
digitalWrite(xm, LOW);
digitalWrite(yp, LOW);
digitalWrite(ym, LOW);
delay(Tw);

while(1)}; //stop
```

Додаток 2. Версія 2 програми (основний цикл)

```
void loop()          // Основний цикл програми
{
    digitalWrite(xp, LOW); //0 - 0000
    digitalWrite(xm, LOW);
    digitalWrite(yp, LOW);
    digitalWrite(ym, LOW);
    delay(Tw);

    digitalWrite(xp, HIGH); //1 - 1010
```

```
digitalWrite(yp, HIGH);
delay(Tw);

digitalWrite(xp, HIGH); //2 - 1000
digitalWrite(yp, LOW);
delay(Tw);

digitalWrite(xp, LOW); //3 - 0010
digitalWrite(yp, HIGH);
delay(Tw);

digitalWrite(xp, HIGH); //4 - 1000
digitalWrite(yp, LOW);
delay(Tw);

digitalWrite(xp, LOW); //5 - 0010
digitalWrite(yp, HIGH);
delay(Tw);

digitalWrite(xp, LOW); //6 - 0000
digitalWrite(yp, LOW);
delay(Tw);

while(1) } ; //stop
```

Додаток 3. Версія 3 програми (Low code)

/* Керування переміщенням за допомогою сигналів xp (X), xm (X-), yp (Y), ym (Y-) по послідовності N точок. Напрямок переміщення на кожному кроці задається значеннями масиву cmd[], а керування переміщенням у будь-якому напрямі - відповідним набором сигналів з масиву dir[]. Напрями мають номери:

```
8 1 2
7 0 3
6 5 4
```

Початкове положення кожного кроку - 0 (Стоп). Тривалість кожного кроку (мс) задається в масиві time[]:

```

byte dir[36]={0,0,0,0, 0,0,1,0, 1,0,1,0, 1,0,0,0, 1,0,0,1,
              0,0,0,1, 0,1,0,1, 0,1,0,0, 0,1,1,0};
byte cmd[]={0,2,3,1,3,1,0};
unsigned int time[7]={100,100,100,100,100,100,100};
byte j, N=7;

// Pin 2-5 has an CNC connected on Arduino board.
int xp = 5;
int xm = 4;
int yp = 3;
int ym = 2;

void setup() {
    // initialize the digital pin as an output.
    pinMode(xp, OUTPUT);
    pinMode(xm, OUTPUT);
    pinMode(yp, OUTPUT);
    pinMode(ym, OUTPUT);
}

void move(byte num){
    j=0;
    if(num<9) j=num*4;
    digitalWrite(xp, dir[j]); j++;
    digitalWrite(xm, dir[j]); j++;
    digitalWrite(yp, dir[j]); j++;
    digitalWrite(ym, dir[j]); }

void loop() // Основний цикл програми v3
{
    for(int i=0;i<N;i++){
        move(cmd[i]);
        delay(time[i]);
    }
    while(1); //stop
}

```

Додаток 4. Варіанти завдань

