НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»
ім. Ігоря Сікорського

Навчально-науковий комплекс «Інститут прикладного системного аналізу»
(повна назва інституту/факультету)

Кафедра Системного проектування
(повна назва кафедри)

«До захисту допущено»
Завідувач кафедри
_____ А.І.Петренко
(підпис) (ініціали, прізвище)
“___”_____20__ р.

Дипломна робота

на здобуття ступеня бакалавра

з напряму підготовки 6.050101 Комп’ютерні науки
(код і назва)

на тему: «Розробка "розумного" модуля до 3Д принтеру для автоматизації прототипування електронних пристроїв.»

Виконав (-ла): студент (-ка) IV курсу, групи ДА-32
(шифр групи)

Колінько Анжеліка Михайлівна _____
(прізвище, ім’я, по батькові) (підпис)

Керівник доцент Кирюша Б. А. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант економічний к.е.н. Рощина Н.В. _____
(назва розділу) (посада, вчене звання, науковий ступінь, прізвище, ініціали) (підпис)

Рецензент _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Нормоконтроль старший викладач Британ О.А. _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2017 року

Національний технічний університет України
«Київський політехнічний інститут»
ім. Ігоря Сікорського

Інститут (факультет) ННК «Інститут прикладного системного аналізу
(повна назва)

Кафедра Системного проектування
(повна назва)

Рівень вищої освіти – перший (бакалаврський)

Напрям підготовки 6.050101 Комп'ютерні науки
(код і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ А.І.Петренко
(підпис) (ініціали, прізвище)
«___»_____20__ р.

ЗАВДАННЯ
на дипломну роботу студенту
Колінько Анжеліці Михайлівні
(прізвище, ім'я, по батькові)

1. Тема роботи «Розробка "розумного" модуля до 3D принтеру для автоматизації прототипування електронних пристроїв»,
керівник роботи доцент Кирюша Богдан Анатолійович,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «10» травня 2017 р. № 1477-с

2. Термін подання студентом роботи 15.06.2017

3. Вихідні дані до роботи:

1. Три осі переміщення з відповідними кроковими двигунами.
2. Для захоплення та позиціонування деталі – насос і кроковий двигун.
3. Для захоплення зображення – камера Raspberry Pi Camera Module.
4. Керування кроковими двигунами мікроконтролером Arduino Mega 2560.
5. Бібліотека для розпізнавання деталей – OpenCV.

4. Зміст роботи:

1. Проаналізувати існуючі аналоги автоматичних модулів та технології, які використовуються для розміщення та пайки радіодеталей.
2. Розробити апаратну частину модулю.
3. Розробити програмну частину модулю.
4. Визначити можливості та способи подальшого покращення чи доробки модулю.

5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо):

1. Загальна схема модулю – плакат.
2. Зображення з накладеною сіткою – плакат.
3. Інтерфейс програми – плакат.

6. Консультанти розділів роботи*

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання 01.02.2017

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1	Отримання завдання	01.02.2017	
2	Збір інформації	15.02.2017	
3	Аналіз існуючих рішень	28.02.2017	
4	Вибір варіанту взаємодії елементів модуля та передачі даних між ними	10.03.2017	
5	Розробка апаратної частини модулю	15.03.2017	
6	Розробка програмної частину модулю	25.03.2017	
7	Визначення переваг і недоліків обраного рішення та способів покращення модулю	25.04.2017	
8	Оформлення дипломної роботи	31.05.2017	
9	Отримання допуску до захисту та подача роботи в ДЕК	15.06.2017	

Студент _____ Колінько А. М.
(підпис) (ініціали, прізвище)

Керівник роботи _____ Кирюша Б. А.
(підпис) (ініціали, прізвище)

АНОТАЦІЯ

бакалаврської дипломної роботи Колінько Анжеліки Михайлівни

на тему «Розробка "розумного" модуля до 3D принтеру для автоматизації прототипування електронних пристроїв»

Дана дипломна робота присвячена розробці модулю для прототипування електронних пристроїв шляхом розпізнавання деталей на робочій площадці та їх переміщення. Приведено огляд та класифікація існуючих аналогів.

Досліджено способи розстановки електронних компонентів на друкованій платі, види 3D принтерів та сучасні методи обробки зображення. Розглянуто алгоритм виділення контурів за допомогою детектора кордонів Кенні (Canny).

Розроблено програму для керування переміщенням по осям. Також було розроблено алгоритм та архітектуру програми розпізнавання кордонів деталей з визначенням найближчої до робочого захвату маніпулятору. Наведені рекомендації по покращенню та доопрацюванню модулю.

Загальний обсяг роботи: 83 сторінки, 31 рисунків, 11 таблиць, 15 посилань.

Ключові слова: кроковий двигун, OpenCV, розпізнавання кордонів, 3D принтер.

АННОТАЦИЯ

бакалаврской дипломной работы Колинько Анжелики Михайловны

на тему «Разработка" умного "модуля к 3D принтеру для автоматизации прототипирования электронных устройств»

Данная дипломная работа посвящена разработке модуля для прототипирования электронных устройств путем распознавания деталей на рабочей площадке и их перемещения. Приведены обзор и классификация существующих аналогов.

Исследованы способы расстановки электронных компонентов на печатной плате, виды 3D принтеров и современные методы обработки изображения. Рассмотрен алгоритм выделения контуров с помощью детектора границ Кенни (Canny).

Разработана программа для управления перемещением по осям. Также был разработан алгоритм и архитектура программы распознавания границ деталей с определением ближайшей к рабочему носику. Приведены рекомендации по улучшению и доработке модуля.

Общий объем работы 83 страниц, 31 рисунков, 11 таблиц, 15 ссылок.

Ключевые слова: шаговый двигатель, OpenCV, распознавание границ, 3D принтер.

ANNOTATION

on Kolinko Anzhelika bachelor's degree

Entitled "Developing" smart "module to 3D printers for automatization of prototyping electronic devices"

This thesis is dedicated to the development of modules for prototyping electronic devices by recognizing boundaries of electronic components at the working field and moving them. An overviewed a review and classification of existing analogues.

Investigated ways of placing electronic components on the circuit board, types of 3D printers and advanced image recognition techniques. The algorithm of edge detection using Canny borders detector was examined.

A program to control movement along the axis was developed. It was also developed the algorithm and program architecture for details recognition and defining the nearest one to work spout. Recommendations to improve and refine the module were provided.

Total volume of work: 83 pages, 31 illustrations, 11 tables, 15 references.

Keywords: stepping motor, OpenCV, border recognition, 3D printer.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

OpenCV – бібліотека комп’ютерного зору з відкритим кодом

Кроковий двигун – електричний двигун, в якому імпульсне живлення електричним струмом призводить до того, що його ротор не обертається неперервно, а виконує щоразу обертальний рух на заданий кут.

Контур – зовнішня границя об’єкту.

3D принтер – одна з форм технологій адитивного виробництва, де тривимірний об'єкт створюється шляхом накладання послідовних шарів матеріалу (друку, вирощування) за даними цифрової моделі.

ПП – програмний продукт.

ВСТУП

В умовах тенденції зменшення розмірів електронних компонентів головним напрямком при виробництві електронних модулів залишається зниження собівартості складання і монтажу друкованих плат при підтримці стабільно високого рівня якості. Операція встановлення компонентів на друковану плату багато в чому визначає економічність і продуктивність цього процесу. Автоматичні системи для складання електронних модулів у все більшій мірі орієнтуються на програмне забезпечення. Це комп'ютеризована техніка, керована потужними контролерами, здатними обробити великий обсяг інформації в реальному часі, з широким спектром функцій. Безумовно, як механічні, так і програмні функції обладнання стають більш складними, але завдання полягає в тому, щоб забезпечити навіть більш просте управління як окремою машиною, так і комплексною лінією на рівні оператора.

Актуальність.

Тема дипломної роботи є актуальною в зв’язку з все більшим поширенням та розвитком автоматизації та мінімізацією радіоелектронних компонентів. Ручне встановлення та пайка електронних компонентів – доволі рутинна та трудомістка задача, а більшість модулів для автоматизації цього процесу орієнтовані на серійне виробництво.

Мета.

Метою даної роботи є розробка модулю до 3D принтеру для захвату та позиціонування радіоелектронних деталей. Розробка даного модулю включає розробку програмного модулю для керування кроковими двигунами для переміщення робочих площадок по осям та повороту (позиціонування) захопленої деталі, регулярний в часі захват зображення з робочої площадки, розпізнавання на отриманих зображеннях границь елементів, захоплення, переміщення та позиціонування захопленої деталі.

Основні завдання.

До основних завдань відносяться:

- аналіз існуючих аналогів автоматичних модулів та технології, які використовуються для розміщення та пайки радіодеталей.
- визначення взаємодії елементів модуля та передачу даних між ними.
- розробка апаратної частини модулю.
- розробка програмної частину модулю.
- аналіз переваги і недоліки обраного рішення.
- визначення можливостей подальшого покращення чи доробки модулю.

Об’єкт дослідження – програмна і апаратна частина модулю для автоматизації прототипування електронних пристроїв.

Предмет дослідження – напівавтоматичні пристрої для розташування електронних деталей на друкованих платах.

1. НЕОБХІДНІ ТЕОРИТИЧНІ ВІДОМОСТІ

1.1 Встановлення компонентів на друкованій платі

Головним напрямком при виробництві електронних модулів залишається зниження собівартості збірки і монтажу друкованих плат при підтримці стабільно високого рівня якості. Операція встановлення компонентів на друковану плату багато в чому визначає економічність і продуктивність. Автоматичні системи для складання електронних модулів у все більшій мірі орієнтуються на програмне забезпечення. Це комп'ютеризована техніка, керована потужними контролерами, здатними обробити великий обсяг інформації в реальному часі, з широким спектром функцій. Безумовно, як механічні, так і програмні функції обладнання стають більш складними, але завдання полягає в тому, щоб забезпечити навіть більш просте управління як окремої машиною, так і комплексної лінією на рівні оператора.

Виробництво друкованих плат на стадії монтажних операцій зазвичай включає в себе наступні основні етапи:

- підготовка компонентів і матеріалів;
- нанесення паяльної пасти;
- установка компонентів;
- припаювання елементів;

Нанесення пасти на контактні площадки виконується дозатором при відпрацюванні макетного зразка плати, а при серійному виготовленні модулів використовується трафарет спільно з оснащенням. Трафарет виготовляється з металевої фольги, що має товщину від 0,075 до 0,2 мм з отворами прямокутної, трапецієподібної або круглої форми. може бути виготовлений з різних матеріалів: нержавіючої сталі, нікелю, бронзи. Часто для виготовлення трафарету застосовується сталь (рис. 1.1).

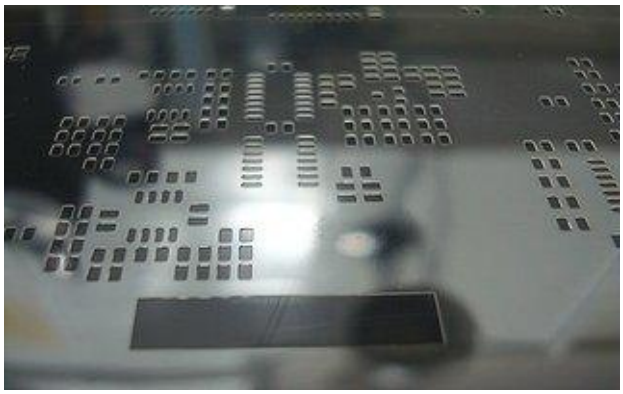


Рис. 1.1 – Сталевий трафарет

Отвори роблять за допомогою травлення, вирізаються лазером або за допомогою гальванопластики.

Після перенесення пасти на групову плату на неї встановлюються компоненти і виконується розплавлення пасти.

Встановлення планарних компонентів на плату з нанесеною пастою може виконуватися вручну або за допомогою засобів автоматизації. При ручній установці неминучі помилки в номіналах компонентів. Неможливо забезпечити вірний та однаковий притиск компонентів до пасти. Щоб не допустити помилок при збірці модулів застосовують різну ступінь автоматизації.

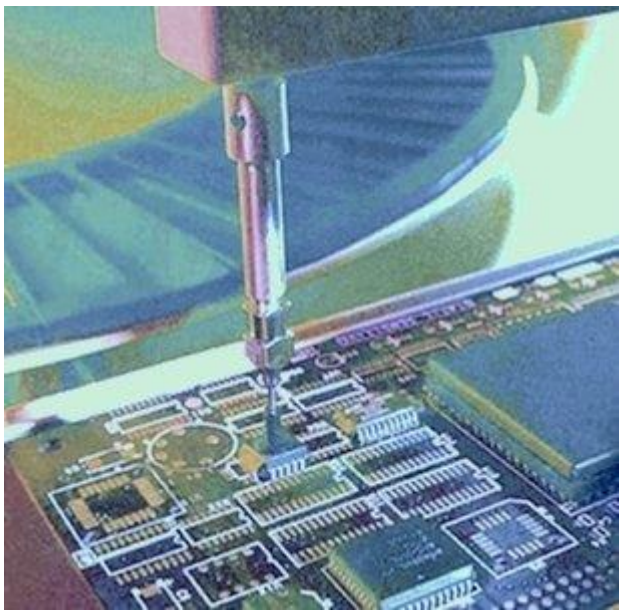


Рис. 1.2 – Установка мікросхеми за допомогою напівавтомата

1.1.1 Маніпулятори та напівавтомати

В умовах одиничного і дрібносерійного виробництва для установки поверхнево-монтованих елементів використовуються маніпулятори і напівавтомати.

Маніпулятор – найпростіший пристрій, оснащений вакуумним пінцетом, який керується оператором по осях X, Y, Z і забезпечує правильну орієнтацію компонента. Вакуум в пінцеті включається при захопленні компонента і відключається при його установці автоматично. Продуктивність, що забезпечується маніпулятором, залежить від досвіду оператора і становить від 150 до 600 компонентів на годину.

Наприклад, одним з маніпуляторів є маніпулятор Fritsch LM900 (рис. 1.3)

Рис. 1.3 – Маніпулятор Fritsch LM900

Він працює з різними типами живильників: для стрічки, для пенала, для палети, з розсипу і має наступні характеристики:

– максимальний розмір друкованих плат: до 440 × 245мм

– робоче поле монтажу: до 350х245мм

– швидкість установки: до 600 компонентів на годину

– модульна конструкція

– можливість підключення відеокамери

Напівавтомат установки компонентів допомагає оператору збирати електронні модулі. Напівавтомат, наприклад, здійснює переміщення ємності з потрібним компонентом, променем світла вказує точку установки компонента чи полегшує складання іншими способами, але установка компонента виконується оператором. Це зменшує ймовірність браку при установці. Продуктивність напівавтомату знаходиться в діапазоні 150 ... 1000 компонентів на годину і залежить від досвіду оператора.

Напівавтомат дозволяє підвищити продуктивність і виключити помилки, пов'язані з неправильно встановленими і неправильно орієнтованими компонентами. Це досягається завдяки програмному забезпеченню, контролюючому дії оператора і видає підказки про місце захоплення і місце установки. Захоплення компонента з неправильного джерела (живильника) виключається, тому що в цьому випадку в пінцеті не включається вакуумний захват.

Наприклад, одним з напівавтоматів є напівавтомат Fritsch SM902 (рис. 1.4)

Рис. 1.4 – Напівавтоматичний маніпулятор Fritsch SM902

1.1.2. Повні автомати

В автомат по конвеєру надходить друкована плата і фіксується в ньому тим чи іншим чином. Для того щоб точно встановити компоненти в задані місця, автомат повинен визначити місце розташування друкованої плати. Для цього автомати оснащені камерою, за допомогою якої проводиться зчитування спеціальних маркерів - реперних знаків, нанесених на друковану плату. Наявність реперних знаків - обов'язкова умова для друкованих плат, що підлягають автоматичній збірці. Автомат зчитує реперні знаки і визначає реальний стан друкованої плат в автоматі. Алгоритми які використовуються в автоматах дозволяють визначити не тільки лінійне і кутове зміщення, а й компенсувати нелінійні спотворення малюнка друкованої плати. За допомогою трьох реперів можна скорегувати похибки виготовлення друкованої плат, що виражаються у відхиленні від ортогональності - нахил осей. За допомогою чотирьох реперів можуть коригуватися похибки, пов'язані з нелінійним спотворенням фотошаблона при виготовленні друкованої плати.

Повні автомати використовуються в стабільно працюючому виробництві при випуску великих партій модулів. Ціна повного автомата визначається конфігурацією і функціями: технічний зір, роздільна здатність установки, темп роботи, число головок і інші.

Існує кілька типів автоматів (автоматичних установників елементів):

– револьверні;

– портальні (гнучкі системи для установки повернево-встановлюваних елементів (smd) з малим кроком виводів) системи широко використовуються у виробництві побутової електроніки, телекомунікації, ЕОМ, серверних комп'ютерів, а також в менших обсягах - у виробництві високонадійній електронної техніки.

Проте, потреба в ще більш високих обсягах виробництва, гнучких виробничих лініях змусили виробників розробити альтернативну архітектуру автоматів, здатних паралельно встановлювати безліч радіокомпонентів. Такі

автоматичні установники радіокомпонентів включають в себе високошвидкісні крокові електродвигуни, оптичні датчики.

1.1.2.1. Револьверні установчі автомати (для установки пасивних пристроїв)

Базовий револьверний автомат використовувався для установки пасивних smd компонентів (наприклад, конденсаторів, резисторів) на самих ранніх стадіях розвитку технології поверхневого монтажу. Револьверна голівка автоматичного установника представлена на малюнку 1.5

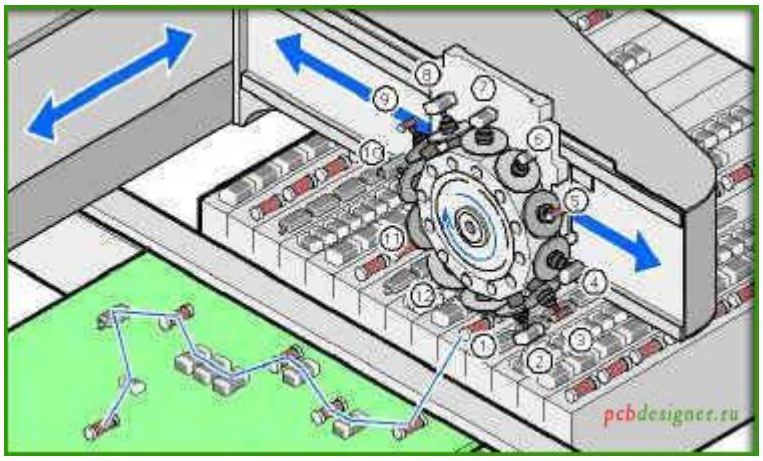


Рис. 1.5 – Приклад автоматичного установника з револьверною голівкою

Кілька захоплюючих головок розташовують навколо стаціонарної револьверної голівки, що обертається в горизонтальній площині. Рухомий візок встановлює стрічки живильників, з яких радіокомпоненти подаються в кожну голівку. Після захоплення елемента револьверний барабан повертає його до оптичної робочої станції для отримання зображення на камері приладу. Це зображення потім обробляється і ідентифікується, щоб встановити радіокомпонент в заданому місці друкованої плати. При обертанні револьверної голівки рухомий стіл позиціонує друковану плату таким чином, щоб потрібне місце знаходилося точно під голівкою з компонентом. Голівка опускається на

друковану плату, встановлюючи елемент. Потім голівка повертається для захоплення наступного радіоелементу - цикл повторюється. У таблиці 1.1 дані загальні технічні характеристики револьверних установчих автоматів.

Таблиця 1.1 – Технічні характеристики установочних револьверних автоматів

Характеристика	Величина (опис)
Розмір встановлюваних елементів	Від пасивних пристроїв типорозміром 0201 до компонентів в матричних корпусах розміром 10 мм
Кількість встановлюваних елементів	Кілька сотень
Швидкість установки	25 000-40 000 компонентів на годину
Основні функції	1. Захоплення кристалів (чіпів) і компонентів в матричних корпусах. 2. Рухома револьверна голівка і друкована плата. 3. Гнучкість в експлуатації – захоплення компонентів як з стрічок, так і розсипом

1.1.2.2. Портальні установчі автомати (для установки активних компонентів)

Конструкція портальних автоматів відрізняється від револьверних тим, що друкована плата нерухомо фіксується на місці, а рухається установча

голівка: захоплює радіоелементи і встановлює їх в правильне положення (малюнок 1.6).

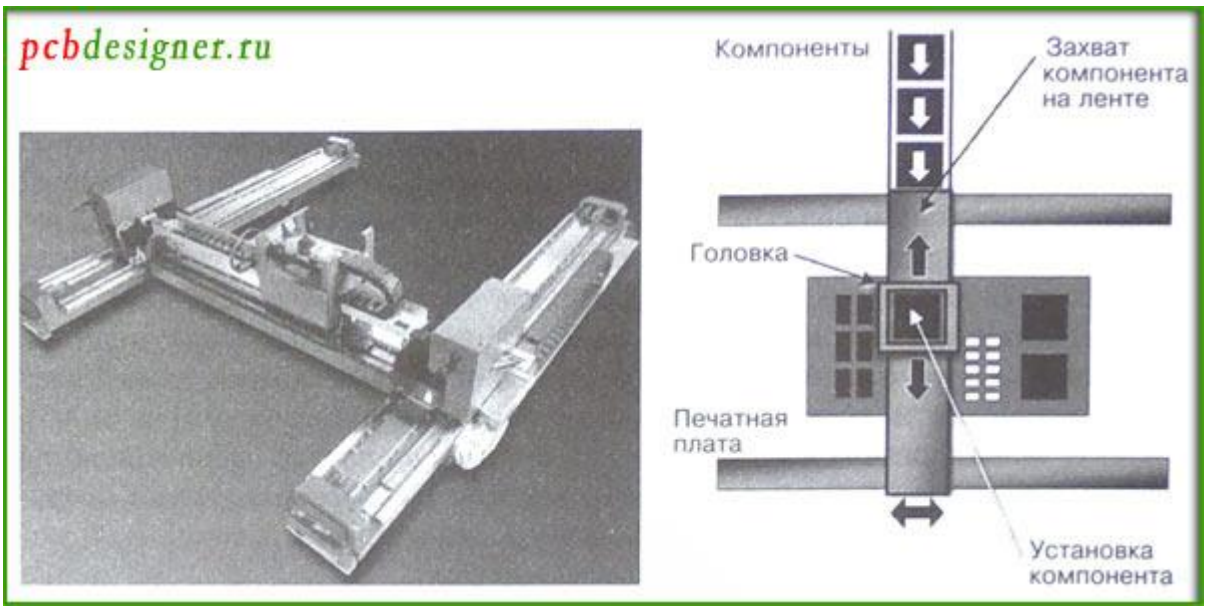


Рис. 1.6 – Портальна система, яка використовується для установки великих компонентів в матричних піддонах

Живильники також є нерухомими. Портальні автомати, як правило, використовуються для розміщення великих радіокомпонентів (наприклад, SOICs, PLCCs і т.д.). Деякі технічні характеристики портальних автоматичних установників наведені в таблиці 1.2.

Монтажна голівка автоматичного установника розміщується над живильником, звідки захоплює елемент. Потім голівка переносить елемент до відеокамери нижнього перегляду для перевірки, після чого встановлює компонент на друкованій платі. Другим варіантом є двпортальна система з двома блоками установчих головок, кожна з яких може встановлювати як один тип радіокомпонентів, так і кілька завдяки шпиндельній голівці.

Таблиця 1.2 – Технічні характеристики портальних установчих автоматів

характеристика	Величина (опис)
Розмір встановлюваних елементів	Елементи в корпусах SMT (SOD, SOT, SOIC, PLCC, CCGA, BGA), а також компоненти складної форми
Кількість встановлюваних елементів	Декілька сотень
Швидкість установки	5 000-15 000 компонентів на годину
Основні функції	1. Установка елементів негабаритів розмірів і складної форми з високою точністю 2. Рухома голівка, нерухомі друкована плата і живильники 3. Гнучкість в експлуатації – захоплення елементів з стрічок, з туб, матричних носіїв і розсипом

Деякі функції портальних установників дозволяють наблизити швидкість їх монтажу до найбільш «високошвидкісних» моделей револьверних автоматів. У таблиці 1.3 наведені деякі технічні характеристики двопортальних систем. Функції портального автомата, які дозволяють підвищити швидкість установки радіоелементів: слайсинг стрічки живильника для безперебійної роботи машини і спрощення банку даних, що дозволяє швидко замінювати типи компонентів на різних технологічних лініях. Крім того, контроль радіокомпонентів може проводитися під час руху установчих голівок («на льоту») завдяки відеокамері, встановленії на самій голівці, що виключає з виробничого циклу проміжок часу, необхідний для центрування елементу за

допомогою стаціонарної камери нижнього перегляду.

Таблиця 1.3 – Технічні характеристики високошвидкісних портальних настановних автоматів

Характеристика	Величина (опис)
Розмір встановлюваних елементів	Елементи в корпусах SMT (SOD, SOT, SOIC, PLCC, CCGA, BGA), а також компоненти складної форми
Кількість встановлюваних компонентів	Декілька сотень
Швидкість установки	15 000-21 000 компонентів на годину
Основні функції	1. Установка елементів надгабаритних розмірів і складної форми з високою точністю 2. Встановлення елементів з підвищеною щільністю 3. Живильники підвищеної ємності подачі елементів зі стрічок, туб, матричних носіїв і розсипом

Подальші поліпшення обладнання для монтажу радіокомпонентів можуть бути реалізовані в автоматах з модулями паралельно працюючих голівок. Технічні характеристики такого обладнання наведені в таблиці 1.4. Кілька установчих модулів здатні захоплювати, центрувати і встановлювати радіоелементи в відповідне місце друкованої плати. Друкована плата переміщається кроковим конвеєром таким чином, щоб місце установки знаходилося точно під захопленим елементом.

Таблиця 1.4 – Технічні характеристики модульних портальних систем

Характеристика	Величина (опис)
Розмір встановлюваних компонентів	Від пасивних пристроїв типорозміром 0201 до компонентів в матричних корпусах розміром 25 мм
Кількість встановлюваних компонентів	Декілька сотень
Швидкість установки	60 000-100 000 компонентів на годину
Основні функції	1. Установка чіп-компонентів і компонентів в матричних корпусах 2. Монтаж друкованих плат великого розміру 3. Кілька стрічкових живильників

Машини для автоматичної установки працюють за трьома основними принципами: почергової, почергово-одночасної і одночасної установки компонентів. В апаратах почергової установки один компонент весь час встановлюється однією або двома установчими голівками. Почергова установка, також може проводитися за допомогою револьверної голівки. При послідовно-одночасній установці кілька компонентів може бути встановлено одночасно. Установчі машини одночасного типу, встановлюють всі або найбільш можливу кількість компонентів за один раз.

Почергові і почергово-одночасні машини, також називаються послідовними і їх основна перевага в гнучкості налаштування. Якщо машина почергової установки оснащена револьверної голівкою, швидкість установки компонентів на друковану плату значно зростає. Ці машини можуть встановлювати компоненти декількох типів (див. таблицю 1.1). Місце установки компонента може бути легко змінене, а точність установки досить

висока.

Машини одночасної установки компонентів значно продуктивніші. Швидкість установки компонентів може досягати 300000 компонентів на годину, проте ці машини не такі прості і гнучкі в налаштуванні. Якщо для зміни місця установки компонента в машинах почергового і по почергово-одночасного типу досить змінити програми, то для машини одночасної установки можуть бути потрібні значні механічні зміни. Тому, ці машини використовуються, в основному, для великих партій виробів.

Однак слід зазначити, що максимальна продуктивність на практиці не досягається. Для реальної оцінки продуктивності автомата необхідно його максимальну продуктивність помножити на емпіричний коефіцієнт, що залежить від складності пристроїв, що випускаються на підприємстві. У більшості випадків даний коефіцієнт лежить в межах 0,5-0,6.

У сучасному обладнанні захоплення компонентів здійснюється вакуумної головкою. Для захоплення важких компонентів застосовуються спеціальні насадки. Розробники компонентів для забезпечення можливості вакуумного захоплення створюють збалансовані компоненти з певним центром мас. У деяких випадках (наприклад, у довгих мезонінних роз'ємів), можливість захоплення вакуумом вимагає спеціальних деталей з широкою горизонтальною площиною, що знімаються з компонента після установки.

1.2 Види 3D принтерів

3D принтер представляє собою високотехнологічне обладнання, яке служить для створення різних об'єктів з декількох шарів, при цьому в якості зразка використовується тривимірна модель. Для створення певних моделей використовуються спеціальні програми, які викачуються на ПК і принтер. Ці програми призначаються для тривимірного моделювання.

3D принтери можна класифікувати:

– за кінематичною схемою;

– за технологією друку;

– на монохромні,що друкують одним кольором та кольорові, що дозволяють створювати різнокольорові фізичні об’єкти;

– принтери з здатністю 3D-прототипування. Такі пристрої дозволяють виготовляти найдрібніші деталі; тощо.

1.2.1. Види 3D принтерів за кінематичною схемою

Кінематична схема (принципова) – це така схема, на якій показана послідовність передачі руху від двигуна через передавальний механізм до робочих органів машини (наприклад, шпинделя верстата, ріжучого інструменту, коліс автомобіля і ін.) і їх взаємозв'язок.

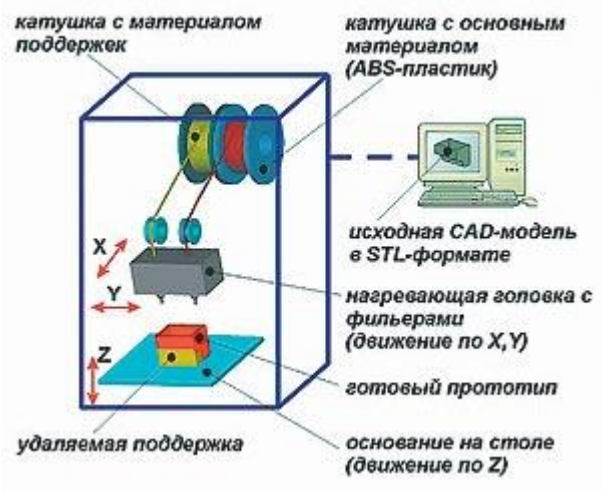


Рис. 1.7 – Модель 3D принтеру

Як видно з рис. 1.7 об'єкт повинен переміщатися в трьох площинах: вправо-вліво (вісь X), вперед-назад (вісь Y) і вгору-вниз (вісь Z).

Зворотне справедливо і для друкуючої головки екструдера, тобто якщо об'єкт залишатиметься на місці, а переміщатися почне тільки головка, то це не стане на заваді повноцінному процесу побудови деталі.

Можуть бути різні варіанти, наприклад, коли головка має одну ступінь свободи (піднімається в площині Z), а робочий стіл з об'єктом рухається в двох інших або навпаки – стіл опускається, тоді як екструдер переміщається над ним по осях X і Y. Ці відмінності кінематичної частини, реалізовані в деяких поширених моделях 3D принтерів.

Перша і найчисленніша група: екструдер переміщається по осях X і Z, а по осі Y переміщається платформа.

З найбільш популярних принтерів що входять в це сімейство можна назвати Prusa Mendel, PrintrBot, і інші Rep-Rap-подібні конструкції.

Відмінною їх особливістю є відкрита платформа і два трикутних елементи розташованих з боків.

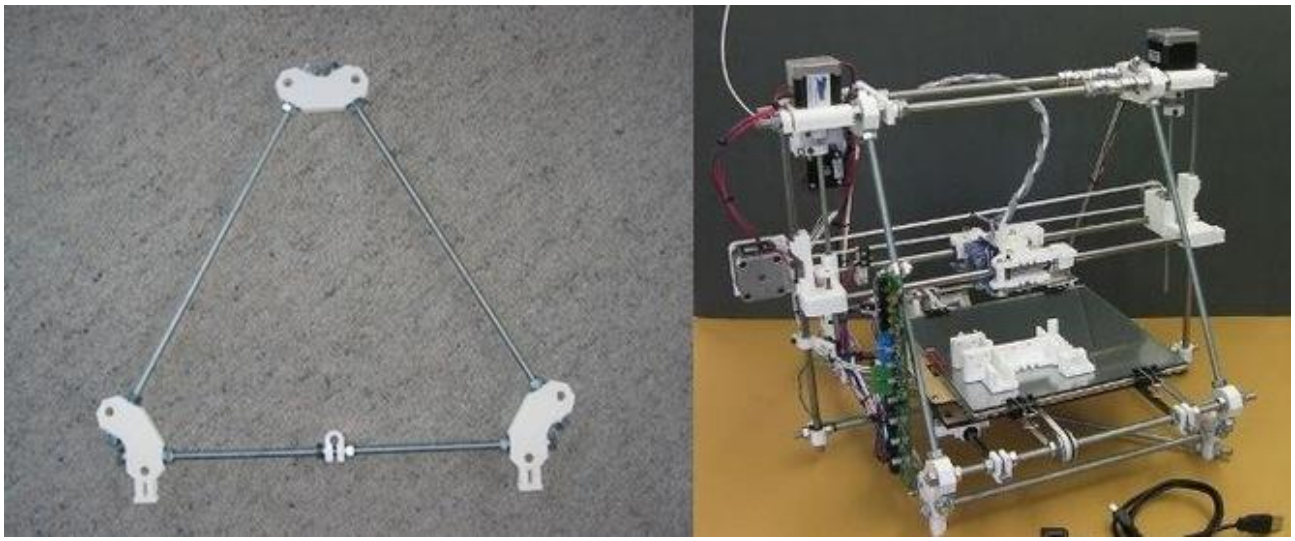


Рис. 1.8 – Приклад принтеру в якому екструдер переміщається по осях X і Z, а по осі Y переміщається платформа

Це спрощує процес складання, але породжує масу проблем, пов'язаних з недостатньою жорсткістю несучих елементів каркасу, що призводить до паразитних вібрацій від яких страждає точність друку.

У конструкцію PrintrBot’а його творцем Бруком Драммом, були внесені деякі зміни покликані поліпшити жорсткість несучої рами. Для цього він відмовився від трикутного каркаса і зробив важчою основу з одночасним розміщенням там електронних компонентів.

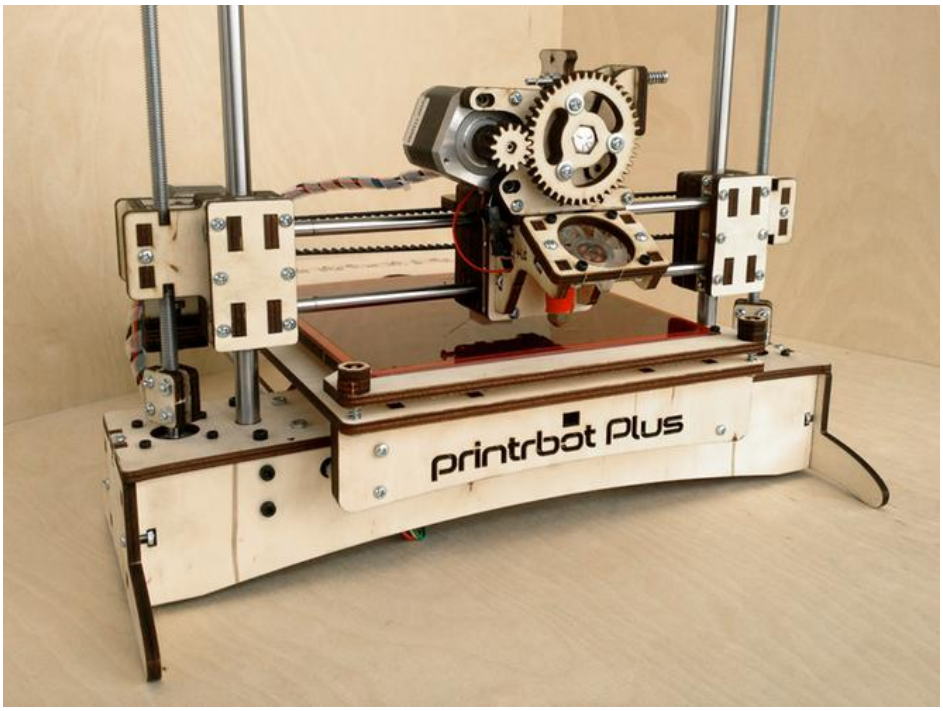


Рис. 1.9 – Printrbot Plus

Це дозволило спростити процес складання і здешевити конструкцію принтера.

Друга група принтерів. У цій групі робочий стіл з друкованими об'єктами рухається тільки вгору або вниз по осі Z, а по осях X, Y рухається друкуюча головка екструдера.

Представником цього сімейства є Makerbot (рис. 1.10).

Рис. 1.10 – Makerbot

Третя група. Схема, за якою друкуючий вузол переміщається по осі X, а робочий стіл - по осях Y і Z, не отримала широкого поширення внаслідок складної реалізації конструкції і налагодження пристрою.

Четверта група. Це так звані дельта-принтери. Принтери, в яких робочий стіл нерухомий, а переміщається тільки друкуюча голівка екструдера, що приводиться в рух трьома маніпуляторами розташованими навкруги називаються delta-printers (рис. 1.11).

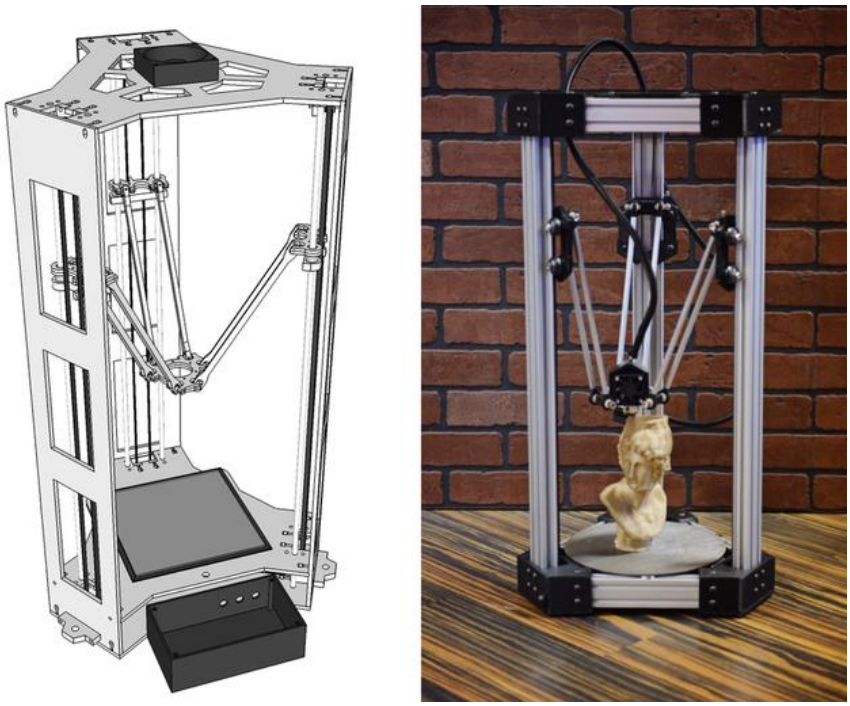


Рис. 1.11 – дельта-принтер

Сам по собі, принтер в якому робочий стіл нерухомий, не може за замовчуванням називатися дельта-принтером, тому що його екструдер може мати привід від трьох двигунів, розташованих перпендикулярно один одному в кожній з площин - X, Y і Z.

Але тільки в конструкції дельта-принтера це переміщення реалізується за кінематичною схемою дельта-робота, за допомогою паралельно встановлених по колу приводів.

Одним з чинників знижують інтерес до такої кінематичної схеми, може бути і його доволі невелика область друку в площині XY.

1.2.2. Види 3D принтерів за технологією друку

1) Принтери що видавлюють або виливають або розпилюють

1.1) FDM (fused deposition modeling) принтери які видавлюють матеріал шар за шаром через сопло-дозатор. Це, наприклад, принтери Stratasys,

різні кулінарні принтери (використовують глазур, сир, тісто), медичні які друкують "живими чорнилом" (коли набір живих клітин поміщається в спеціальний медичний гель які використовується далі в біомедицині)

1.2) Технологія Polyjet, була винайдена ізраїльською компанією Objet в 2000 р. В даній технології фотополімер маленькими дозами вистрілюється з тонких сопел, як при струменевому друку, і відразу полімеризується на поверхні під впливом ультрафіолетового випромінювання. В даній технології є можливість друку різними матеріалами.

Переваги технології:

а) товщина шару до 16 мікрон (клітка крові 10 мікрон)

б) швидко друкує, так як рідину можна наносити дуже швидко.

Недоліки технології:

а) друкує тільки з використанням фотополімеру – вузькоспеціалізований, дорогий пластик, як правило, чутливий до ультрафіолету і досить крихкий.

1.3) LENS (Laser Engineered Net Shaping)

Матеріал у формі порошку видувається з сопла і потрапляє на сфокусований промінь лазера. Частина порошку пролітає повз, а та частина, яка потрапляє у фокус лазера спікається і шар за шаром формує тривимірну деталь. За такою технологією друкують сталеві і титанові об'єкти. Порошок різних матеріалів можна змішувати і отримувати таким чином сплави, на льоту.

1.4) LOM (laminated object manufacturing)

Тонкі ламіновані листи матеріалу вирізаються за допомогою ножа або лазера і потім спікаються або склеюються в тривимірний об'єкт. Тобто укладається тонкий лист матеріалу, який вирізається по контуру об'єкта, таким чином виходить один шар, на нього укладається наступний лист і так далі. Після цього всі листи пресуються або спікаються.

Таким чином друкують 3D моделі з паперу, пластику або з алюмінію. Для друку моделей з алюмінію використовується тонка алюмінієва фольга, яка вирізається по контуру шар за шаром і потім спікається з допомогою ультразвукової вібрації.

2. Принтери які спікають або склеюють

2.1) SL (Stereolithography) Стереолітографія.

Є невелика ванна з рідким полімером. Промінь лазера проходить по поверхні, і в цьому місці полімер під впливом ультрафіолету полімеризується. Після того як один шар готовий платформа з деталлю опускається, рідкий полімер заповнює пустоту далі запікається наступний шар і так далі. Іноді відбувається навпаки: платформа з деталлю піднімається вгору, лазер відповідно розташований знизу.

Після друку таким методом необхідна додаткова обробка об'єкту – видалення зайвого матеріалу, іноді поверхню шліфують. Залежно від необхідних властивостей кінцевого об'єкту модель запікають у так званих ультрафіолетових духовках.

Фотополімер часто буває токсичним тому при роботі з ним треба користуватися засобами захисту і респіраторами.

Серед переваг даної технології можна виділити швидкість і точність, точність до 10 мікрон. Для спікання фотополімеру досить лазера від Blu-ray програвача, завдяки чому на ринку з'являються порівняно дешевші і при цьому точні принтери, що працюють за такою технологією.

2) LS (laser sintering)

Лазерне спікання. Схоже на SL, тільки замість рідкого фотополімеру використовується порошок, який спікається лазером.

Переваги:

а) менш імовірно, що деталь зламається в процесі друку;

б) матеріали в формі порошку досить легко знайти в продажу; в тому числі це можуть бути: бронза, сталь, нейлон, титан;

Недоліки:

а) поверхня виходить пориста;

б) деякі порошки вибухонебезпечні, тому повинні зберігатися в камерах, заповнених азотом;

в) спікання відбувається при високих температурах, тому готові деталі довго остигають, в залежності від розміру і товщини шарів, деякі предмети можуть остигати до одного дня.

3) 3DP (three dimensional printing)

Технологія винайдена в 1980 році в MIT студентом Paul Williams.

На матеріал в порошкової формі наноситься клей, який пов'язує гранули, потім поверх клеєного шару наноситься свіжий шар порошку, і так далі. На виході, як правило, виходить матеріал sandstone (схожий за властивостями на гіпс).

Переваги:

а) оскільки використовується клей, в нього можна додати фарбу і таким чином друкувати кольорові об'єкти;

б) технологія відносна дешева і енергоефективна;

в) можна використовувати в умовах будинку або офісу;

в) можна друкувати використовувати порошок скла, кістковий порошок, перероблену гуму, бронзу і навіть деревну тирсу. Використовуючи подібну технологію можна друкувати їстівні об'єкти, наприклад, з цукру або

шоколадного порошку. Порошок склеюється спеціальним харчовим клеєм, в клей може додаватися барвник і ароматизатор. Як приклад, нові 3D принтери від компанії 3D systems, які були продемонстровані на CES 2014 року - ChefJet і ChefJet Pro.

Недоліки:

а) виходить досить груба поверхня;

б) матеріал потрібно піддавати додатковій обробці (запікати), щоб надати йому необхідні властивості.

1.3 Крокові двигуни

Основна відмінність між кроковим двигуном і всіма іншими типами двигунів складається в способі, завдяки якому відбувається обертання. На відміну від інших моторів, крокові двигуни обертаються не безперервно. Замість цього, вони обертаються кроками (звідси і їх назва). Кожен крок є частиною повного обороту. Ця частина залежить, в основному, від механічного пристрою мотора і від обраного способу управління ім. Крокові двигуни також розрізняються способами живлення. На відміну від двигунів змінного або постійного струму, зазвичай вони керуються імпульсами. Кожен імпульс перетворюється в градус, на який відбувається обертання. Наприклад, 1.8º кроковий двигун, повертає свій вал на 1.8° при кожному вступнику імпульсі. Часто, через цю характеристики, крокові двигуни ще називають цифровими.

Розрізняють біполярні та уніполярні крокові двигуни.

У біполярного двигуна використовуються 4 проводи для підключення мотора до контролера. Обмотки з'єднуються всередині послідовно або паралельно.

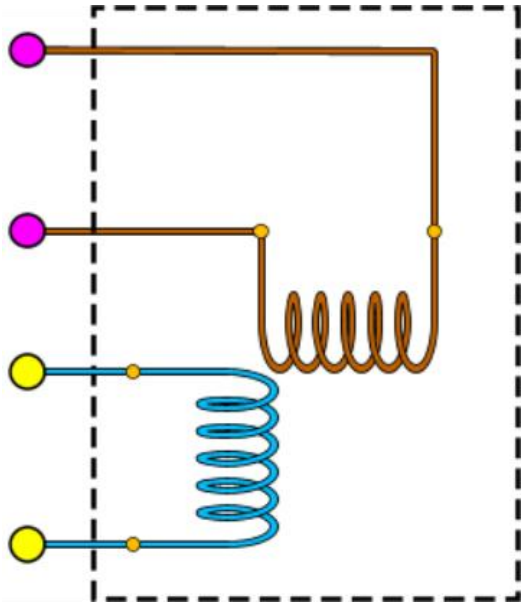
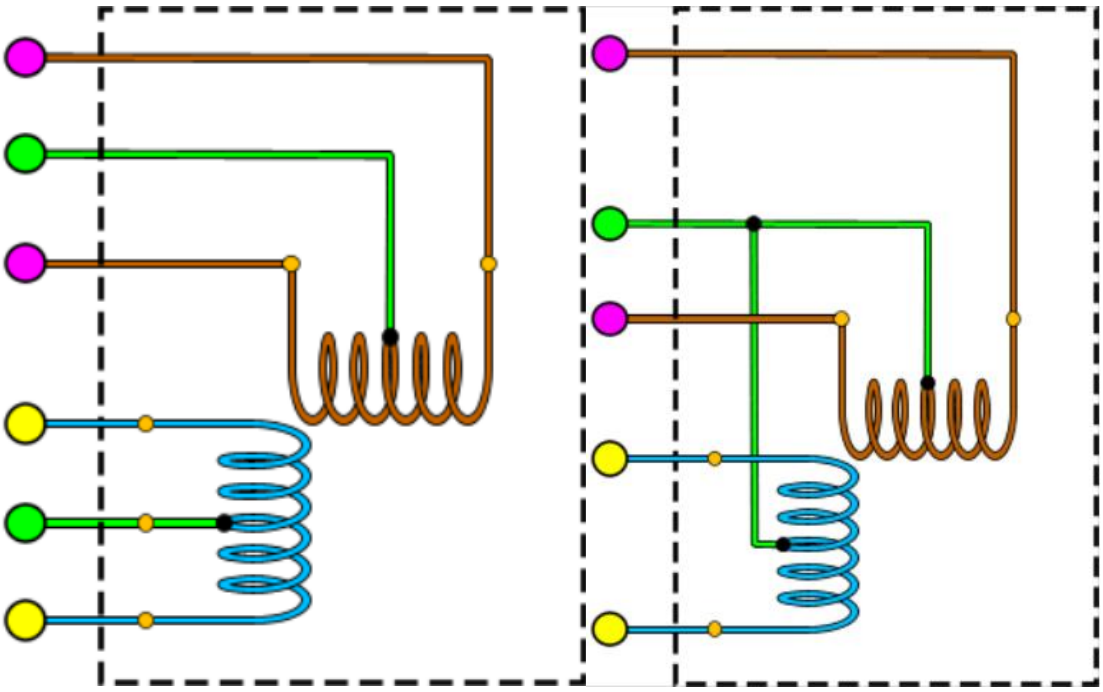


Рис. 1.12 – Приклад біполярного крокового двигуна

У уніполярному двигуні загальний провід підключений до точки, де дві обмотки з'єднані разом:



а) б)

Рис. 1.13а, б – Приклади уніполярних крокових двигунів

1.4 Детектор кордонів Кенні (Canny)

Хоча робота Кенні була проведена на зорі комп'ютерного зору (1986), детектор границь Кенні досі є одним з кращих детекторів.

Кроки детектора:

- Прибрати шум і зайві деталі з зображення

- Розрахувати градієнт зображення

- Зробити краї тонкими (edge thinning)

- Зв'язати краї в контури (edge linking)

Детектор використовує фільтр на основі першої похідної від Гауссіани. Так як він сприйнятливий до шумів, краще не застосовувати даний метод на необроблених зображення. Перш за все, вихідні зображення потрібно звернути з гаусовим фільтром.

Межі на зображенні можуть перебувати в різних напрямках, тому алгоритм Кенні використовує чотири фільтра для виявлення горизонтальних, вертикальних і діагональних кордонів.

Кут напрямку границі округляється до одного з чотирьох кутів, що представляють вертикаль, горизонталь і дві діагоналі (наприклад, 0, 45, 90 і 135 градусів).

Потім йде перевірка того, чи досягає величина градієнта локального максимуму в відповідному напрямку.

Наприклад, для сітки 3x3:

– якщо кут напрямку градієнта дорівнює нулю, точка буде вважатися межею, якщо її інтенсивність більше ніж у точки вище і нижче даної точки;

– якщо кут напрямку градієнта дорівнює 90 градусів, точка буде вважатися межею, якщо її інтенсивність більше ніж у точки зліва і справа розглядуваної точки;

– якщо кут напрямку градієнта дорівнює 135 градусам, точка буде вважатися межею, якщо її інтенсивність більше ніж у точок що знаходяться в верхньому лівому і нижньому правому куті від даної точки;

– якщо кут напрямку градієнта дорівнює 45 градусам, точка буде вважатися межею, якщо її інтенсивність більше ніж у точок знаходяться у верхньому правому і нижньому лівому кутку від даної точки.

Таким чином, отримуємо двійкове зображення, що містить границі (т.зв. «тонкі краї»).

У OpenCV, детектор кордонів Кенні реалізується функцією cvCanny ().

CVAPI (void) cvCanny (const CvArr * image, CvArr * edges, double threshold1, double threshold2, int aperture_size CV_DEFAULT (3));

image - одноканальне зображення для обробки (градації сірого)

edges - одноканальне зображення для зберігання кордонів, знайдених функцією

threshold1 - поріг мінімуму

threshold2 - поріг максимуму

aperture_size - розмір для оператора Собеля

1.4.1 Оператор Собеля

Оператор Собеля - це дискретний диференціальний оператор, який обчислює наближення градієнта яскравості зображення.

Оператор обчислює градієнт яскравості зображення в кожній точці. Так знаходиться напрямок найбільшого збільшення яскравості і величина її зміни в цьому напрямку. Результат показує, наскільки «різко» або «плавно» змінюється яскравість зображення в кожній точці, а значить, ймовірність знаходження точки на межі, а також орієнтацію кордону.

Таким чином, результатом роботи оператора Собеля в точці області постійної яскравості буде нульовий вектор, а в точці, що лежить на кордоні областей різної яскравості – вектор, що перетинає кордон в напрямку збільшення яскравості.

Найбільш часто оператор Собеля застосовується в алгоритмах виділення кордонів.

Оператор Собеля заснований на згортці зображення невеликими цілочисельними фільтрами в вертикальному і горизонтальному напрямках, тому його відносно легко обчислювати. Оператор використовує ядра (матриці, наприклад, розмірності 3x3), з якими згортають вихідне зображення для обчислення наближених значень похідних по горизонталі і по вертикалі.

Оператор використовує ядра (наприклад, 3x3), з якими згортають вихідне зображення для обчислення наближених значень похідних по горизонталі і по вертикалі. Нехай A – це вихідне зображення, а G_x і G_y - два зображення, на яких кожна точка містить наближені похідні по x і по y. Вони обчислюються наступним чином:

$$\mathbf{G}_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix} * \mathbf{A} \quad \text{and} \quad \mathbf{G}_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} * \mathbf{A}$$

де * позначає двовимірну операцію згортки.

Координата x тут зростає «направо», а y – «вниз». У кожній точці зображення наближене значення величини градієнта можна обчислити шляхом використання отриманих наближених значень похідних:

$\mathbf{G} = \sqrt{\mathbf{G}_x{}^2 + \mathbf{G}_y{}^2}$ (мається на увазі поелементно).

Використовуючи цю інформацію, ми можемо також обчислити напрямок градієнта:

$\mathbf{\Theta} = \arctan\left(\frac{\mathbf{G}_y}{\mathbf{G}_x}\right)$ де, наприклад, кут Θ дорівнює нулю для вертикальної межі, у якій темна сторона зліва.

1.5 Висновки до розділу 1

В даному розділі було наведено необхідні теоретичні відомості. Їх можна умовно поділити на 4 частини.

В першій частині було проведено аналіз та класифікацію існуючих програмно-апаратних рішень для встановлення компонентів на друкованій платі. Зокрема було розглянуто маніпулятори, напівавтомати, револьверні та портальні автомати. Також були наведені їх технічні характеристики та області використання.

Друга частина присвячена дослідженню видів 3D принтерів та їх технологій друку. Описані переваги, недоліки та області застосування 3D принтерів за їх кінематичною схемою. Розглянуті поширені технології друку.

В третій частині даються короткі відомості про крокові двигуни та їх види.

Четверта частина присвячена опису детектору кордонів Кенні (Canny) та її реалізації в OpenCV. Також наведено математичний опис роботи даного детектору з використанням оператора Собеля.

2. ПРАКТИЧНА РЕАЛІЗАЦІЯ МОДУЛЮ

2.1. Апаратна частина

Апаратна частина модулю складається з осей від 3D принтеру, робочої площадки, вертикальної площадки, муфт, крокових двигунів та мікроконтролеру з камерою, насосу, реле, механічних кінцевиків (Endstop Switch Module) джерел живлення, драйверів крокових двигунів, керуючого мікроконтролеру, кріплень (гайок, болтів, ізоляційної стрічки тощо) та проводів.

Загальну схему модулю можна побачити на рис. 2.1. На даному малюнку 2.1 – крокові двигуни, 2 – муфти, 3 – осі, 4 – основа, 5 – робоча площадка, 6 – основа для кріплення осі OY, 7 – основа для кріплення осі OZ, 8 – тримач, 9 – модуль отримання зображення (Raspberry Pi та Raspberry Pi Camera Module), 10 – джерела живлення, 11 – драйвери крокових двигунів, 12 – реле, 13 – насос, 14 – керуючий мікроконтролер (Arduino Mega 2560), 15 – маршрутизатор для отримання зображення від Raspberry Pi, 16 – комп’ютер, 17 – трубка. Зовнішній вигляд апаратної частини модулю зображений на малюнку 2.2.

Габарити вертикальної площадки ≈ 39x11 $см^2$.

Габарити горизонтальної площадки ≈ 26x22 $см^2$.

Параметри деталей можуть варіюватися. Проте умовою є можливості насосу захопити деталь даної ваги та форми, а також темний колір деталі (аби вона краще розпізнавалась, адже деталі знаходяться на білому фоні). Проте і ці обмеження можна дещо пом’якшити – можна за необхідності замінити насос на більш потужний, зробити насадку на робочий носик зручнішої форми, світлі деталі розміщувати на темному фоні.

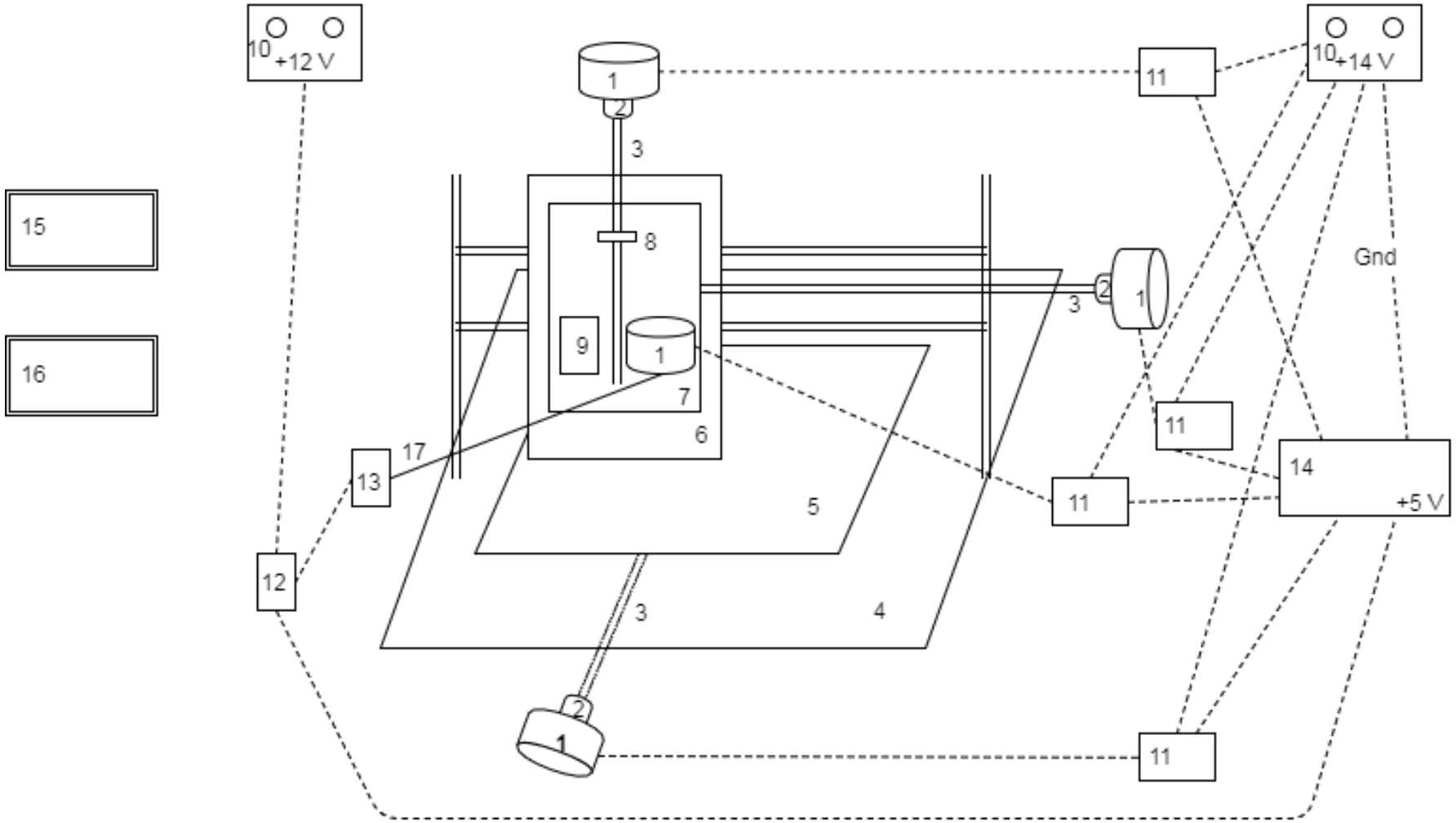


Рис. 2.1 – Загальна схема модулю

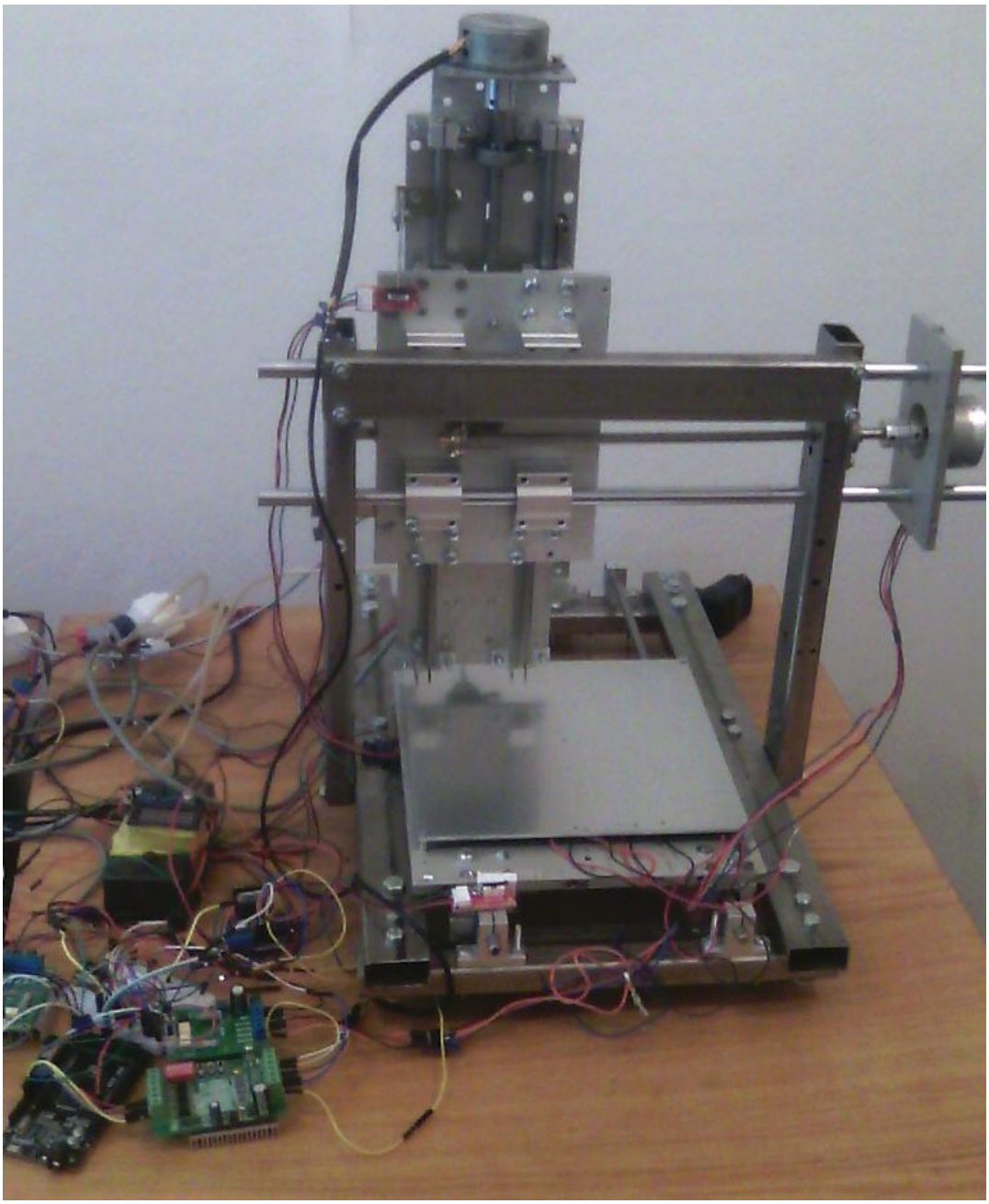


Рис. 2.2 – Вигляд апаратної частини модулю

Послідовність роботи модулю наступна:

1. Мікроконтролер (рис. 2.1, елемент 14) подає сигнали для установлення всіх площадок в початкове положення. Це положення в якому площадки замикають всі механічні кінцевики і є точкою відліку (координати O(0,0,0)). Сигнали від мікроконтролеру подаються до драйверів крокових двигунів, які в свою чергу живлять

двигуни від зовнішнього джерела. Площадки переміщуються кроковими двигунами, що прикріплені до гвинтових осей (рис. 2.1, елемент 3) за допомогою муфт (рис. 2.1, елемент 2).

2. Модуль з камерою, закріплений на вертикальній площадці ((рис. 2.1, елемент 9) з певним часовим інтервалом (напр. 2 секунди) захоплює зображення та передає його. В даній роботі використовувався Raspberry Pi та Raspberry Pi Camera Module. Raspberry Pi – одноплатний комп'ютер, що керується операційною системою. Тому для отримання зображень з інтервалом в 2 секунди був написаний відповідний bash-скрипт (див. додаток 1).

3. Зображення можна отримати на комп'ютер підключивши Raspberry Pi та комп'ютер до маршрутизатору(рис. 2.1, елементи 9, 16 та 15) та перейшовши за адресою 192.168.1.100:8080.

4. Зображення оброблюється оператором за необхідності та визначаються координати найближчого елемента.

5. Крокові двигуни повинні перемістити площадки в визначені координати, опустити вертикальну пощадку (рис. 2.1, елементи 7 і 8), подати дозволяючий сигнал на реле (рис. 2.1, елемент 12), що живить насос (рис. 2.1, елемент 13) від зовнішнього джерела (рис. 2.1, елемент 10), перемістити захоплену деталь, за необхідності позиціонувати її та вимкнути насос (відповідний сигнал реле від мікроконтролеру).

6. Повторити від пункту 2, доки не будуть розставлені всі необхідні деталі, або доки не закінчаться деталі з робочої площадки.

2.1.1 Переміщення по осям

Вертикальна площадка переміщається по осям OY та OZ. Площадка з елементами та платою (горизонтальна площадка) переміщається по OX.

Переміщення здійснюється за допомогою крокових двигунів.

В даній роботі тип двигунів (уніполярні чи біполярні) не є критичною характеристикою модулю, адже уніполярний двигун можна використовувати як біполярний просто залишивши загальний провід непідключеним, обо підключити його до спільної землі. Для переміщення по осям використовувались 2 біполярні двигуни (рис. 2.3) та 1 уніполярний (бо ці двигуни повинні переміщати навантаження – площадки, а біполярні потужніші за аналогічні уніполярні двигуни).

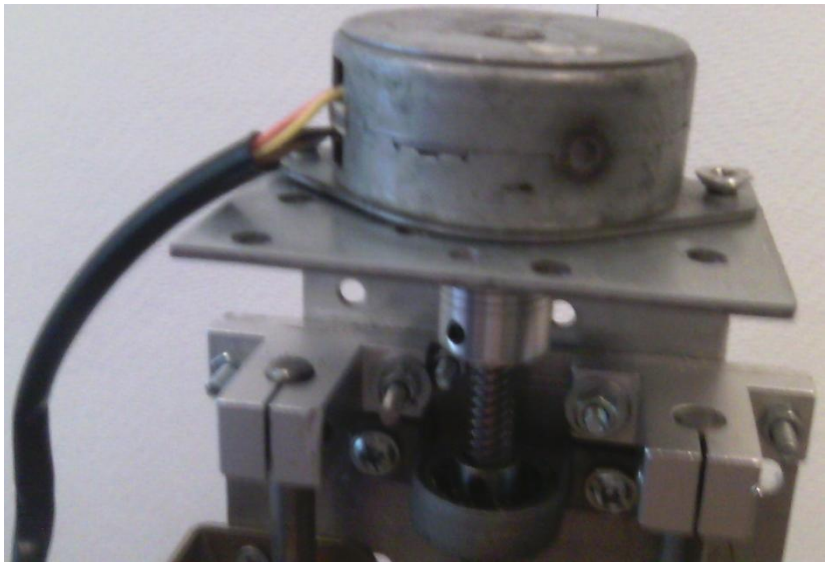


Рис. 2.3 – Біполярний двигун на осі модулю

Сигнали для переміщення передаються мікроконтролером (в даній роботі використовувався Arduino Mega 2560 (рис. 2.1, елемент 14)) до драйверу відповідного двигуна (для переміщення по осям використовувались 2 драйвери A3977 та 1 драйвер TB6560 V2).

Драйвери відповідно до отриманих сигналів подають на двигуни послідовність сигналів з величиною напруги пропорційною напрузі живлення. Таким чином земля зовнішнього джерела живлення і земля мікроконтролеру

повинна бути спільною. В деяких драйверах це вже реалізовано апаратно (зазвичай в них є спеціальні входи для 2-х сигналів землі – один для землі джерела живлення, один для землі мікроконтролеру), в деяких землі потрібно підключати разом (в таких драйверах зазвичай один вхід для землі).

Отримана від драйверів послідовність сигналів повертає вал двигуна. Дані двигуни прикріплені до гвинтових осей за допомогою муфт. Таким чином при повороті валу двигуна повертається відповідна гвинтова вісь, що в свою чергу переміщує відповідну площадку.

В залежності від характеристик крокового двигуна, тертя на осі тощо, можливе різне співвідношення параметрів кількість кроків двигуна – довжина пройденого шляху. Емпіричні вимірювання цих параметрів в даного модулю наведені в таблиці 2.1:

Таблиця 2.1 – Співвідношення параметрів кількість кроків двигуна – довжина пройденого шляху для різних осей модулюю

Вісь	Кількість кроків	Пройдена відстань (мм)
OX	100	3
OY	100	1.5
OZ	100	0.85

Проте ці дані не є точним орієнтиром, адже іноді крокові двигуни пропускають кроки. Саме тому необхідний зворотній зв’язок для контролю положення площадок модулю. Цей зворотній зв’язок здійснюється за допомогою камери. Зображення з камери бажано отримувати і обробляти в реальному часі, але для камери Raspberry Pi Camera Module, яка була використана в даній роботі, існує технологічне обмеження – камера може

захватувати зображення 1 раз в 2 секунди. Цей час потрібен камері аби встановити її рівні освітленості.

2.1.2 Захоплення та позиціонування деталі

Коли робочий носик знаходиться над деталлю це фіксується камерою. Відповідне зображення через маршрутизатор передається до комп'ютеру. Дана ситуація повинна бути зафіксована обробляючою програмою (різка зміна координат найближчої деталі) і керуючому мікроконтролеру повинен бути переданий відповідний сигнал (наприклад по UART(universal asynchronous receiver/transmitter)). Обробка даного сигналу повинна здійснюватись контролером у перериванні для оперативного реагування. Мікроконтролер у відповідь повинен подати послідовність сигналів до драйверу крокового двигуна що керує переміщенням по осі OZ та опустити кроковий двигун з насадкою (робочим носиком), що прикріплені до вертикальної площадки. Після цього необхідно подати дозволяючий сигнал на реле, що в свою чергу ввімкне живлення насосу від зовнішнього джерела.

Насос трубкою з'єднаний з кроковим двигуном що призначений для позиціонування деталі. Таким чином насос повинен захватити деталь через носик двигуна. Далі мікроконтролер повинен подати сигнали драйверу крокового двигуна що керує переміщенням по осі OZ та підняти кроковий двигун з насадкою (піднімання та опускання вертикальної площадки повинне здійснюватись на невелику, але фіксовану величину). Деталь потрібно перемістити на необхідну позицію. Під час переміщення насос повинен бути ввімкнутим (інакше деталь просто випаде).

Перемістившись до відповідної координати, деталь, за необхідності, позиціонують. В даному випадку позиціонування може здійснюватись поворотом відповідного крокового двигуна. Для позиціонування використовувався біполярний двигун (рис. 2.4) та драйвер L298N.

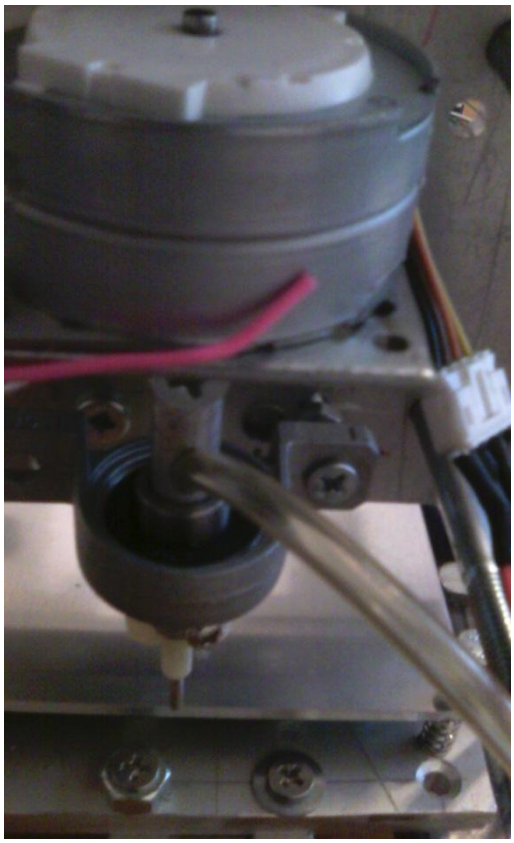


Рис. 2.4 – Біполярний кроковий двигун для позиціонування деталі

Мікроконтролер повинен подати сигнали драйверу крокового двигуна що керує переміщенням по осі OZ та опустити кроковий двигун з насадкою і подати сигнал реле для відключення насосу. Деталь залишиться на місці. Мікроконтролер повинен подати сигнали драйверу крокового двигуна що керує переміщенням по осі OZ та підняти кроковий двигун з насадкою. Далі необхідно знову здійснити пошук найближчої деталі та почати все спочатку доки не закінчаться деталі, або доки не будуть встановлені всі необхідні деталі.

2.2. Програмна частина

2.2.1 Захоплення зображення з камери

В даній роботі використовувалися Raspberry Pi та Raspberry Pi Camera Module. Raspberry Pi – одноплатний комп’ютер, що керується операційною системою. Була установлена Raspbian GNU/Linux 7 (wheezy). Для доступу до

Raspberry Pi використовувався маршрутизатор: Raspberry та комп'ютер підключались до маршрутизатору. Далі визначалась IP адреса Raspberry (рис. 2.5)

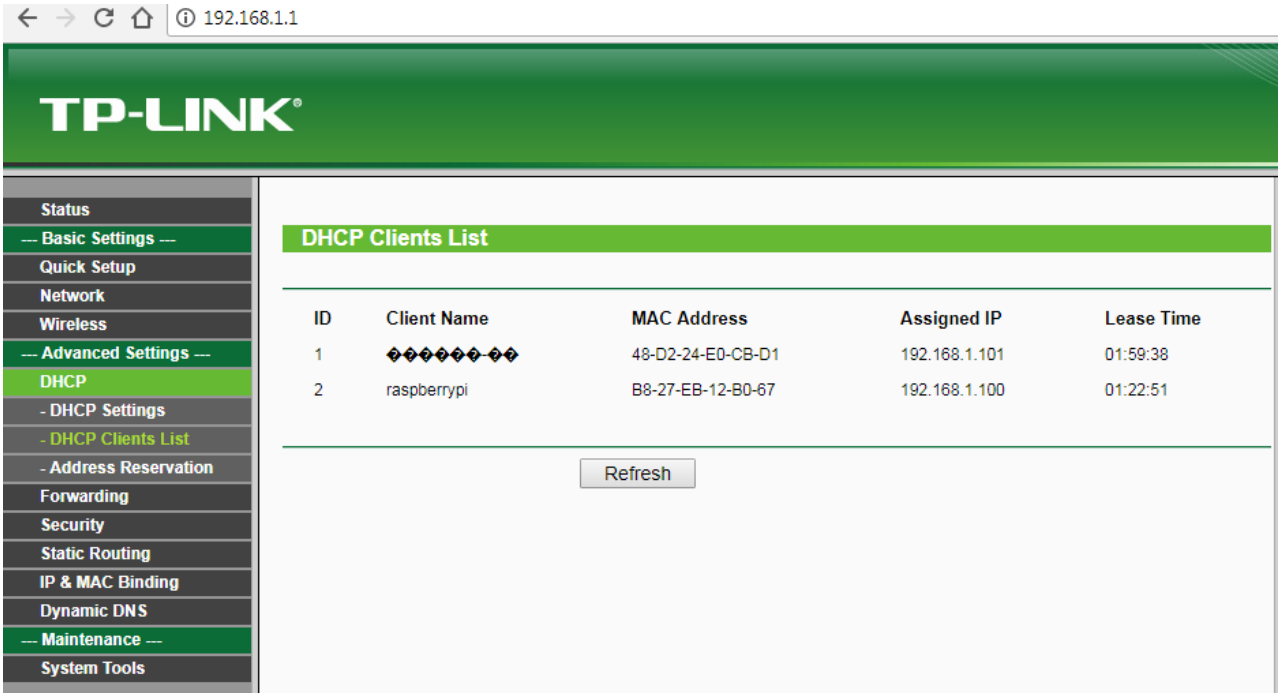


Рис. 2.5 – Визначення IP адреси Raspberry Pi

Для доступу за цією адресою використовувалась програма PuTTY. Для того аби на Raspberry Pi 1 раз в 2 секунди захоплювалось зображення, був написаний bash-скрипт (додаток 1). Зображення з камери бажано отримувати і обробляти в реальному часі, але для камери Raspberry Pi Camera Module існує технологічне обмеження – камера може захватувати зображення максимум 1 раз в 2 секунди. Цей час потрібен камері аби встановити її рівні освітленості. Приклад захопленого зображення – рис. 2.6.

Рис. 2.6 – Зображення робочої площадки отримане з камери

Доступ до даного зображення можливий за адресою < Raspberry_Pi_IP_address>:8080.

2.2.2 Обробка зображення та розпізнавання деталей

Була використана бібліотек OpenCV. Лістинг – додаток 2. Розпізнавалися темні деталі на світлому фоні. Для розпізнавання деталей використовувався детектор кордонів Кенні.

Спочатку в зображення збільшується яскравість для зменшення впливу шумів та можливого недостатнього освітлення. В результаті отримуємо картинку зображену на малюнку 2.7.

Рис. 2.7 – Картинка після застосування початкової згортки

Це здійснюється за допомогою згортки з матрицею:

$$\begin{pmatrix} -0.1 & 0.2 & -0.1 \\ 0.2 & 2.0 & 0.2 \\ -0.1 & 0.2 & -0.1 \end{pmatrix}$$

Рис. 2.8 – Інтерфейс програми

Інтерфейс даної програми можна побачити на рис. 2.8. Параметр aperture_size відповідає за розмір оператора Собеля (див. розділ 1 підпункт 1.4.1). aperture_size може приймати значення 1, 3, 5 або 7. Проте в програмі можна виставити значення цього параметру 3, 5 або 7:

```
if(pos>=3){
        if(pos%2==1)
        Hmin = pos;
        else Hmin = pos+1;
```

2 наступні параметри (threshold1 та treshold2) – пороги виділення границь. Якщо границі виділяються ледь помітно, різницю між цими дома параметрами треба збільшити. Проте чим більша різниця між ними, тим більші області виділяються як границі, що може призвести до хибного виділення об’єкту.

Параметри gridw і gridh – ширина і висота сітки. Ці параметри потрібні для визначення регіонів з деталями. Якщо одна деталь попаде відразу в декілька регіонів, то вона просто двічі розпізнається з близькими координатами. При переміщенні модулю, коли деталь попаде в той же регіон що і робочий носик, вона буде розпізнана як одна деталь з відповідними координатами. Якщо в один регіон попаде декілька деталей, то вони будуть неправильно розпізнані як одна. Тому цієї ситуації потрібно уникати. Тобто краще зробити сітку з меншими регіонами, але достатньо великими аби найбільша з деталей правильно розізналась в тому ж регіоні що і робочий носик. Зображення з накладеною сіткою на малюнку 2.9

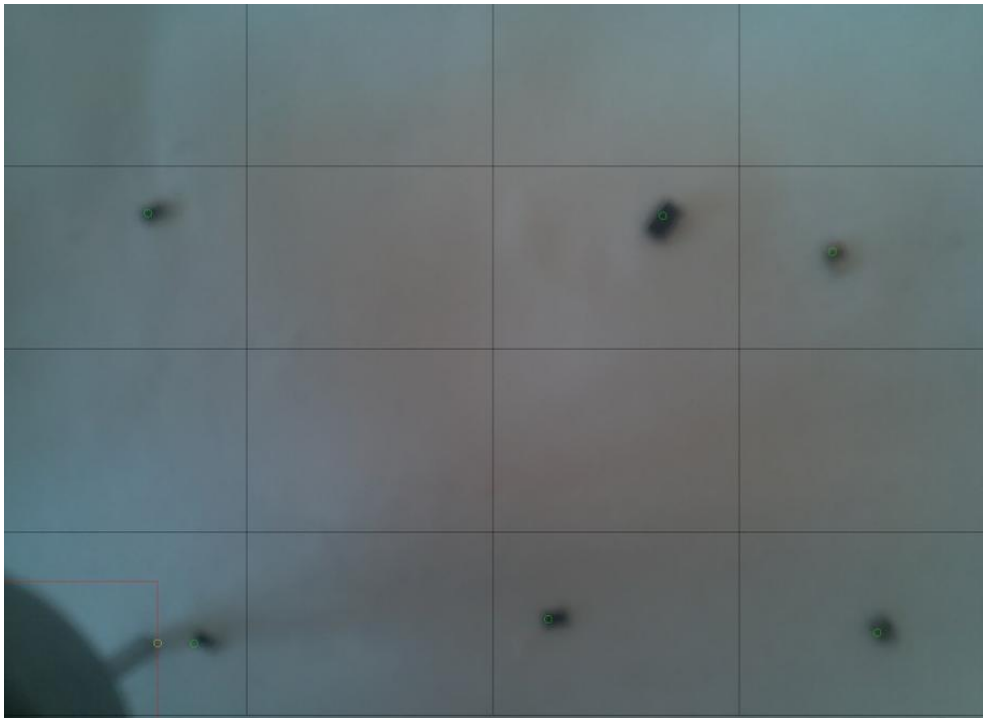


Рис. 2.9 – Зображення з накладеною сіткою

В результаті отримуємо зображення з границями (рис. 2.10)

Рис. 2.10 – Результуюче зображення

Для правильної обробки зображення необхідно з розглядуваних координат виключити координати з зображенням робочого носику. Це робиться наступним чином: на результуючому зображення кліком правої кнопки миші заходимо в режим pmp_mode (рис. 2.11). Далі клікаємо на кінці самого робочого носику, нижній горизонтальній границі носику та його крайній вертикальній границі. Отримана квадратна область (позначено червоним на малюнку 2.9) виключається з розпізнавання. Наступним кліком лівої кнопки миші можна вийти з режиму pmp_mode.

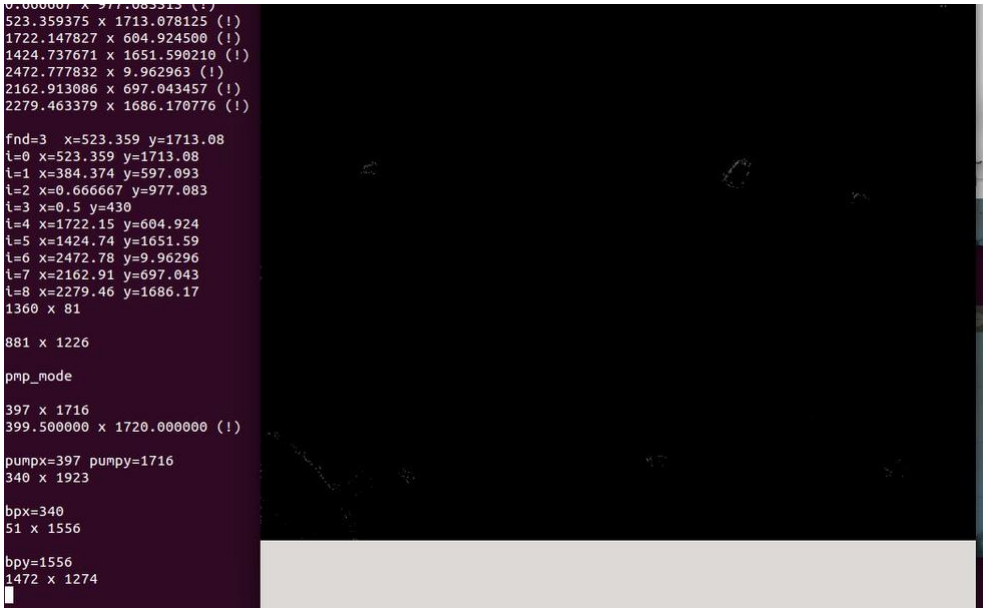


Рис. 2.11 – Виділення робочого носику

Також є можна визначити координату довільної точки на малюнку кліком лівої кнопки миші результуючому зображенні в відповідній точці (рис. 11).

В результаті на консоль виводяться координати всіх розпізнаних деталей, а також координати найближчої з них до робочого носику з рядком “fnd” (рис. 2.12).

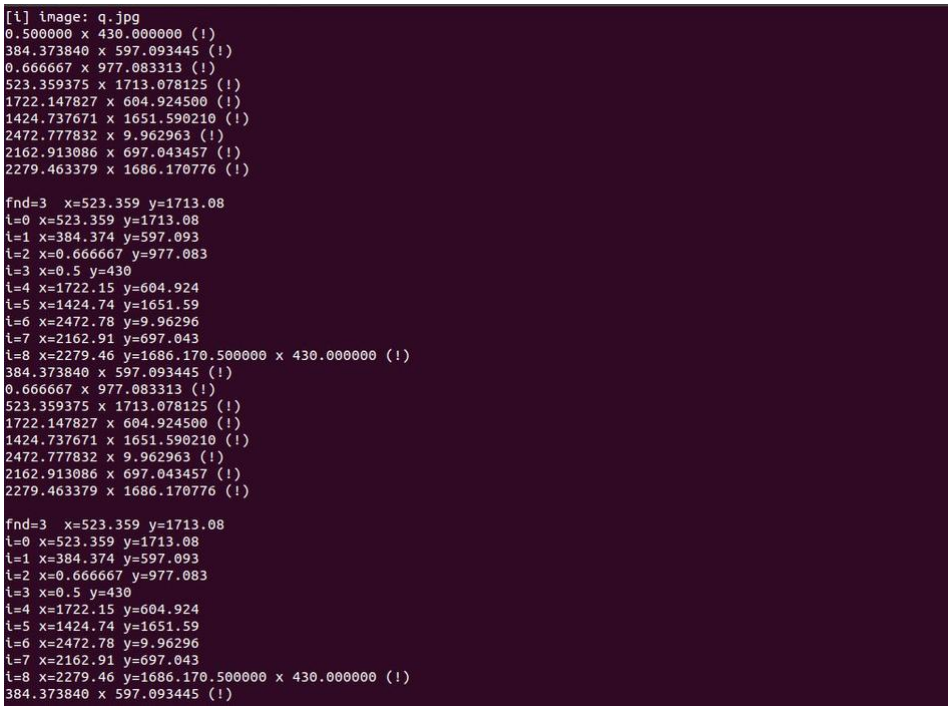


Рис. 2.12 Отримання координат зображення

На рис. 2.9 жовтим виділений кінець робочого носику, червоним область що виключається з розпізнавання, зеленим – розпізнані деталі. Центр границі деталі визначається як середнє арифметичне точок границі.

2.2.3 Керування кроковими двигунами

В роботі використовувались як біполярні так і і крокові двигуни. Лістинг – додаток 3

За керування біполярними кроковими двигунами відповідає клас my_bp. З початку цим двигунам необхідно привести положення робочих площадок в точку O(0,0,0). Це робить метод void my_bp::initb(). В цьому методі рух відбувається у напрямку механічних кінцевиків доки ці кінцевики не замкнуться. При замиканні вони подадуть відповідний сигнал на керуючий мікроконтролер, який по даному сигналу зупиняє рух відповідного крокового двигуна. Далі рух керується методом void my_bp::mv(unsigned int num,bool to),

де num – кількість кроків, to – напрям: якщо to==1, то це рух до відповідного механічного кінцевика, інакше – від нього.

За керування уніполярним крововим двигуном відповідає клас my_rot. Рух цього двигуна керується методом void my_rot::mv(unsigned int num,bool to), де num – кількість кроків, to – напрям: якщо to==1, то це рух за годинниковою стрілкою, інакше – проти годинникової стрілки.

2.3 Подальші перспективи

В подальшому варто об’єднати програмну і апаратну частину модулю та протестувати їх спільну роботу.

Також існує ряд варіантів для покращення модулю:

1) Налагодити роботу UART між комп’ютером і керуючим мікроконтролером.
2) Покращення розпізнавання деталей: спробувати відмовитися від сітки і розпізнавати деталі по ступеню близькості сусідніх граничних точок.
3) Розпізнавати деталі з побудовою SIFT(Scale-invariant feature transform) дескрипторів.
4) Виконувати обробку зображення та розпізнавання деталей на Raspberry Pi і відповідно налагодити роботу UART між Raspberry Pi та керуючим мікроконтролером.
5) Замінити Raspberry Pi та Raspberry Pi Camera Module відповідно на STM32 та OV7670 і відповідно налагодити роботу UART між STM32 та комп’ютером і комп’ютером та керуючим мікроконтролером.
6) Замінити 2 джерела живлення на 1 з використанням подільника напруги і, можливо, стабілітрона.

7) Написати нейронну мережу для переміщення робочих площадок.

8) Написати модуль для виділення з креслень трасувань плат координати, позицію та тип елемента.

2.4 Висновки до розділу 2

В даному розділі наведено опис реалізації апаратної та програмної частини модулю. Також було описані можливі доопрацювання та варіанти покращення модулю.

В першій частині наведені загальна схема модулю та опис взаємодії його частин. Також було наведені габарити робочих площадок, наведений опис складових частин модулю та бажаний алгоритм його роботи.

Друга частина присвячена опису програмної частини модулю. Було описано та обґрунтовано процес захвату зображення з камери, наведено опис програми обробки та розпізнавання деталей та керування кроковими двигунами. Були наведені пояснення до інтерфейсу та режимів роботи програми для виділення країв елементів.

В третій частині описані варіанти доопрацювання та покращення апаратної та програмної частин модулю.

3. ФУНКЦІОНАЛЬНО-ВАРТІСНИЙ АНАЛІЗ ПРОГРАМНОГО ПРОДУКТУ

У даному розділі проводиться оцінка основних характеристик програмного продукту призначеного для розпізнавання контурів smd деталей робочої площадки оператора. Інтерфейс користувача був розроблений за допомогою мови програмування C++ з використанням бібліотеки OpenCV.

Програмний продукт призначено для використання на персональних комп’ютерах під управлінням операційних системи Windows та Linux.

Нижче наведено аналіз різних варіантів реалізації модулю з метою вибору оптимальної, з огляду при цьому як на економічні фактори, так і на характеристики продукту, що впливають на продуктивність роботи і на його сумісність з апаратним забезпеченням. Для цього було використано апарат функціонально-вартісного аналізу.

Функціонально-вартісний аналіз (ФВА) – це технологія, яка дозволяє оцінити реальну вартість продукту або послуги незалежно від організаційної структури компанії. Як прямі, так і побічні витрати розподіляються по продуктам та послугам у залежності від потрібних на кожному етапі виробництва обсягів ресурсів. Виконані на цих етапах дії у контексті метода ФВА називаються функціями.

Мета ФВА полягає у забезпеченні правильного розподілу ресурсів, виділених на виробництво продукції або надання послуг, на прямі та непрямі витрати. У даному випадку – аналізу функцій програмного продукту й виявлення усіх витрат на реалізацію цих функцій.

Фактично цей метод працює за таким алгоритмом:

– визначається послідовність функцій, необхідних для виробництва продукту. Спочатку – всі можливі, потім вони розподіляються по двом групам: ті, що впливають на вартість продукту і ті, що не впливають. На цьому ж етапі оптимізується сама послідовність скороченням кроків, що не впливають на цінність і відповідно витрат.

– для кожної функції визначаються повні річні витрати й кількість робочих часів.

– для кожної функції на основі оцінок попереднього пункту визначається кількісна характеристика джерел витрат.

– після того, як для кожної функції будуть визначені їх джерела витрат, проводиться кінцевий розрахунок витрат на виробництво продукту.

3.1 Постановка задачі техніко-економічного аналізу

У роботі застосовується метод ФВА для проведення техніко-економічного аналізу розробки системи розпізнавання контурів smd деталей робочої площадки оператора. Оскільки основні проектні рішення стосуються всієї системи, кожна окрема підсистема має їм задовольняти.

Відповідно цьому варто обирати і систему показників якості програмного продукту.

Технічні вимоги до продукту наступні:

– програмний продукт повинен функціонувати на персональних комп’ютерах із стандартним набором компонент;

– забезпечувати високу швидкість обробки даних;

– забезпечувати зручність і простоту взаємодії з користувачем або з розробником програмного забезпечення у випадку використовування його як модуля;

– забезпечувати можливість автономної роботи програмної частини модулю для її тестування та налагодження;

– передбачати мінімальні витрати на впровадження програмного продукту.

3.1.1 Обґрунтування функцій програмного продукту

Головна функція F0– розробка програмного продукту, який отримує зображення робочої площадки оператора з smd компонентами, та визначає координати їх границь. Виходячи з конкретної мети, можна виділити наступні основні функції ПП:

F1 – вибір мови програмування;

F2 – спосіб отримання вхідних даних (зображення робочої площадки);

F3 – інтерфейс користувача.

Кожна з основних функцій може мати декілька варіантів реалізації.

Функція F1:

а) мова програмування C++;

б) мова програмування Java;

Функція F2:

а) отримання зображення з командного рядку;

б) написання нового модулю безпосереднього обміну даними мікроконтроллеру з камерою і комп’ютером.

Функція F3:

а) інтерфейс користувача, створений за технологією Windows Forms;

б) інтерфейс користувача, створений з використанням бібліотеки OpenCV.

3.1.2 Варіанти реалізації основних функцій

Варіанти реалізації основних функцій наведені у морфологічній карті системи (рис. 2.1). На основі цієї карти побудовано позитивно-негативну матрицю варіантів основних функцій (таблиця 3.1).

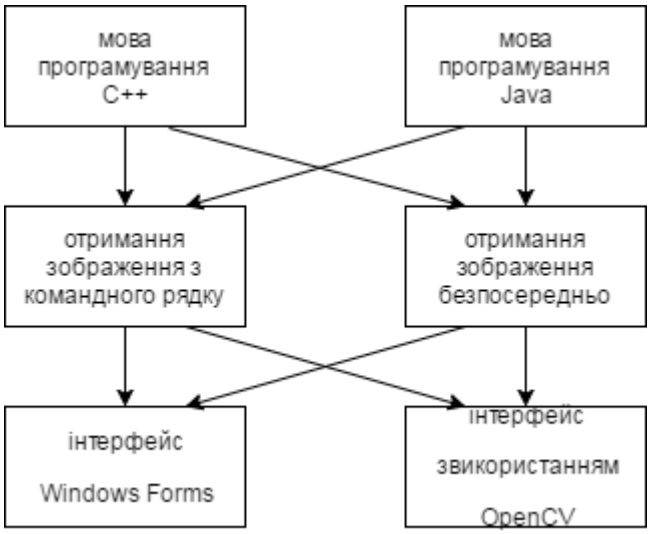


Рис. 3.1 – Морфологічна карта

Морфологічна карта відображує всі можливі комбінації варіантів реалізації функцій, які складають повну множину варіантів ПП.

Таблиця 3.1 – Позитивно-негативна матриця

Основні функції	Варіанти реалізації	Переваги	Недоліки
F1	*А*	Займає менше часу при написанні коду	Код швидко виконується
	Б	Кросплатформений код	Займає більше часу при написанні коду
F2	А	Простий у використанні	Необхідно додатково автоматизоввати процес написанням відповідного скрипту
	Б	Повна автоматизація отримання вхідних даних	Налагодження та тестування відбувається за умови повної збірки модулю
F3	*А*	Легкий у створенні	Відсутність кросплатформеності
	Б	Основні компоненти для розроби вже містяться в бібліотеці OpenCV	Потрібно додатково вивчати документацію

На основі аналізу позитивно-негативної матриці робимо висновок, що при розробці програмного продукту деякі варіанти реалізації функцій варто відкинути, тому, що вони не відповідають поставленим перед програмним продуктом задачам. Ці варіанти відзначені у морфологічній карті.

Функція F1:

Оскільки час виконання програмного коду є важливою характеристикою, варіант б) має бути відкинутий.

Функція F2:

Необхідно забезпечувати можливість автономної роботи програмної частини модулю для її тестування та налагодження,але в процесі експлуатації краще мати максимально автоматизованих процес тому вважаємо варіанти а) та б) гідними розгляду.

Функція F3:

Інтерфейс користувача не відіграє велику роль у даному програмному продукту, але необхідно забезпечити кросплатформенність, тому обираємо варіант б).

Таким чином, будемо розглядати такі варіанти реалізації ПП:

1. F1а – F2а – F3б

2. F1а – F2б – F3б

Для оцінювання якості розглянутих функцій обрана система параметрів, описана нижче.

3.2 Обґрунтування системи параметрів ПП

3.2.1 Опис параметрів

На підставі даних про основні функції, що повинен реалізувати програмний продукт, вимог до нього, визначаються основні параметри виробу, що будуть використані для розрахунку коефіцієнта технічного рівня.

Для того, щоб охарактеризувати програмний продукт, будемо використовувати наступні параметри:

– X1 – швидкодія мови програмування;

– X2 – об’єм пам’яті для збереження даних;

– X3 – час обробки даних;

– X4 – потенційний об’єм програмного коду.

X1: Відображає швидкодію операцій залежно від обраної мови програмування.

X2: Відображає об’єм пам’яті в оперативній пам’яті персонального комп’ютера, необхідний для збереження та обробки даних під час виконання програми.

X3: Відображає час, який витрачається на дії.

X4: Показує розмір програмного коду який необхідно створити безпосередньо розробнику.

3.2.2 Кількісна оцінка параметрів

Гірші, середні і кращі значення параметрів вибираються на основі вимог замовника й умов, що характеризують експлуатацію ПП як показано у табл. 3.2.

За даними таблиці 3.2 будуються графічні характеристики параметрів – рис. 3.2 – рис. 3.5.

Таблиця 3.2 – Основні параметри ПП

Назва Параметра	**Умовні позначення**	**Одиниці виміру**	**Значення параметра**		
			гірші	**середні**	**кращі**
Швидкодія мови програмування	X1	Оп/мс	19000	11000	2000
Об’єм пам’яті для збереження даних	X2	Мб	32	16	8
Час обробки даних алгоритмом	X3	мс	10000	2000	500
Потенційний об’єм програмного коду	X4	кількість рядків коду (в сумі)	2000	1500	1000

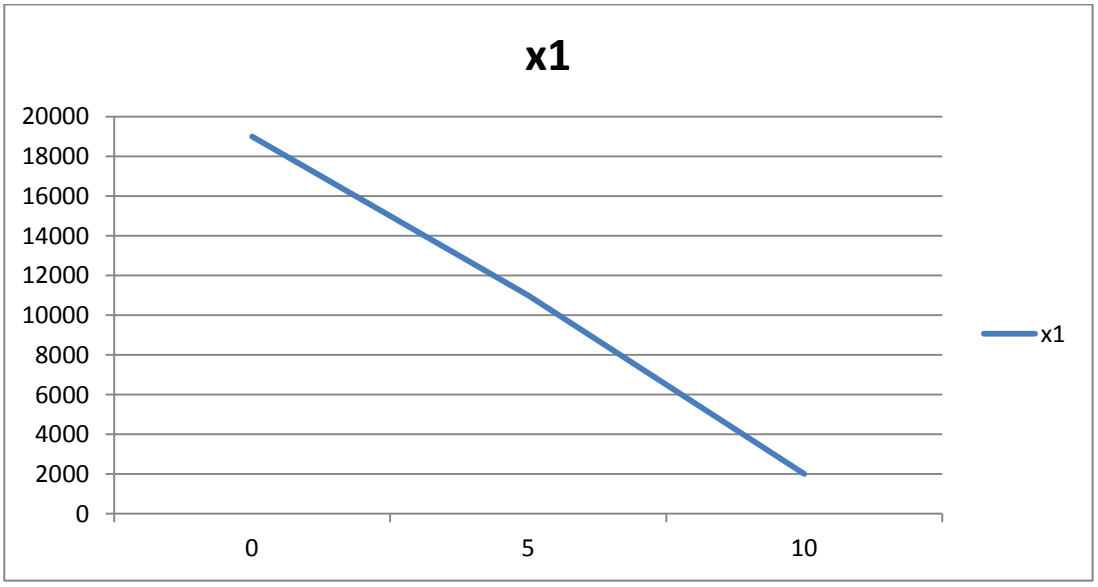


Рис. 3.2 – X1, швидкодія мови програмування

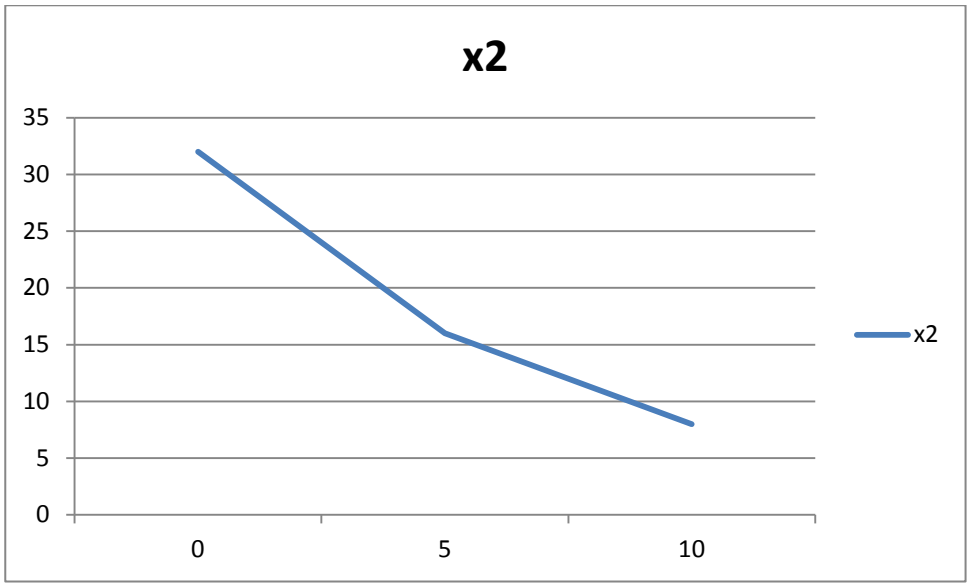


Рис. 3.3 – X2, об’єм пам’яті для збереження даних

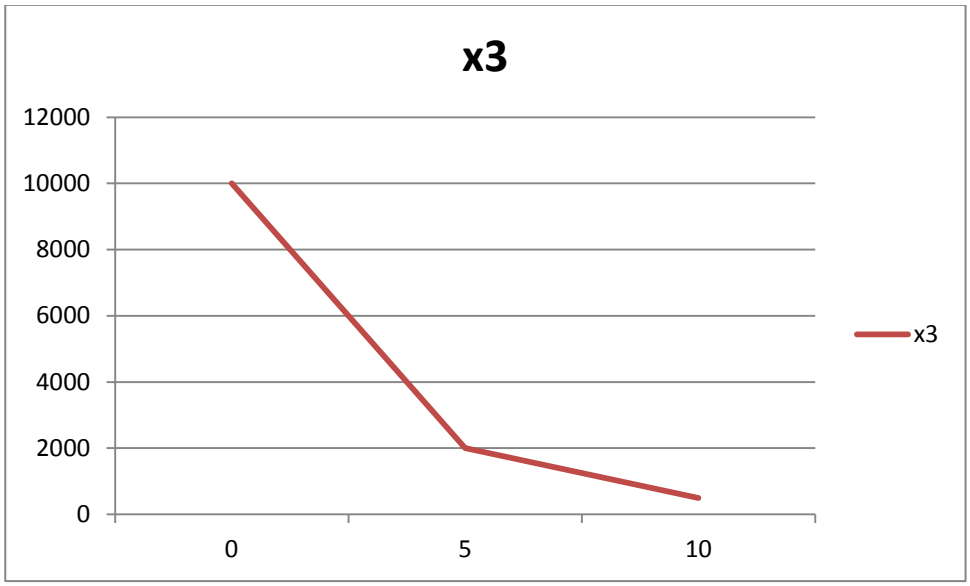


Рисунок 3.4 – X3, час обробки даних алгоритмом

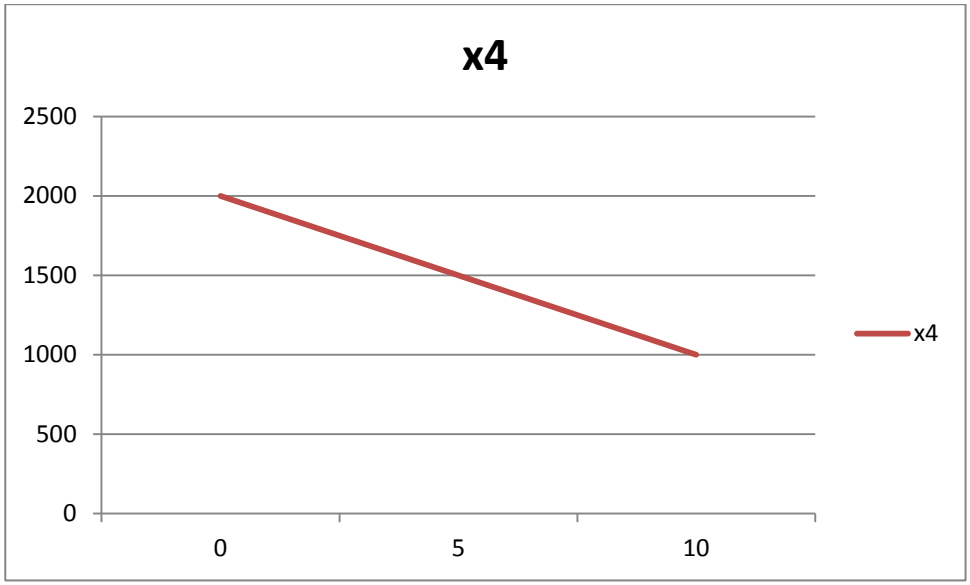


Рисунок 3.5 – X4, потенційний об’єм програмного коду

3.2.3 Аналіз експертного оцінювання параметрів

Після детального обговорення й аналізу кожний експерт оцінює ступінь важливості кожного параметру для конкретно поставленої цілі – розробка програмного продукту, який дає найбільш точні результати при знаходженні параметрів моделей адаптивного прогнозування і обчислення прогнозних значень.

Значимість кожного параметра визначається методом попарного порівняння. Оцінку проводить експертна комісія із 7 людей. Визначення коефіцієнтів значимості передбачає:

– визначення рівня значимості параметра шляхом присвоєння різних рангів;

– перевірку придатності експертних оцінок для подальшого використання;

– визначення оцінки попарного пріоритету параметрів;

– обробку результатів та визначення коефіцієнту значимості.

Результати експертного ранжування наведені у таблиці 3.3.

Таблиця 3.3 – Результати ранжування параметрів

Позначення параметра	Назва параметра	Одиниці виміру	Ранг параметра за оцінкою експерта							Сума рангів R_i	Відхилення Δ_i	Δ_i^2
			1	2	3	4	5	6	7			
X1	Швидкодія мови програмування	Оп/мс	4	3	4	4	4	4	4	27	0,75	0,56
X2	Об’єм пам’яті для збереження даних	Мб	4	4	4	3	4	3	3	25	-1,25	1,56
X3	Час обробки даних алгоритмом	Мс	2	2	1	2	1	2	2	12	-14,25	203,06
X4	Потенційний об’єм програмного коду	кількість рядків коду	5	6	6	6	6	6	6	41	14,75	217,56
	Разом		15	15	15	15	15	15	15	105	0	420,75

Для перевірки степені достовірності експертних оцінок, визначимо наступні параметри:

а) сума рангів кожного з параметрів і загальна сума рангів:

$$R_i = \sum_{j=1}^{N} r_{ij} R_{ij} = \frac{Nn(n+1)}{2} = 105,$$

де N – число експертів, n – кількість параметрів;

б) середня сума рангів:

$$T = \frac{1}{n} R_{ij} = 26{,}25.$$

в) відхилення суми рангів кожного параметра від середньої суми рангів:

$$\Delta_i = R_i - T$$

Сума відхилень по всім параметрам повинна дорівнювати 0;

г)загальна сума квадратів відхилення:

$$S = \sum_{i=1}^{N} \Delta_i^2 = 420{,}75.$$

Порахуємо коефіцієнт узгодженості:

$$W = \frac{12S}{N^2(n^3 - n)} = \frac{12 \cdot 420{,}75}{7^2(5^3 - 5)} = 1{,}03 > W_k = 0{,}67$$

Ранжування можна вважати достовірним, тому що знайдений коефіцієнт узгодженості перевищує нормативний, котрий дорівнює 0,67.

Скориставшись результатами ранжирування, проведемо попарне порівняння всіх параметрів і результати занесемо у таблицю 3.4.

Таблиця 3.4 – Попарне порівняння параметрів

Параметри	Експерти							Кінцева оцінка	Числове значення
	1	2	3	4	5	6	7		
X1 і X2	=	>	=	<	=	<	<	<	0,5
X1 і X3	<	<	<	<	<	<	<	<	0,5
X1 і X4	>	>	>	>	>	>	>	>	1,5
X2 і X3	<	<	<	<	<	<	<	<	0,5
X2 і X4	>	>	>	>	>	>	>	>	1,5
X3 і X4	>	>	>	>	>	>	>	>	1,5

Числове значення, що визначає ступінь переваги *i*–го параметра над *j*–тим, a_{ij}визначається по формулі:

$$a_{ij} = \begin{cases} 1{,}5 \text{ при } X_i > X_j \\ 1.0 \text{ при } X_i = X_j \\ 0.5 \text{ при } X_i < X_j \end{cases}$$

З отриманих числових оцінок переваги складемо матрицю $A = \| a_{ij} \|$.

Для кожного параметра зробимо розрахунок вагомості $K_{вi}$ за наступними формулами:

$$K_{\text{вi}} = \frac{b_i}{\sum_{i=1}^{n} b_i}, \text{де } b_i = \sum_{i=1}^{N} a_{ij}.$$

Відносні оцінки розраховуються декілька разів доти, поки наступні значення не будуть незначно відрізнятися від попередніх (менше 2%).На другому і наступних кроках відносні оцінки розраховуються за наступними формулами:

$K_{\text{ві}} = \frac{b_i'}{\sum_{i=1}^{n} b_i'}$,де $b_i' = \sum_{i=1}^{N} a_{ij} b_j$.

Як видно з таблиці 3.5, різниця значень коефіцієнтів вагомості не перевищує 2%, тому більшої кількості ітерацій не потрібно.

Таблиця 3.5 – Розрахунок вагомості параметрів

Параметриx_i	Параметриx_j				Перша ітер.		Друга ітер.		Третя ітер	
	X1	X2	X3	X4	b_i	$K_{\text{ві}}$	b_i^1	$K_{\text{ві}}^1$	b_i^2	$K_{\text{ві}}^2$
X1	1,0	0,5	0,5	1,5	3,5	0,219	22,25	0,216	100	0,215
X2	1,5	1,0	0,5	1,5	4,5	0,281	27,25	0,282	124,25	0,283
X3	1,5	1,5	1,0	1,5	5,5	0,344	34,25	0,347	156	0,348
X4	0,5	0,5	0,5	1,0	2,5	0,156	14,25	0,155	64,75	0,154
Всього:					16	1	98	1	445	1

3.3 Аналіз рівня якості варіантів реалізації функцій

Визначаємо рівень якості кожного варіанту виконання основних функцій окремо.

Абсолютні значення параметрів *X2*(об’єм пам’яті для збереження даних) та *X1* (швидкодія мови програмування) відповідають технічним вимогам умов функціонування даного ПП.

Абсолютне значення параметра *X3 (*час обробки даних) обрано не найгіршим (не максимальним), тобто це значення відповідає або варіанту а) 2000 мс або варіанту б) 500мс.

Коефіцієнт технічного рівня для кожного варіанта реалізації ПП розраховується так (таблиця 6):

$$K_K(j) = \sum_{i=1}^{n} K_{\text{в}i,j} B_{i,j},$$

де n – кількість параметрів;$K_{\text{в}i}$– коефіцієнт вагомості i–го параметра;B_i – оцінка i–го параметра в балах.

Таблиця 3.6 – Розрахунок показників рівня якості варіантів реалізації основних функцій ПП

Основні функції	Варіант реалізації функції	Абсолютне значення параметра	Бальна оцінка параметра	Коефіцієнт вагомості параметра	Коефіцієнт рівня якості
F1(X1)	А	11000	3,6	0,215	0,774
F3(X2)	А	16	3,4	0,283	0,962
F2(X3,X4)	А	2000	2,4	0,389	0,934
	Б	500	1	0,054	0,054

За даними з таблиці 6 за формулою

$$K_K = K_{\text{ТУ}}[F_{1k}] + K_{\text{ТУ}}[F_{2k}] + \ldots + K_{\text{ТУ}}[F_{zk}],$$

визначаємо рівень якості кожного з варіантів:

K_{K1} = 0,774 + 0,962 + 0,934 = 2,67

K_{K2} = 0,774 + 0,962 + 0,054 = 1,79

Як видно з розрахунків, кращим є перший варіант, для якого коефіцієнт технічного рівня має найбільше значення.

3.4 Економічний аналіз варіантів розробки ПП

Для визначення вартості розробки ПП спочатку проведемо розрахунок трудомісткості.

Всі варіанти включають в себе два окремих завдання:

1. Розробка проекту програмного продукту;

2. Розробка програмної оболонки;

Завдання 1 за ступенем новизни відноситься до групи А, завдання 2 – до групи Б. За складністю алгоритми, які використовуються в завданні 1 належать до групи 1; а в завданні 2 – до групи 3.

Для реалізації завдання 1 використовується довідкова інформація, а завдання 2 використовує інформацію у вигляді даних.

Проведемо розрахунок норм часу на розробку та програмування для кожного з завдань.

Проведемо розрахунок норм часу на розробку та програмування для кожного з завдань. Загальна трудомісткість обчислюється як

ТО = ТР· КП· КСК· КМ· КСТ· КСТ.М, (5.1)

де ТР – трудомісткість розробки ПП;КП – поправочний коефіцієнт;КСК – коефіцієнт на складність вхідної інформації; КМ – коефіцієнт рівня мови програмування;КСТ – коефіцієнт використання стандартних модулів і прикладних програм;КСТ.М – коефіцієнт стандартного математичного забезпечення

Для першого завдання, виходячи із норм часу для завдань розрахункового характеру степеню новизни А та групи складності алгоритму 1, трудомісткість дорівнює: $T_Р$ =90 людино-днів. Поправочний коефіцієнт, який враховує вид нормативно-довідкової інформації для першого завдання: $К_П$ = 1.7. Поправочний коефіцієнт, який враховує складність контролю вхідної та вихідної інформації для всіх семи завдань рівний 1: $К_{СК}$ = 1. Оскільки при розробці першого завдання використовуються стандартні модулі, врахуємо це за допомогою коефіцієнта $К_{СТ}$ = 0.8. Тоді, за формулою 5.1, загальна трудомісткість програмування першого завдання дорівнює:

T_1 = 90·1.7·0.8 = 122.4 людино-днів.

Проведемо аналогічні розрахунки для подальших завдань.

Для другого завдання (використовується алгоритм третьої групи складності, степінь новизни Б), тобто $T_Р$ =27 людино-днів, $К_П$ =0.9,$К_{СК}$ = 1,$К_{СТ}$ =0.8:

T_2 = 27 · 0.9 · 0.8 = 19.44 людино-днів.

Складаємо трудомісткість відповідних завдань для кожного з обраних варіантів реалізації програми, щоб отримати їх трудомісткість:

T_I = (122.4 + 19.44 + 4.8 + 19.44) · 8 = 1328,64 людино-годин;

T_{II} = (122.4 + 19.44 + 6.91 + 19.44) · 8 = 1345.52 людино-годин;

Найбільш високу трудомісткість має варіант ІІ.

В розробці беруть участь два програмісти з окладом 6000 грн., один фінансовий аналітик з окладом 9000грн. Визначимо зарплату за годину за формулою:

$$\text{СЧ} = \frac{\text{М}}{T_m \cdot t} \text{грн.},$$

де М – місячний оклад працівників;T_m – кількість робочих днів тиждень;t – кількість робочих годин в день.

$$\text{СЧ} = \frac{6000 + 6000 + 9000}{3 \cdot 21 \cdot 8} = 41{,}67 \text{ грн.}$$

Тоді, розрахуємо заробітну плату за формулою

$$\text{СЗП} = \text{С}_\text{ч} \cdot \text{Т}_i \cdot \text{КД},$$

де $\text{С}_\text{ч}$– величина погодинної оплати праці програміста;Т_i – трудомісткість відповідного завдання; $\text{К}_\text{Д}$ – норматив, який враховує додаткову заробітну плату.

Зарплата розробників за варіантами становить:

І. $\text{С}_\text{ЗП} = 41{,}67 \cdot 1328.64 \cdot 1.2 = 66437{,}31$ грн.

ІІ. $\text{С}_\text{ЗП} = 41{,}67 \cdot 1345.52 \cdot 1.2 = 67281{,}38$ грн.

Відрахування на єдиний соціальний внесок в залежності від групи професійного ризику (ІІ клас) становить 22%:

І. $\text{С}_\text{ВІД} = \text{С}_\text{ЗП} \cdot 0.22 = 66437{,}31 * 0.22 = 14616$ грн.

ІІ. $\text{С}_\text{ВІД} = \text{С}_\text{ЗП} \cdot 0.22 = 67281{,}38 * 0.22 = 14802$ грн.

Тепер визначимо витрати на оплату однієї машино-години. ($\text{С}_\text{М}$)

Так як одна ЕОМ обслуговує одного програміста з окладом 6000 грн., з коефіцієнтом зайнятості 0,2 то для однієї машини отримаємо:

$С_Г = 12 \cdot М \cdot К_З = 12 \cdot 6000 \cdot 0,2 = 14400$ грн.

З урахуванням додаткової заробітної плати:

$С_{ЗП} = С_Г \cdot (1 + К_З) = 14400 \cdot (1 + 0.2) = 17280$ грн.

Відрахування на єдиний соціальний внесок:

$С_{ВІД} = С_{ЗП} \cdot 0.22 = 17280 \cdot 0,22 = 3801,6$ грн.

розраховуємо при амортизації 25% та вартості ЕОМ – 8000 грн.

$С_А = К_{ТМ} \cdot К_А \cdot Ц_{ПР} = 1.15 \cdot 0.25 \cdot 8000 = 2300$ грн.,

де $К_{ТМ}$– коефіцієнт, який враховує витрати на транспортування та монтаж приладу у користувача; $К_А$– річна норма амортизації; $Ц_{ПР}$– договірна ціна приладу.

Витрати на ремонт та профілактику розраховуємо як:

$С_Р = К_{ТМ} \cdot Ц_{ПР} \cdot К_Р = 1.15 \cdot 8000 \cdot 0.05 = 460$ грн.,

де $К_Р$– відсоток витрат на поточні ремонти.

Ефективний годинний фонд часу ПК за рік розраховуємо за формулою:

$Т_{ЕФ} = (Д_К - Д_В - Д_С - Д_Р) \cdot t_З \cdot К_В = (365 - 104 - 8 - 16) \cdot 8 \cdot 0.9 = 1706.4$ годин,

де $Д_К$ – календарна кількість днів у році; $Д_В$, $Д_С$ – відповідно кількість вихідних та святкових днів; $Д_Р$ – кількість днів планових ремонтів устаткування; t –кількість робочих годин в день; $К_В$– коефіцієнт використання приладу у часі протягом зміни.

Витрати на оплату електроенергії розраховуємо за формулою:

$С_{ЕЛ} = Т_{ЕФ} \cdot N_С \cdot К_З \cdot Ц_{ЕН} = 1706,4 \cdot 0,156 \cdot 1,93819 = 515,94$ грн.,

де $N_С$ – середньо-споживча потужність приладу; $К_З$– коефіцієнтом зайнятості приладу; $Ц_{ЕН}$ – тариф за 1 КВт-годин електроенергії.

Накладні витрати розраховуємо за формулою:

$С_Н = Ц_{ПР} \cdot 0.67 = 8000 \cdot 0,67 = 5360$ грн.

Тоді, річні експлуатаційні витрати будуть:

$С_{ЕКС} = С_{ЗП} + С_{ВІД} + С_А + С_Р + С_{ЕЛ} + С_Н$

$С_{ЕКС} = 17280 + 3801,6 + 2300 + 460 + 515,94 + 5360 = 29717,54$ грн.

Собівартість однієї машино-години ЕОМ дорівнюватиме:

$С_{М-Г} = С_{ЕКС} / Т_{ЕФ} = 29717,54/1706,4 = 17,4$ грн/час.

Оскільки в даному випадку всі роботи, які пов‘язані з розробкою програмного продукту ведуться на ЕОМ, витрати на оплату машинного часу, в залежності від обраного варіанта реалізації, складає:

$С_М = С_{М-Г} \cdot Т$

I. $С_М = 18,91 * 1328,64 = 25124,58$ грн.;

II. $С_М = 18,91 * 1345.52 = 25443,78$ грн.;

Накладні витрати складають 67% від заробітної плати:

$С_Н = С_{ЗП} \cdot 0,67$

I. $С_Н = 66437,31 \cdot 0,67 = 44513$ грн.;

II. $С_Н = 67281,38 \cdot 0,67 = 45078,52$ грн.;

Отже, вартість розробки ПП за варіантами становить:

$С_{ПП} = С_{ЗП} + С_{ВІД} + С_{М} + С_{Н}$

I. $С_{ПП} = 66437,31 + 24429 + 25124,58 + 44513 = 160503,89$ грн.;

II. $С_{ПП} = 67281,38 + 24739 + 25443,78 + 45078,52 = 162542,68$ грн.;

3.5 Вибір кращого варіанта ПП техніко-економічного рівня

Розрахуємо коефіцієнт техніко-економічного рівня за формулою:

КТЕРj =ККj/СФj,

КТЕР1 = 5,214 / 160503,89 = 0,32·10-4;

КТЕР2 = 3,569 / 162542,68 = 0,21·10-4;

Як бачимо, найбільш ефективним є перший варіант реалізації програми з коефіцієнтом техніко-економічного рівня КТЕР1= 0,33· 10-4.

3.6 Висновки до розділу 3

В даному розділі проведено повний функціонально-вартісний аналіз ПП, який було розроблено в рамках дипломного проекту. Процес аналізу можна умовно розділити на дві частини.

В першій з них проведено дослідження ПП з технічної точки зору: було визначено основні функції ПП та сформовано множину варіантів їх реалізації; на основі обчислених значень параметрів, а також експертних оцінок їх важливості було обчислено коефіцієнт технічного рівня, який і дав змогу визначити оптимальну з технічної точки зору альтернативу реалізації функцій ПП.

Другу частину ФВА присвячено вибору із альтернативних варіантів реалізації найбільш економічно обґрунтованого. Порівняння запропонованих варіантів реалізації в рамках даної частини виконувалось за коефіцієнтом ефективності, для обчислення якого були обчислені такі допоміжні параметри, як трудомісткість, витрати на заробітну плату, накладні витрати.

Після виконання функціонально-вартісного аналізу програмного комплексу що розроблюється, можна зробити висновок, що з альтернатив, що залишились після першого відбору двох варіантів виконання програмного комплексу оптимальним є перший варіант реалізації програмного продукту. У нього виявився найкращий показник техніко-економічного рівня якості

КТЕР = 0,32· 10-4.

Цей варіант реалізації програмного продукту має такі параметри:

– мова програмування – C++;

– отримання зображення з командного рядку;

– інтерфейс користувача, створений з використанням бібліотеки OpenCV.

Даний варіант виконання програмного комплексу дає користувачу зручний інтерфейс, непоганий функціонал і швидкодію.

ВИСНОВКИ

В результаті виконання роботи було розроблено програму для керування крововими двигунами для переміщення площадок по осям та позиціонування деталі. також було розроблено програму для розпізнавання границь елементів та визначення найближчого до робочого носику елемента.

Вданій роботі є три розділи основної часини.

В першому розділі в першій частині проводиться аналіз існуючих аналогів, їх класифікація, опис характеристик та технічних можливостей. В другій частині проводиться класифікація видів 3D принтерів за кінематичною схемою та технологією друку. Розглядаються їх переваги, недоліки та області застосування. В третій частині наводиться класифікація крокових двигунів за типом з’єднання обмоток. В четвертій частині наводиться інформація про визначення контурів об’єктів за допомогою детектору кордонів Кенні (Canny) та його математична формалізація.

В другому розділі наводиться опис практичної реалізації модулю та подальші перспективи його розробки. В першій частині даного розділу наводиться опис реалізації апаратної частини модулю, її загальна схема та зовнішній вигляд. Також в цій частині наводиться перелік компонентів та способи їх взаємодії. В другій частині описується реалізація програмної частини модулю: спосіб отримання зображень його та технічні обмеження, опис програми розпізнавання елементів та її інтерфейсу з поясненнями та ілюстраціями, а також опис програми для керування крововими двигунами. Відповідні лістинги наводяться в додатках. В третій частині наводяться варіанти можливих доопрацювань та способи покращення модулю.

В третьому розділі описується економічна частина. В ньому обґрунтовується вибір варіанту реалізації програми для розпізнавання границь

деталей: мова програмування (C++), спосіб вводу зображення (з командного рядку) та спосіб створення інтерфейсу користувача (з використанням бібліотеки OpenCV).

ПЕРЕЛІК ПОСИЛАНЬ

1. Компьтерное зрение.Современний подход/Форсайт Д., Понс Ж.,:Пер. с англ. – М.:Издательский дом «Вильямс», 2004. – 928с.
2. Цифровая обработка видеоизображений / А. А.Лукьяница,А.Г.Шишкин. – М.: «Ай-Эс-Эс Пресс», 2009. – 518с.
3. Principles of filter design. In Handbook of Computer Vision and Applications/ B. Jähne, H. Scharr, and S. Körkel. – Academic Press, 1999 – 234 c.
4. Stepping Motors and Their Microprocessor Controls, 2nd Edition, Oxford/T. Kenjo, A. Sugawara, – University Press, Oxford, 2003 – 292 c.
5. Stepping Motors – A guide to theory and practice, 4th Edition/ P. Acarnley, – The Institution of Electrical Engineers, Лондон, 2002 – 170 c.
6. Установка компонентів на друкованих платах [Електронний ресурс] – Режим доступу: http://www.geoin.org/design/lecture/app/lec15.doc.– Дата доступу : 14.06.2017.
7. Монтаж друкованих плат [Електронний ресурс] – Режим доступу: https://xn--80avlo9b.xn--p1ai/?pcb-mounting,15 https://xn--80avlo9b.xn--p1ai/?pcb-mounting,15– Дата доступу : 14.06.2017.
8. Автоматичне встановлення компонентів на друковану плату в технології поверхневого монтажу [Електронний ресурс] – Режим доступу: http://pcbdesigner.ru/pcb/montazh-pechatnykh-plat/avtomaticheskaya-ustanovka-komponentov-na-pechatnuyu-platu-v-texnologii-poverxnostnogo-montazha.html – Дата доступу : 14.06.2017.

9. Основи технології та обладнання для поверхневого монтажу [Електронний ресурс] – Режим доступу: http://www.elinform.ru/articles_4.htm – Дата доступу : 14.06.2017.

10. Технологія поверхневого монтажу [Електронний ресурс] – Режим доступу: http://kkbweb.narod.ru/teoriya/smt_tehnology.htm – Дата доступу : 14.06.2017.

11. Класифікація 3D принтерів [Електронний ресурс] – Режим доступу: https://geektimes.ru/post/208906/ – Дата доступу : 14.06.2017.

12. Обробка зображення – оператори Собеля і Лапласа [Електронний ресурс] – Режим доступу: http://robocraft.ru/blog/computervision/460.html – Дата доступу : 14.06.2017.

13. Обробка зображення – детектор кордонів Кенні (Canny) [Електронний ресурс] – Режим доступу: http://robocraft.ru/blog/computervision/484.html – Дата доступу: 14.06.2017.

14. Пашин В.П. Управление качеством изделий на основе функционально-стоимостного анализа / Пашин В.П. // Технология и организация производства. — 1989. – № 12.— С. 17-19.

15. Пашін В.П. Оцінка конкурентоспроможності електронних пристроїв на стадії проектування : Пашін В.П., Бровкін А. Г.,. Павлуша І. А. // Економічний вісник. — 2006. — № 4. — С. 56-65.

ДОДАТОК А

Код програми для отримання зображень з Raspberry Pi та Raspberry Pi Camera Module з інтервалом в 2 секунди.

```
#!/bin/bash

while true

do

    raspistill -o ./www/pic.jpg

    echo "pic.jpg done"

    sleep 2

done
```

ДОДАТОК Б

Код програми обробки зображення та розпізнавання контурів деталей

```
#include <cv.h>

#include <highgui.h>

#include <stdlib.h>

#include <stdio.h>

#include<math.h>

#include<vector>

#include<iostream>

#include <algorithm>



#define qmode

#define sh_coords

#define dvdr 10


typedef unsigned short int usi;


IplImage* image = 0;

IplImage* gray = 0, *gray1 = 0;
```

```
IplImage* dst = 0;
int Hmin=3;
int threshold1=91;
int threshold2=120;
int ngridw=640;//=125;
int ngridh=475;//=47;
unsigned int pumpx=395,pumpy=1715,bpx=350,bpy=1550;
unsigned char ispmp=0;

struct crd{
  float x,y;
  bool inR(unsigned int rds){
        return (sqrt((x-pumpx)*(x-pumpx)+(y-pumpy)*(y-pumpy))<rds);
  }
  friend std::ostream& operator<<(std::ostream& os, const crd& curcrd){
        os<<" x="<<curcrd.x<<" y="<<curcrd.y;
        return os;
  }
};
std::vector<crd> crds;
```

```
void myTrackbarHmin(int pos) {
  if(pos>=3){
        if(pos%2==1)
        Hmin = pos;
        else Hmin = pos+1;
        cvCanny(gray, dst, threshold1, threshold2, Hmin);
    }
}

void threshold1f(int pos){
  threshold1=pos;
  cvCanny(gray, dst, threshold1, threshold2, Hmin);
}

void threshold2f(int pos){
  threshold2=pos;
  cvCanny(gray, dst, threshold1, threshold2, Hmin);
}
```

```
// рисуем целеуказатель
void drawTarget(IplImage* img, int x, int y, int radius)
{
    //cvCircle(gray1,cvPoint(x, y),radius,CV_RGB(1,0,0),1,8);
    //cvLine(img, cvPoint(x-radius/2, y-radius/2), cvPoint(x+radius/2, y+radius/2),CV_RGB(250,0,0),1,8);
    //cvLine(img, cvPoint(x-radius/2, y+radius/2), cvPoint(x+radius/2, y-radius/2),CV_RGB(250,0,0),1,8);
}

void fndc(IplImage* result,int lbx,int lby,int rtx,int rty,bool isput){
  int Xc = 0;
   int Yc = 0;
   int counter = 0; // счётчик числа белых пикселей
   #ifndef qmode
   printf("\nroi=(%d;%d)x(%d;%d)\n",lbx,lby,rtx,rty);
   #endif
   cvLine(gray1, cvPoint(lbx, lby), cvPoint(rtx, lby),CV_RGB(0,0,0),1,8);
   cvLine(gray1, cvPoint(lbx, lby), cvPoint(lbx, rty),CV_RGB(0,0,0),1,8);
   cvLine(gray1, cvPoint(rtx, lby), cvPoint(rtx, rty),CV_RGB(0,0,0),1,8);
   cvLine(gray1, cvPoint(lbx, rty), cvPoint(rtx,rty),CV_RGB(0,0,0),1,8);
```

```
// пробегаемся по пикселям изображения
for(int y=rty; y<result->height&&y<=lby; y++)
{
   uchar* ptr = (uchar*) (result->imageData + y * result->widthStep);
   for(int x=lbx; x<result->width&&x<=rtx; x++)
   {
      if( ptr[x]>0 &&(x>bpx||y<bpy))
      {
         Xc += x;
         Yc += y;
         counter++;
      }
   }
}
if(counter!=0)
{
   float centerx = float(Xc)/counter;
   float centery = float(Yc)/counter;
   #ifdef sh_coords
```

```
        printf("%f x %f (!)\n", centerx, centery);
        #endif
        if(centerx>bpx||centery<bpy){
            cvCircle(gray1,cvPoint(centerx, centery),10,CV_RGB(0,255,0),1,8);
            if(isput){
                crd tmpcrd; tmpcrd.x=centerx; tmpcrd.y=centery;
                crds.push_back(tmpcrd);
            }
        }
        //printf("\n1tmpcrd.x=%f,tmpcrd.y=%f",tmpcrd.x,tmpcrd.y);
    }
    else{
        #ifndef qmode
        printf("counter==0\n");
        #endif
    }
}

// обработчик событий от мышки
void myMouseCallback( int event, int x, int y, int flags, void* param )
```

```
{
    IplImage* img = (IplImage*) param;

    switch( event ){
        case CV_EVENT_MOUSEMOVE:
            break;

        case CV_EVENT_LBUTTONDOWN:
                //#ifndef qmode
            printf("\n%d x %d\n", x, y);
            //#endif
            drawTarget(img, x, y, 50);
            fndc(img,x-20,y+20,x+20,y-20,0);
            break;
        case CV_EVENT_LBUTTONUP:
           if(ispmp>=1){
                if(ispmp==1){
                    printf("\npumpx=%d pumpy=%d",x,y);
                    pumpx=x; pumpy=y; ispmp++;
                }
```

```
                else if(ispmp==2){
                        printf("\nbpx=%d",x);
                        bpx=x; ispmp++;
                }
                else if(ispmp==3){
                        printf("\nbpy=%d",y);
                        bpy=y; ispmp=0;
                }
                /*else if(ispmp==4){
                        ispmp=0;
                }*/
        }
        break;
        case CV_EVENT_RBUTTONUP:
        printf("\npmp_mode\n");
        ispmp=1;
        break;
    }
}
```

```
unsigned int fndall(IplImage* result, int gridw=40, int gridh=40){//R; TODO;
  int hor=result->width/gridw;
  int ver=result->height/gridh;
  for(usi i=0;i<hor;i++){
      for(usi j=0;j<ver;j++){
          fndc(result,i*gridw,(j+1)*gridh,(i+1)*gridw,j*gridh,1);
          /*if(tmpcrd->x!=0||tmpcrd->y!=0){
              printf("\ntmpcrd->x=%f,tmpcrd->y=%f",tmpcrd->x,tmpcrd->y);
              crds.push_back(*tmpcrd);
              //delete tmpcrd;
          }*/
      }
  }
  //printf("\nqqq\n"); cvWaitKey(0);
  /*if(hor*gridw<result->width||ver*gridh<result->height){
      fndc(result,result->width-gridw,result->height,result->width,result->height-gridh);
      printf("\neoi; ni\n"); cvWaitKey(0);
  }*/
  bool isclsr=0; unsigned int rds=std::min(gridw,gridh),itr=0;
```

```
  while(isclsr==0){
        for(itr=0;itr<crds.size()&&isclsr==0;itr++){
              isclsr=crds.at(itr).inR(rds);
        }
        if(isclsr==0){
              if(rds<std::min(result->width-pumpx,result->height-(result-
>height-pumpy)));
                    rds+=std::min(gridw,gridh)/dvdr;
        }
        else{
              std::iter_swap(crds.begin(),crds.begin()+itr-1);
              std::cout<<"\nfnd="<<itr-1<<" "<<crds.at(0);
        }
   }
 }

 void chnggw(int pos){
   ngridw=pos;
   //fndall(dst,ngridw,ngridh);
 }
```

```
void chnggh(int pos){
  ngridh=pos;
  //fndall(dst,ngridw,ngridh);
}

void putpmp(IplImage* result){
  if(pumpx>bpx) bpx=pumpx;
  if(pumpy<bpy) bpy=pumpy;//less y -- more high;
  cvCircle(result,cvPoint(pumpx,pumpy),10,CV_RGB(255,255,0),1,8);
  cvLine(result, cvPoint(bpx, result->height), cvPoint(bpx, bpy),CV_RGB(255,0,0),1,8);
  cvLine(result, cvPoint(0, bpy), cvPoint(bpx, bpy),CV_RGB(255,0,0),1,8);
}

/*void fndpmp(IplImage* result){
  printf("\npump:\n");

  float xc=0,yc=result->height-1; unsigned char coh=1;
  for(int y=result->height-2; y>0; y--){
      uchar* ptr = (uchar*) (result->imageData + y * result->widthStep);
      for(int x=0; x<result->width; x++){
```

```
            if( ptr[x]>0 &&coh>252&&x>=xc){
                xc=x; yc=y; coh++;
            }
            else coh--;
        }
    }
    cvCircle(gray1,cvPoint(xc,yc),100,CV_RGB(0,0,0),1,8);
    cvShowImage("gray1",gray1);
    printf("\npmp: %f x %f\n",xc,yc);
    printf("\npmp\n");
    printf("\npmp1\n");
}*/

int main(int argc, char* argv[])
{
            float kernel[9];
        kernel[0]=-0.1; kernel[1]=0.2; kernel[2]=-0.1;

        kernel[3]=0.2; kernel[4]=2; kernel[5]=0.2;
```

```
kernel[6]=-0.1; kernel[7]=0.2; kernel[8]=-0.1;
CvMat kernel_matrix=cvMat(3,3,CV_32FC1,kernel);
// имя картинки задаётся первым параметром
const char* filename = argc == 2 ? argv[1] : "q.jpg";
// получаем картинку
image = cvLoadImage(filename,1);

printf("[i] image: %s\n", filename);
assert( image != 0 );

// создаём одноканальные картинки
gray = cvCreateImage( cvGetSize(image), IPL_DEPTH_8U, 1 );
//gray1 = cvCreateImage( cvGetSize(image), IPL_DEPTH_8U, 1 );
dst = cvCreateImage( cvGetSize(image), IPL_DEPTH_8U, 1 );

// окно для отображения картинки
cvNamedWindow("original",CV_WINDOW_NORMAL);
cvNamedWindow("gray",CV_WINDOW_NORMAL);
cvNamedWindow("gray1",CV_WINDOW_NORMAL);
cvNamedWindow("cvCanny",CV_WINDOW_NORMAL);
```

```
// преобразуем в градации серого
cvCvtColor(image, gray, CV_RGB2GRAY);
gray1=cvLoadImage(filename,1);//cvCvtColor(image, gray1, CV_RGB2GRAY);
// показываем картинки
cvShowImage("original",image);
cvShowImage("gray1",gray1);


cvFilter2D(gray, gray, &kernel_matrix, cvPoint(-1,-1));
cvShowImage("gray",gray);
/*cvWaitKey(0);
kernel[0]=-0.1; kernel[1]=-0.1; kernel[2]=-0.1;
kernel[3]=-0.1; kernel[4]=2; kernel[5]=-0.1;
kernel[6]=-0.1;kernel[7]=-0.1;kernel[8]=-0.1;
kernel_matrix=cvMat(3,3,CV_32FC1,kernel);
cvFilter2D(gray, gray, &kernel_matrix, cvPoint(-1,-1));
cvShowImage("gray",gray);*/
//cvWaitKey(0);
putpmp(gray1);
cvShowImage("gray1",gray1);
//fndpmp(dst);
```

```
// получаем границы
cvCanny(gray, dst, threshold1, threshold2, Hmin);
cvCreateTrackbar("aperture_size", "original", &Hmin, 7, myTrackbarHmin);
cvCreateTrackbar("threshold1", "original", &threshold1, 256, threshold1f);
cvCreateTrackbar("threshold2", "original", &threshold2, 256, threshold2f);
cvCreateTrackbar("gridw", "original", &ngridw, dst->width, chnggw);
cvCreateTrackbar("gridh", "original", &ngridh, dst->height, chnggh);
cvSetMouseCallback( "cvCanny", myMouseCallback, (void*) dst);

    while(1){
    cvShowImage("cvCanny", dst );
    gray1=cvLoadImage(filename,1);//cvCvtColor(image, gray1, CV_RGB2GRAY);
    // ждём нажатия клавиши
    fndall(dst,ngridw,ngridh);
    for(unsigned int i=0;i<crds.size();i++){
        std::cout<<"\ni="<<i<<crds.at(i);
    }
    putpmp(gray1);
```

```
    cvShowImage("gray1",gray1);
    crds.clear();
    #ifdef sh_coords
    char c = cvWaitKey(0);
    #else
    char c = cvWaitKey(50);
    #endif
    if (c == 27) { // если нажата ESC - выходим
          break;
    }
    }


// освобождаем ресурсы
cvReleaseImage(&image);
cvReleaseImage(&gray);
cvReleaseImage(&gray1);
cvReleaseImage(&dst);
// удаляем окна
cvDestroyAllWindows();
```

```
    return 0;
}
```

ДОДАТОК В

Код програми

Файл mBP.ino

```
#include "my_bp.h"
#include "my_rot.h"

#define pumpp 30

my_bp mbpz(7/*en*/,8/*dirp*/,
      9/*stpp*/,10/*dlyv*/,
     10/*cntrlp*/,3/*Ocd*/);
my_bp mbpy(3/*en*/,4/*dirp*/,
      5/*stpp*/,1/*dlyv*/,
     6/*cntrlp*/,2/*Ocd*/);
my_bp mbpx(22/*en*/,23/*dirp*/,
      24/*stpp*/,1/*dlyv*/,
     25/*cntrlp*/,1/*Ocd*/);

my_rot mrot(38,39,40,41,50);
```

```
void setup() {
  pinMode(pumpp,OUTPUT);
  mbpz.initb();
  mbpy.initb();
  mbpx.initb();
  //mbp.mv(100,1);
  mbpz.mv(15,0);
  mbpy.mv(200,0);
  //mbpz.mv(90,1);
  /*mbpy.mv(50,0);
  delay(100);
  //mbpx.mv(mbpx.cntm-50,0);
  mbpx.mv(50,0);
  delay(100);
  digitalWrite(pumpp,HIGH);*/

  /*delay(1000);
  mbpz.mv(200,0);*/
  /*delay(100);
  digitalWrite(pumpp,LOW);
```

```
  delay(1000);*/
  /*mrot.mv(50,0);
  mrot.mv(50,1);*/
}

void loop() {
  //mbp.incntz=1400;
/*  mbpz.mv(mbpz.cntm-50,0);
  delay(100);
  mbpy.mv(mbpy.cntm-50,0);
  delay(100);
  mbpx.mv(mbpx.cntm-50,0);
  delay(100);*/

}
```

Файл my_bp.h

```
#ifndef MY_BP_H
#define MY_BP_H

#define Ox 1
```

```
#define Oy 2
#define Oz 3


class my_bp{
  public:
    my_bp(unsigned char en,unsigned char dirp,
      unsigned char stpp,unsigned char dlyv,
      unsigned char cntrlp,unsigned char Ocd);
    unsigned char enp,dir,stp,dly,cntrl;
    unsigned char O;//Ox -- 1; Oy -- 2; Oz -- 3;
    unsigned int incnt,cntm=0,cntzm=1400,cntym=150000,cntxm=23500;
    void initb();
    void mv(unsigned int num,bool to);
};


#endif
```

Файл my_bp.cpp

```
#include "my_bp.h"
#include <Arduino.h>
my_bp::my_bp(unsigned char en,unsigned char dirp,
```

```
      unsigned char stpp,unsigned char dlyv,
      unsigned char cntrlp,unsigned char Ocd){
  enp=en;dir=dirp;stp=stpp;dly=dlyv;cntrl=cntrlp;O=Ocd;
  pinMode(dir,OUTPUT);pinMode(stp,OUTPUT);
  pinMode(cntrl,INPUT);pinMode(enp,INPUT);
  switch(O){
    case Ox: cntm=cntxm;
    break;
    case Oy: cntm=cntym;
    break;
    case Oz: cntm=cntzm;
    break;
  }
}

void my_bp::initb(){
  pinMode(enp,OUTPUT); digitalWrite(enp,LOW);//enable move;
  if(digitalRead(cntrl)!=LOW){//to the cntrl;
     if(O==Oz){
      digitalWrite(dir,LOW);//for Oz;
```

```
  }
 if(O==Oy||O==Ox){
   digitalWrite(dir,HIGH);//for Oy;
 }
}
bool cfl=1,cfl1=0; unsigned long cmll=millis();
while(digitalRead(cntrl)!=LOW){//move;
 //for(unsigned int i=0;i<num;i++){
 /*if(digitalRead(cntrl)==LOW){//fin;
   pinMode(enp,INPUT);//off move;
   incnt=0; break;
 }*/
 if(cfl==1){
   digitalWrite(stp, HIGH); cfl=0;
   cmll=millis();
 }
 if(millis()-cmll>=dly){//delay(dly);
   cfl1=1; cmll=millis();
 }
 if(cfl1==1){
```

```
      digitalWrite(stp, LOW); cfl1=0;
      cmll=millis();
    }
    if(millis()-cmll>=dly){//delay(dly);
      cfl=1; cmll=millis();
    }
  }
  incnt=0; pinMode(enp,INPUT);//off move;
}

void my_bp::mv(unsigned int num,bool to){
  pinMode(enp,OUTPUT); digitalWrite(enp,LOW);//enable move;
  if(to==1&&incnt>0){//to the cntrl;
    if(O==Oz){
      digitalWrite(dir,LOW);//for Oz;
    }
    if(O==Oy||O==Ox){
      digitalWrite(dir,HIGH);//for Oy;
    }
  }
```

```
else if(to==0&&incnt<cntm){//from cntrl;
  if(O==Oz){
    digitalWrite(dir,HIGH);//for Oz;
  }
  if(O==Oy||O==Ox){
    digitalWrite(dir,LOW);//for Oy;
  }
}
unsigned int i=0;bool cfl=1,cfl1=0; unsigned long cmll=millis();
while(i<num){//move;
//for(unsigned int i=0;i<num;i++){
  if(digitalRead(cntrl)==LOW&&to==1){//fin;
    pinMode(enp,INPUT);//off move;
    i=num; incnt=0; break;
  }
  if(to==0&&incnt>=cntm-50){
    pinMode(enp,INPUT);//off move;
    i=num; break;
  }
  if(cfl==1){
```

```
    digitalWrite(stp, HIGH); cfl=0;
    cmll=millis();
   }
   if(millis()-cmll>=dly){//delay(dly);
    cfl1=1; cmll=millis();
   }
   if(cfl1==1){
    digitalWrite(stp, LOW); cfl1=0; i++;
    if(to==1) incnt--;
    else incnt++;
    cmll=millis();
   }
   if(millis()-cmll>=dly){//delay(dly);
    cfl=1; cmll=millis();
   }
  }
  pinMode(enp,INPUT);//off move;
  //else
}
```

Файл my_rot.h

```
#ifndef MY_ROT_H
#define MY_ROT_H


class my_rot{
  public:
    my_rot(unsigned char _in1,unsigned char _in2,
      unsigned char _in3, unsigned char _in4,unsigned int _dly);
    unsigned char in1,in2,in3,in4,curpos;
    unsigned int n = 4, steps = 8,dly;
    /*unsigned int mass[4*8]={
          1,0,0,0,
          0,1,0,0,
          0,0,1,0,
          0,0,0,1,
          1,0,0,0,
          0,1,0,0,
          0,0,1,0,
          0,0,0,1};*/
    unsigned int mass[4*8]={
          1,0,0,0,
```

```
        1,1,0,0,
        0,1,0,0,
        0,1,1,0,
        0,0,1,0,
        0,0,1,1,
        0,0,0,1,
        1,0,0,1};
   void mv(unsigned int num,bool to);
};

#endif
```

Файл my_rot.cpp

```
#include "my_rot.h"
#include <Arduino.h>

my_rot::my_rot(unsigned char _in1,unsigned char _in2,
    unsigned char _in3, unsigned char _in4,unsigned int _dly):
    in1(_in1), in2(_in2), in3(_in3), in4(_in4), dly(_dly){
  pinMode(in1,OUTPUT);
  pinMode(in2,OUTPUT);
```

```
  pinMode(in3,OUTPUT);
  pinMode(in4,OUTPUT);
  curpos=255;
}

void my_rot::mv(unsigned int num,bool to){
  for (unsigned int j=0; j<num; j++){
    if(to==0){//counterclockwise;
      curpos++; curpos= (curpos==steps)? 0:curpos;
      //for(unsigned char i=0;i<steps;i++){
        digitalWrite(in1,mass[/*i*/curpos*n+0]);
        digitalWrite(in2,mass[/*i*/curpos*n+1]);
        digitalWrite(in3,mass[/*i*/curpos*n+2]);
        digitalWrite(in4,mass[/*i*/curpos*n+3]);
        delay(dly);
      //}
    }
    else{//clockwise;
      curpos--; curpos= ((255-curpos)<2)? steps-1:curpos;
      //for(unsigned char i=steps-1;i<255;i--){
```

```
        digitalWrite(in1,mass[/*i*/curpos*n+0]);
        digitalWrite(in2,mass[/*i*/curpos*n+1]);
        digitalWrite(in3,mass[/*i*/curpos*n+2]);
        digitalWrite(in4,mass[/*i*/curpos*n+3]);
        delay(dly);
      //}
    }
  }
}
```