

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Інститут інформатики та радіоелектроніки
Факультет комп'ютерних наук та технологій
(повне найменування інституту, факультету)

Кафедра комп'ютерних систем та мереж
(повне найменування кафедри)

Пояснювальна записка

до дипломного проекту (роботи)

бакалавра

(ступінь вищої освіти (освітній ступінь))

на тему РОЗРОБКА МІКРОПРОЦЕСОРНОЇ СИСТЕМИ УПРАВЛІННЯ
КРОКОВИМ ДВИГУНОМ

Виконав: студент 4 курсу, групи КНТ-518сп
спеціальності _____

123 «Комп'ютерна інженерія»

(код і найменування спеціальності)

Освітня програма (спеціалізація) _____

«Комп'ютерна інженерія»

Тітюєв Олександр Саїдович

(прізвище та ініціали)

Керівник Грушко С.С.

(прізвище та ініціали)

Рецензент Степаненко О.О.

(прізвище та ініціали)

м. Запоріжжя
2021 рік

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
(повне найменування вищого навчального закладу)

Інститут, факультет інформатики та радіоелектроніки, комп'ютерних наук і технологій

Кафедра «Комп'ютерні системи та мережі»

Ступінь вищої освіти (освітній ступінь) бакалаврський

Спеціальність 123 Комп'ютерна інженерія
(код і найменування)

Освітня програма (спеціалізація) Комп'ютерна інженерія
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри Кудерметов Р.К.

“ ” 2021 року

ЗАВДАННЯ
НА ДИПЛОМНИЙ ПРОЕКТ (РОБОТУ) СТУДЕНТА

Тітюєва Олександра Саїдовича

(прізвище, ім'я, по батькові)

1. Тема проекту (роботи) Розробка мікропроцесорної системи управління кроковим двигуном

керівник проекту (роботи) Грушко Світлана Сергіївна, к. т. н., доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу від “17” березня 2021 року №81

2. Строк подання студентом проекту (роботи) 1 травня 2021 року

3. Вихідні дані до проекту (роботи) пристрій який генерує керуючі імпульси до трьох каналів на основі G-коду з максимальною частотою до 15KHz

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1) Аналіз технічного завдання, дослідження існуючих систем чисельної обробки даних які мають у собі реалізацію управління кроковим двигуном;

2) Розробка структури та алгоритму роботи системи;

3) Реалізація та дослідження системи.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

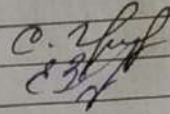
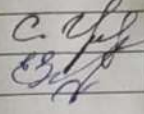
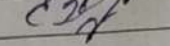
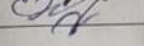
Структурна схема системи;

Електрична схема системи;

Алгоритм програми мікроконтролера;

Результат тестування електричної схеми у програмному засобі Proteus.

6. Консультанти розділів проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4	Грушко С.С., к.т.н., доцент		
нормоконтроль	Зелік О.В., асистент		

7. Дата видачі завдання 02.04.2021 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
1	Аналіз існуючих систем керування кроковими двигунами, перегляд отриманих результатів й виділення їх недоліків та переваг	01.03.2021 р.	
2	Вибір стеку технологій для побудови системи	15.03.2021 р.	
3	Розробка структурної схеми системи	20.03.2021 р.	
4	Розробка принципової електричної схеми системи	01.04.2021 р.	
5	Налаштування мікрокомп'ютеру Raspberry Pi	10.04.2021 р.	
6	Розробка алгоритму програми для мікроконтролера	01.05.2021 р.	
7	Тестування електричної схеми у програмному засобі Proteus	10.05.2021 р.	
		14.05.2021 р.	
8	Оформлення допоміжного матеріалу	17.05.2021 р.	
9	Оформлення графічного матеріалу	20.05.2021 р.	

Студент

О.С. Тітюєв

(ініціали та прізвище)

Керівник проекту (роботи)

С.С. Грушко

(ініціали та прізвище)

РЕФЕРАТ

ПЗ: 72 с. 36 рис., 1 табл., 2 додатки, 14 джерел.

АВТОМАТИЗАЦІЯ, МІКРОКОНТРОЛЕР, ЧПУ, CNC, DPS-КОНТРОЛЕР

Об'єкт розробки – Система управління кроковим двигуном.

Мета роботи – розробити систему керування кроковими двигунами.

В дипломній роботі було розглянуто і проаналізовано принцип управління кроковим двигуном а саме - системи чисельної обробки даних(Data Processing Systems) DSP-контролери.

Розроблено структурну схему системи управління крокового двигуна, проведено вибір стеку технологій програмного й апаратного забезпечення.

Проведено тестування роботи системи у програмному продукті Proteus.

ЗМІСТ

Вступ.....	7
1 Аналітичний огляд стану питання та конкретизація задачі роботи	9
1.1 Загальні відомості о чисельних системах керування станку	9
1.2 Історія розвитку.....	13
1.3 Аналіз існуючих систем та підходів що до розробки чпу	14
1.3.1 Огляд існуючих систем чпу	14
1.3.1.1 Контролер mesa 7i85	14
1.3.1.2 Контролер rmhv3.1	16
1.3.1.3 DSP контролер richauto a11	17
1.3.2 Розробка власної системи чпу.....	20
1.3.3 Комбінування готових рішень з власними розробками	20
1.3.4 Класифікація архітектурних рішень систем чпу.....	20
1.4 Постановка завдання.....	21
2 Теоретична частина.....	21
2.1 Основні функції системи чисельного управління	21
2.2 Порівняння технологій	24
2.2.1 Технології передачі даних.....	24
2.2.2 Апаратні платформи ядра.....	27
2.2.3 Апаратні платформи для розв'язки та передачі даних.....	32
2.3 Особливості архітектури і розробки	42
2.3.1 Архітектура	43
2.3.2 Частина блоку ядра або серверна частина чпу.....	43
2.3.2 Частина блоку розв'язки.....	43
3 Практична частина	44
3.1 Вибір стеку технологій	44
3.1 Налаштування raspberry pi.....	45
3.1.1 Встановлення raspberry os	45
3.1.2 Налаштування віддаленої середи розробки за допомогою ssh.....	46
3.1.3 Налаштування програмного забезпечення руспс.....	48

3.1.4 Встановлення руспс	49
3.2 Розробка структурної схеми пристрою.....	53
3.3 Розробка апаратної частини пристрою	55
3.4 Програмування мікроконтролера avr atmega32	57
3.5 Моделювання роботи пристрою	59
3.5.1 Тестування блоку розв'язки на базі atmega32	60
3.5.2 Розробка віртуальної моделі станку.....	62
3.5.3 Створення програми емуляції станку засобами unity	64
3.5.3.1 Налаштування плати arduino.....	65
3.5.3.2 Налаштування unity.....	67
3.5.4 Розробка програми емуляції станку за допомогою unity	68
3.5.5 Тестування руспс	69
Висновки	71
Перелік джерел посилань	73
Додаток А	75
Додаток В	79
Перелік графічного матеріалу:	
Плакат 1 – Структурна схема пристрою	
Плакат 2 – Електрична схема пристрою	
Плакат 3 – Блок схема основного алгоритму програми мікроконтролера	
Плакат 4 – Результат тестування пристрою	

ВСТУП

Наразі нинішнє становище позначено масовим наповненням ринку всілякою автоматизованою апаратурою найрізноманітнішого призначення та складності, від пластикової платіжної картки до смарт кросівок, автомобіля і найскладніших установок. Причина цьому явищу є стрімкий розвиток та автоматизоване виробництво інтегральних мікросхем одними із яких є мікроконтролери та компактні повноцінні мікрокомп'ютери розміром з платіжну картку, сьогодні мікросхемотехніка скрізь, в усіх сферах життєдіяльності людини, все більше насиченість електроніки в нашому оточенні зростає з року в рік.

Ще зовсім недавно розробнику електронних схем доводилося мати справу з безліччю електронних компонентів, розміщуючи їх з величезними труднощами на печатній платі. Найпоширенішим способом створення макета (прототипу) майбутнього пристрою був “живий”: підібравши електронні компоненти, розробник брав у руки паяльник і збирав на макетних платах окремі вузли або пристрій у цілому. Потім починався процес налагодження: виправлення помилок принципової схеми, встановлення режимів роботи, уточнення параметрів застосовуваних компонентів і т.д.

На сьогодні ми маємо безліч різноманітних систем автоматизованого проектування та засобів розробки, які дозволяють значно спростити роботу розробника електронної апаратури, зменшивши при цьому матеріальні витрати та час виходу готової продукції й ризик при розробці. Застосування систем автоматизованого проектування та інших засобів дозволяє швидко отримувати зручні і досить надійні розробки.

Найбільш численна група засобів розробки розрахована на роботу з мікроконтролерами без яких зараз не обходиться створення нових пристроїв. Застосування мікроконтролерів, мікропроцесорів у зв'язці з програмованими інтегральними схемами, при розробці радіоелектронної апаратури, або

обчислювальних пристроїв та пристроїв автоматики, значно дозволяє спростити схемотехніку проекту; уніфікувати елементну базу - один і той же контролер може застосовуватися для створення безлічі різних пристроїв; уніфікація, в свою чергу призводить до зниження вартості виробництва; підвищити гнучкість розробки.

1 АНАЛІТИЧНИЙ ОГЛЯД СТАНУ ПИТАННЯ ТА КОНКРЕТИЗАЦІЯ ЗАДАЧІ РОБОТИ

У цьому розділі проводиться загальний огляд поняття фрезерувальних машин з чисельними системами керування, історія розвитку подібних систем, та аналіз існуючих систем та концепцій, призначених для вирішення поставлених в роботі завдань, наводиться висновок щодо переваг і недоліків цих рішень, формуються вимоги що до розроблюваної апаратної системи чисельного керування, проводиться вибір оптимального підходу, що дозволяє в повній мірі описати необхідний функціонал для того, щоб мінімізувати витрати як фінансів, так і часу.

1.1 Загальні відомості о чисельних системах керування станку

Під концепцією чисельної системи керування прийнято розуміти сукупність впливів на механізми станку, що забезпечують виконання технологічного циклу обробки під системою управління – пристрій або сукупність пристроїв, що реалізують ці впливи.

Числове програмне управління (ЧПУ) - це управління, при якому програму задають у вигляді записаного на будь-якому носії масив інформації. Керуюча інформація для систем ЧПУ є дискретною а її обробка в процесі управління здійснюється цифровими методами. Управління технологічними циклами практично повсюдно здійснюється за допомогою програмованих логічних контролерів, що реалізуються на основі принципів цифрових електронних

обчислювальних пристроїв. У табл. 1.1 перераховані цілі і функції сучасного багаторівневого пристрою ЧПУ.

Таблиця 1.1 - Цілі та функції багаторівневих пристроїв ЧПУ

Рівні	Ціль	Функції
I	Забезпечення обробки заготовки на станку	Управління циклом роботи станку. Контроль переміщення робочих органів станку
II	Отримання інформації про функціонування станку	Прийняття інженером-технологом або оператором рішення про правильність роботи верстата і його несправності на підставі інформації про роботу станку
III	Отримання вихідних даних для організації виробництва в цеху	Прийняття інженером-технологом або оператором рішення на підставі інформації про роботу станку кваліфікованих рішень що до керування виробництвом

За технологічним призначенням і функціональними можливостями системи ЧПУ поділяють на чотири групи:

Позиційні, в яких задають тільки координати кінцевих точок положення виконавчих органів після виконання ними певних елементів робочого циклу;

Контурні або безперервні, що керують рухом виконавчого органу по заданій криволінійній траєкторії;

Універсальні (комбіновані), в яких здійснюється програмування як переміщень при позиціонуванні, так і руху виконавчих органів по траєкторії, а також зміни інструментів і завантаження-вивантаження заготовок;

Багатоконтурні системи, що забезпечують одночасне або послідовне управління функціонуванням ряду вузлів і механізмів станку.

Прикладом застосування систем ЧПУ першої групи є свердлильні, розточувальні і координатно-розточувальні верстати. Прикладом другої групи

служать системи ЧПУ різних токарних, фрезерних і круглошліфувальних верстатів. До третьої групи відносяться системи ЧПУ різних багатоцільових токарних і свердлильно-фрезерно-розточувальних станків.

До четвертої групи належать без центрові круглошліфувальні верстати, в яких від систем ЧПУ керують різними механізмами: правки, подачі бабок і т.п. Існують позиційні, контурні, комбіновані і багатоконтурні (рис. 1.0.) цикли управління.

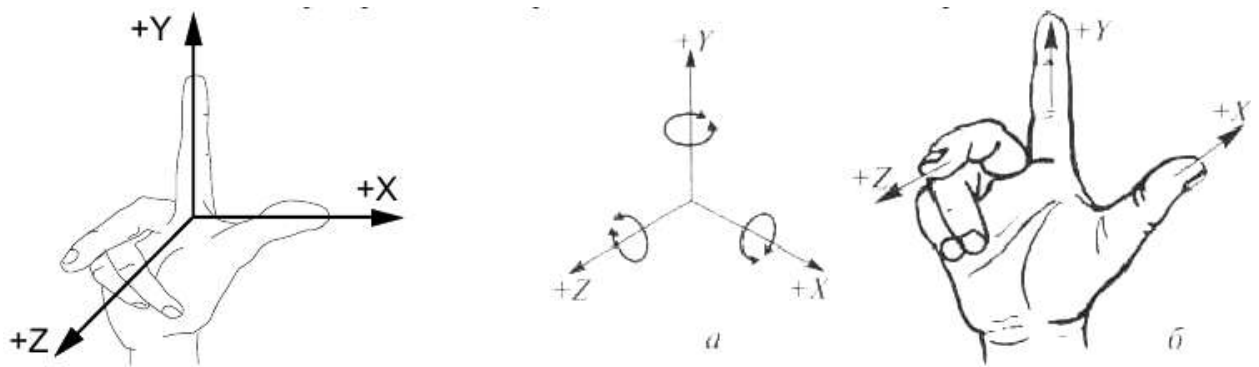


Рисунок 1.1 – розташування осей координат у станках ЧПУ (а);
правостороння система координат(б)

ЧПУ забезпечує управління рухами робочих органів верстата і швидкістю їх переміщення при формоутворенні, а також послідовністю циклу обробки, режимами різання, різними допоміжними функціями.

Система числового програмного керування (СЧПУ) - це сукупність спеціалізованих пристроїв, методів і засобів, необхідних для здійснення ЧПУ станками. Від рівня використання обчислювальної техніки системи ЧПУ класифікують на наступні типи:

Системи типу NC(Numeral Control) – числове програмне управління яке здійснює адресування команд, інтерполяцію проміжних точок по опорним та реалізацію типових циклів по жорстко заданим алгоритмам, реалізованим

апаратним способом. інформація в систему ЧПУ типу NC вводиться з керуючої програми кадрами (Порціями).

Системи типу MNC (Memory NC) або SNC (Stored NC) -агрегатно-блокові системи ЧПУ типу NC, оснащені додатковим блоком оперативної пам'яті, що дозволяє зберігати інформацію про керуючу програму. Програма в пристрій ЧПУ вводиться відразу, перевіряється, а потім видається для обробки кадрами. Перевагою системи типу MNC, в порівнянні з системою типу NC, є висока надійність в роботі, тому необхідність у використанні складного фотозчитувального пристрою для кожного кадру програми не потрібно.

Системи типу HNC (Hand NC) - з ручним завданням керуючої програми на пульті управління. Перевага таких систем в порівнянні з системами типу MNC – відсутність необхідності підготовки КП (керуючої програми) технологом-програмістом.

Системи типу CNC (Computer NC) - системи управління зі вбудованими однією або декількома мікропроцесорами і з програмною реалізацією алгоритмів, які записуються в постійний запам'ятовуючий пристрій при виготовленні пристрою ЧПУ. Системи типу CNC мають можливість формувати типові цикли обробки стосовно до різних технологічних завдань. Програмно-математичне забезпечення для реалізації цієї можливості зберігається в постійно перепрограмованому пристрої зберігання даних. системи CNC дозволяють програмувати логіку роботи електроавтоматики силового обладнання станку.

Система DNC (Direct Numerical Control) - система, що управляє групою верстатів від однієї ЕОМ, що має загальну пам'ять для зберігання програм, які розподіляються за запитом від станків. Такі пристрої ЧПУ є вищого рангу і служать для організації узгодженої роботи технологічних об'єктів, включених в комплекс, наприклад в ГПС (гнучкі виробничі системи).

Система PCNC (Personal Computer NC) - системи управління, що з'явилися останнім часом і побудовані на основі персонального комп'ютера в індустріальному виконанні, основне відмінність яких полягає в захищеному від ударів виконанні, а також в наявності спеціальної схемотехнічної плати, що

забезпечує сполучення ЕОМ з приводами та датчиками. Така побудова дозволяє здешевити систему ЧПУ, легко її адаптувати до різних по функціональним призначенням станків шляхом корекції відповідного програмного забезпечення. Все це дозволяє легко модернізувати застарілі системи ЧПУ NC, MNC, SNC, HNC, CNC, DNC до PCNC, що в ряді випадків успішно виконується.

Данна інформація була узята з джерела autoWelding, у статті про загальні відомості систем ЧПУ.

1.2 Історія розвитку

Першим прийнятим технологічним досягненням у сфері верстатів ЧПУ був присвячений Джеймсу Парсонсу у 1949 році. Вирахувавши координати профілю крила гвинтокрила він перевів ці данні на перфокарту, яку й використав на швейцарському розточеному станку, ця інформація сприяла до виробництва багатьох лопатей для гвинтокрилів і обшивок літаків. Відповідно до загальноприйнятої історії ЧПУ, це вважалося першим станком з ЧПУ. Пізніше Парсон отримав Меморіальну нагороду Джозефа Марії Жаккарда за свою роботу. Цей еволюційний випадок послуговував ще одним вектором розвитку верстатів. Нижче представлений графік еволюції, які були в історії обробки з ЧПУ.

1952-1958: Коли холодна війна стала інтенсивнішою, виникла необхідність в підвищенні ефективності та продуктивності при виробництві багатьох машин і зброї. Тому в 1952 році Річард Кегг разом з Массачусетським технологічним інститутом виготовив перший фрезерний верстат з ЧПУ, відомий як Cincinnati Milacron Hydrotel. Річард Кегг пізніше подав заявку на патент на пристрій з двигуном для позиціонування в 1958 році.

1967-1972: Обробка з ЧПУ ставала все більш визнаною у всьому світі. Це сталося завдяки розвитку систем автоматизованого проектування (CAD) і автоматизованої обробки (CAM) в 1972 році. Включення CAD і CAM в обробку з

ЧПУ призвело до масових розробок в області обробки з ЧПУ. Однак вони не розглядалися як стандартна частина виробничого процесу.

1976-1989: У 1976 році 3D-проективання і комп'ютерна обробка були включені в обробку з ЧПУ. У 1989 році верстати з програмним управлінням CAD і CAM стали промисловим стандартом для верстатів з ЧПУ.

1.3 Аналіз існуючих систем та підходів що до розробки ЧПУ

Пристрій чисельного керування є частиною загального виду структури комплексу «станок з ЧПУ», який здійснює видачу керуючих впливів за заданою програмою. Створення цього блоку можна вирішити декількома способами, кожен із яких має свої переваги так і недоліки. Розглянемо їх. Можна виділити такі підходи що до створення системи ЧПУ:

- Використання готових систем ЧПУ;
- Розробка власної системи ЧПУ;
- Комбінування власних розробок з іншими готовими рішеннями;

1.3.1 Огляд існуючих систем ЧПУ

На сьогодні простим рішенням що до розробки системи ЧПУ є використання вже готової системи чисельної обробки даних а саме DSP-контролери. Проаналізувавши ринок, було знайдено типи, або виконання систем які відрізняються по виду конструкції і монтажу. У зв'язку з цим існує різниця в способі управління станком. Розглянемо найбільш відомі з цих представників.

1.3.1.1 Контролер MESA 7185

Представлений DSP-контролер являє собою блок інтерпретації даних між комп'ютером та пристроями, що реалізують ці впливи на робочі органи станку, якими може бути драйвери крокового двигуна та інвертор шпинделю. В своєму визначенні має змогу приймати данні від верстату, наприклад сигнал від енкодерів. Передача інформації комп'ютер – DSP-контролер відбувається завдяки інтерфейсу RS-422 який забезпечує односпрямовану диференціальну передачу даних по лініях з можливістю з'єднання «точка-точка» або «master - багато slave». Вибраний

інтерфейс RS422 і нині досить часто застосовується в промисловості, має схожість з RS-232, відміна у використанні диференційного сигналу для одномоментної відправки і приймання повідомлень по різних лініях. Зазвичай для організації RS-422 використовується роз'єм DB9 однак тут для підключення контролера використовується роз'єм DB25 задля реалізації ще одного інтерфейсу ПЛІС Anything I/O від Mesa - двоканальний інтерфейс кодувальника для інтерфейсних карт.

Як зазначено в описі пристрою 7I85 зазвичай підключає до чотирьох приводів з послідовним інтерфейсом, до чотирьох з п'яти повно дуплексних інтерфейсів RS-422 7I85 і до чотирьох кодувальників з індексом до входів кодувальника 7I85s. Входи енкодера можуть бути TTL або диференціальними для кожного входу. 7I85 також може подавати живлення 5V на енкомери.

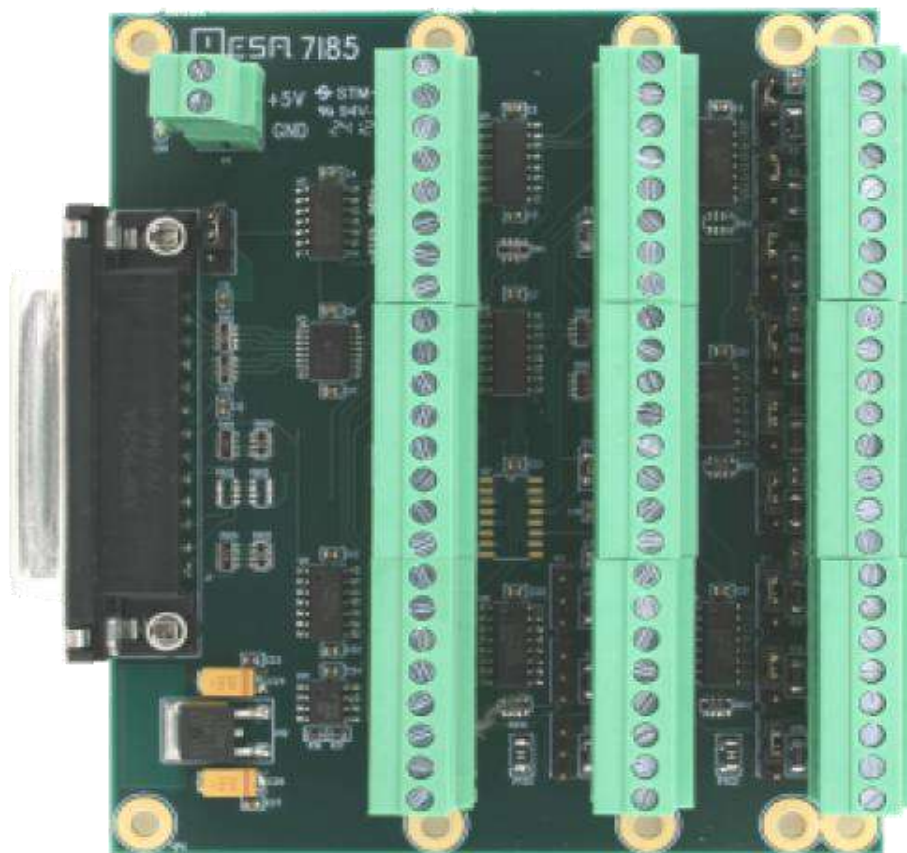


Рисунок 1.2 – контролер MESA 7I85 вигляд зверху

Головним недоліком є необхідність використовувати пристрій з персональним комп'ютером з програмним забезпеченням САМ і драйвери, також схема не передбачає міри запобігання короткому замиканню.

1.3.1.2 Контролер RMHV3.1

RMHV3.1 має 4 керованих осі. Контрольний період для кожної позиції становить всього 4 мілісекунди, з високою точністю управління. Величина вихідного імпульсу становить 500кГц, ширина імпульсу може регулюватися. Контролер підтримує як крокові двигуни так і серво.



Рисунок 1.3 – лицевий вид корпусу контролеру RMHV3.1

Даний пристрій використовує архітектуру ARM + матрицю FPGA. ARM що відповідає за користувацький інтерфейс та обробку даних – основне ядро. FPGA відповідає за базовий алгоритм генерації керуючих імпульсів.

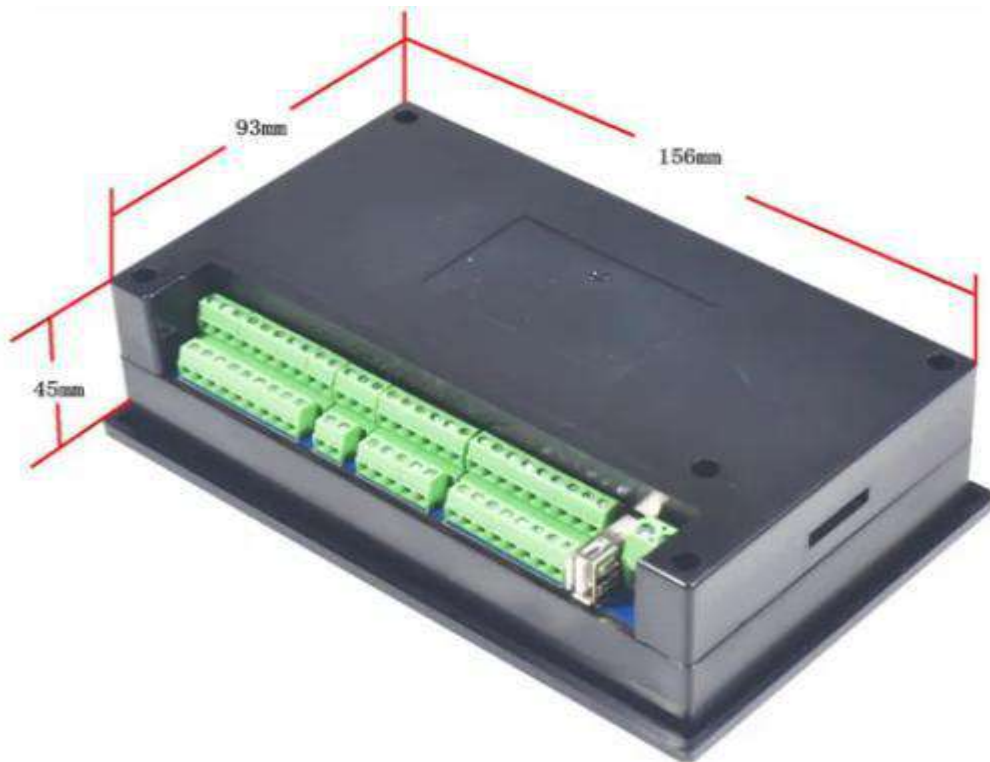


Рисунок 1.4 – вид задньої панелі корпусу контролеру RMHV3.1

Лицева панель корпусу пристрою має 4,3 дюймовий TFT дисплей і 17 фізичних кнопок. Як зазначено у мануалі до пристрою контролер підтримує стиль G – коду, який застосовується в обладнанні з ЧПУ FANUC.

Серед переваг відрізняється автономністю, має DCDC розв'язку, підтримує функцію збереження даних після відімкнення від джерела струму.

Серед недоліків слід зазначити неприємну вартість, неможливість налаштування більш гнучкої реакції поведінки від датчиків по причині закритого ПО, має лише один режим регулювання швидкості шпинделю VSO mode (0-10V) що може запобігти подальшому модернізуванню верстату

1.3.1.3 DSP контролер RichAuto A11

RichAuto A11 – DSP контролер також являється автономною системою керування для верстатів у вигляді компактного пульта з дисплеєм, який може свободно розташуватися в одній руці. Загалом представляє з себе програмно-технічний комплекс що складається з пульта, рисунок 1.5 та плати розв'язки, рисунок 1.6, котрий продається комплектом.



Рисунок 1.5 – RichAuto A11 пульт

Завдяки спеціальному 50 контактному кабелю пульт підключається до відповідного роз'єму плати розв'язки, рисунок 1.7.

DSP контролер має в своїй основі восьмиядерний процесор PARTAN-6, що дозволяє виробляти миттєвий автоматичний контроль фрезерування по трьох осях. Для управління система може виконувати стандартні G-коди і HPGL-3D коди, що в свою чергу дозволяє користуватися практично будь-яким програмним забезпеченням для створення керуючих програм CAM.

Основний недолік висока вартість продукту починаючи від 12 тисяч грн.



Рисунок 1.6 – RichAuto A11 плата розв'язки



Рисунок 1.7 – RichAuto A11

1.3.2 Розробка власної системи ЧПУ

Діаметрально протилежним підходом до створення системи ЧПУ це, не інакше як власна системи керування з повного «нуля». Такий підхід являє собою надзвичайно складну, довготривалу задачу, яка може в результаті призвести до вузько спеціалізованого програмного забезпечення, орієнтованого на певні компоненти системи, які складно додавати. Однією з найважливіших задач є розробка первинної архітектури, яка повинна передбачувати високий рівень розподілу та мінливості, адже помилки, допущені під час цього етапу буде важко виправити під час впровадження системи.

Головною перевагою такого підходу є найвищий рівень задоволення потреб споживача, можливість подальшого розвитку системи та варіативність використання готової системи з власними ідеями, що мають не стандартний характер. Ще однією важливою перевагою можна назвати низьку собівартість отриманої системи, якщо не враховувати ціну інтелектуальних та часових втрат.

1.3.3 Комбінування готових рішень з власними розробками

Суттю даного підходу, як видно із назви, полягає в максимальному делегуванні функціоналу до готових рішень, зосередившись на тих частинах, що потребують власної реалізації. Наприклад, розробка керуючої програми виконання G-коду з власними модулями-драйверами які реалізують робочі імпульси або інтерфейс взаємодії.

1.3.4 Класифікація архітектурних рішень систем ЧПУ

Провівши аналіз існуючих систем, можна виділити декілька концептуально відмінних способів комунікації модулів між собою та з ядром інформаційної системи:

- централізована система;
- децентралізована система;
- змішана система.

1.4 Постановка завдання

Під час предметного аналізу було встановлено, що існуючі системи ЧПУ(не індустріального характеру), хоч і дозволяють себе додати до багатоцільових станків але мають в собі досить специфічний характер та прив'язані до певного кола пристроїв які вдало розташувалися на ринку.

Ядро системи чисельної обробки даних повинно виступати у ролі уніфікованого посередника між апаратними складовими та програмовим забезпеченням, керувати приладами відповідно до загальноприйнятих інструкцій CNC (G-коду, HPGL-3D - коду).

Таким чином, необхідно вирішити наступні задачі:

- Побудувати інформаційну систему, яка зможе підтримувати обробку одного із популярних керуючих кодів – G-коду;
- Розробити функціональну можливість для змоги редагування керуючої програми за допомогою засобів техніки що входять у составі блоку ЧПУ;
- Розробити функціонал притаманний багатоконтурним системам.

2 ТЕОРЕТИЧНА ЧАСТИНА

2.1 Основні функції системи чисельного управління

Слід зазначити, що складність та кількість складових елементів обладнання з ЧПУ може бути дуже варіативне, відповідаючи як загальноприйнятим, так і специфічним потребам.

Основним характерним функціоналом любого ЧПУ – це автоматичне й точне управління рухом робочих органів. Робочі органи будь якого станку з ЧПУ має в собі дві або більше осі руху, рух по осям відбувається в автоматичному режимі з заданою точністю. Найбільш широко використовуваними типами осей є лінійні(що приводяться по прямому шляху) й обертові, яких осі обертання приходяться по

колу. Найчастіше керування робочими органами виконуються у вигляді подачі керуючих дискретних сигналів, прямокутних імпульсів, що надходять до драйверу крокового двигуна.

Ядром систем чисельного програмного управління є керуючий пристрій, який має апаратну й програмну частину, програмне забезпечення виконує керування робочими процесами і відповідає на реєстрацію подій, які надсилають датчики, так відклик відповідних на них реакцій змінює поведінку виконання робочого процесу станку в реальному часі.

На сьогоднішній день не складно сформулювати основний концептуальний підхід що до розробки систем чисельної обробки даних верстату будь якого призначення, достатньо проаналізувати комплекси ключових положень, які визначають загальну направленість, що до підходу конструювання пристрою, структурну схему, та якщо є можливість, використаної технології. Розберемо це, на «живому» загальнодоступному прикладі, системи ЧПУ класу персональний CNC, пристрій NC-201M. В інструкції з експлуатації є достатньо інформації, що надає основні концептуальні підходи до розробки чисельної системи керування.

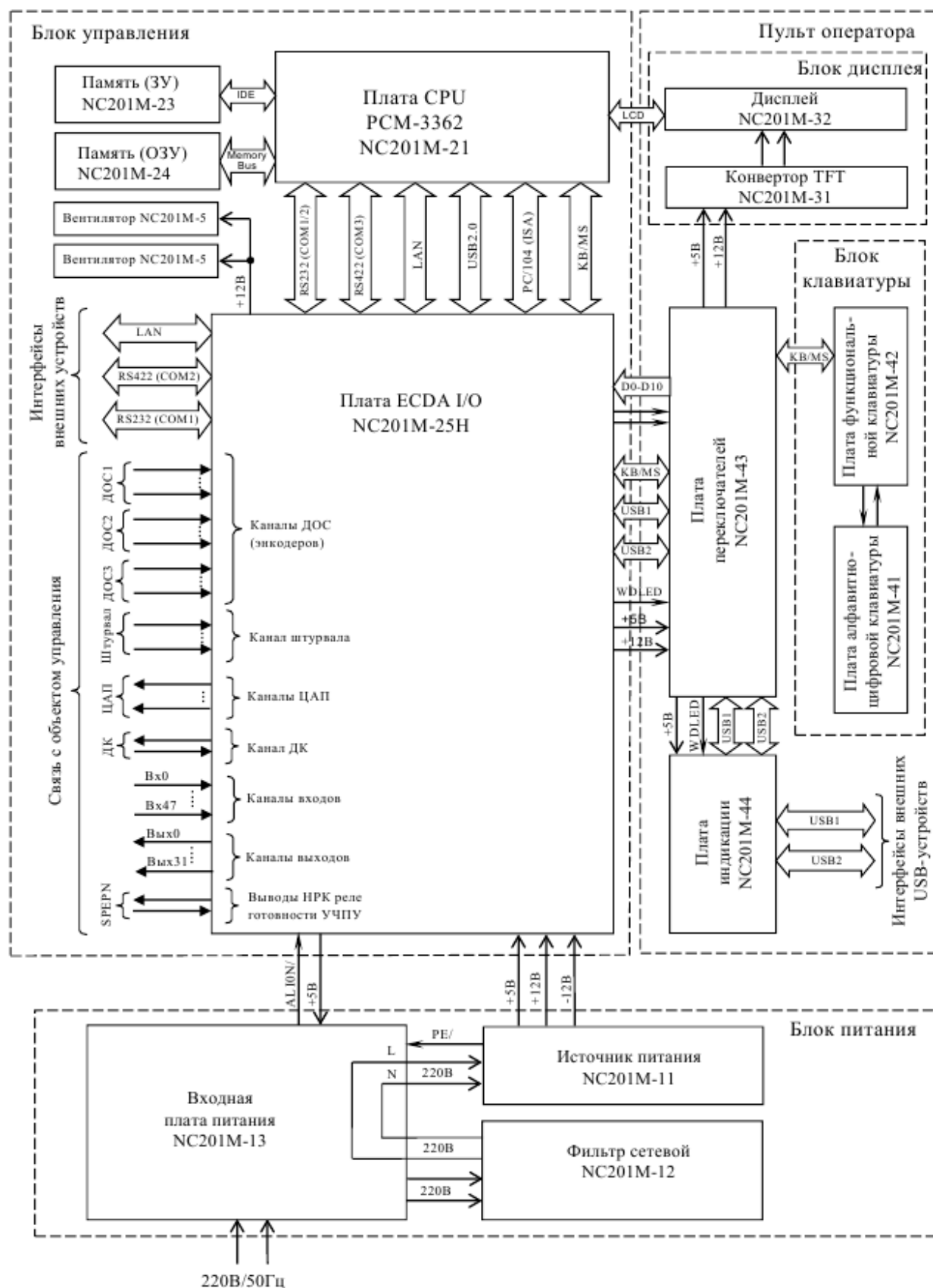


Рисунок 2.1 – Структурна схема пристрою ЧПУ NC-201M

Маючи на увазі вище оглянуті системи які маємо на ринку, не індустріального характеру, та ознайомившись з інструкцією пристрою NC-201M, виявляються певні умовні стратегічні підходи, які пересікаються з розгляненими приладами. Ядром блоку керування виступає плата CPU. Взаємодію модулю блоку керування забезпечують сигнали зовнішніх локальних шин, через канали модуля ECDA I/O відбувається керування периферійними пристроями станку, як датчиків так і робочих органів. Плата ECDA I/O забезпечує двонаправлений зв'язок, опитування та керування, між пристроєм ЧПУ й електрообладнанням станку. Обмін інформацією відбувається під керуванням програмного забезпечення який виконується на платі CPU. Схожий підхід, був оголошений в описі пристрою RMHV3.1 я використовує архітектуру ARM процесор виконує обробку даних, а матриця FPGA відповідає за базовий алгоритм генерації керуючих імпульсів.

2.2 Порівняння технологій

Для подальшої розробки системи проведемо порівняння існуючих технологій передачі даних, апаратних платформ, що підходять для розробки системи числової обробки даних – DPS-контролеру

2.2.1 Технології передачі даних

Список найбільш поширених інтерфейсів передачі даних:

- UART;
- SPI;
- I²C;
- RS-232/422/484;
- USB.

UART (англ. universal asynchronous receiver/transmitter — універсальний асинхронний приймач/передавач). При підключенні UART, два пристрої безпосередньо спілкуються між собою. Передавальний UART перетворює паралельні дані з керуючого пристрою у послідовну форму, послідовно передає їх в приймаючий UART, який потім перетворює послідовні дані назад у паралельні

дані для приймального пристрою. Для передачі даних між двома UART потрібні лише два дроти. Дані надходять від виводу Tx передавального UART до виводу Rx приймаючого UART, як це показано на рис. 2.2.

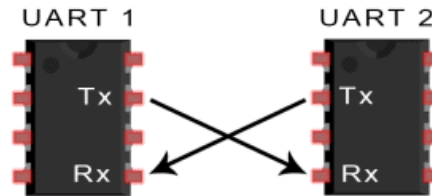


Рисунок 2.2 – Схема підключення UART

UART передають дані асинхронно, що означає, що відсутній тактовий сигнал для синхронізації виходу бітів від передавального UART до вибірки бітів приймаючим UART. Замість тактового сигналу передавальний UART додає біти запуску та зупинки до переданого пакету даних. Ці біти визначають початок і кінець пакета даних, щоб отримуючий UART знав, коли починати читати біти.

Коли приймаючий UART виявляє стартовий біт, він починає зчитувати вхідні біти з певною частотою, відомою як швидкість передачі даних. Обидва UART повинні працювати приблизно з однаковою швидкістю передачі. Швидкість передачі даних між передавальним і приймаючим UART може відрізнятися лише приблизно на 10%, перш ніж час бітів стане занадто далеко.

Обидва UART також повинні бути налаштовані на передачу та отримання однакової структури пакетів даних.

За допомогою UART працює безліч низькорівневих сенсорів, такі як детектори диму, вологості, освітлення, руху та інше.

SPI (англ. Serial peripheral Interface) – послідовний периферійний інтерфейс, шина, був розроблений для зв'язку периферії мікроконтролера. Вважається самим широко використовуваним інтерфейсом для з'єднання мікросхем.

Шина SPI базується на принципі провідний-підлеглий (Master-Slave). У якості провідного шини зазвичай є мікроконтролер, не може бути включенням й

програмуєма логіка, або спеціалізована інтегральна мікросхема. Підключенні к провідній шині, зовнішні пристрої, вважаються підлеглими.

Передача даних завжди синхронізована з загальним тактовим сигналом, генеруючим провідним пристроєм, а приймаюча сторона синхронізує отримані бітові послідовності з тактовим сигналом. До одного послідовного периферійного інтерфейсу провідного пристрою-мікросхеми може приєднуватися кілька мікросхем. Провідний пристрій обирає ведену-периферію для передачі, активуючи сигнал «вибір кристалу» на відому мікросхему. Периферія, яка не обрана, не приймає участі в передачі по SPI.

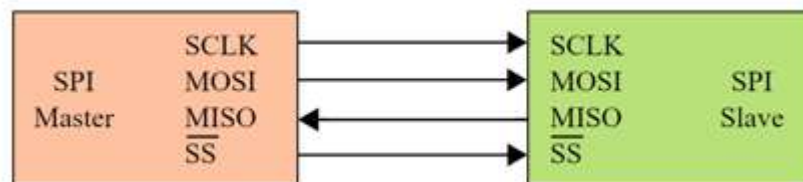


Рисунок 2.3 – Схема підключення SPI

I²C (Inter-Integrated Circuit) – це двонаправлена шина розроблена фірмою Philips, з послідовним форматом передачі даних та можливістю адресації й паралельного підключення до шини у 128 пристроїв. Шина I²C містить дві сигнальні лінії, одна з яких (SCL) призначена для передачі тактового сигналу, інша (SDA) - для передачі даних. Лінії шини повинні бути підключені до джерела живлення +5 В через резистори опором 1-10 кОм, такий підхід потрібен для формування сигналів що проводиться вихідними каскадами з відкритим колектором, тому проdivляється залежність від фізичної довжини ліній й швидкості передачі даних. Паралельне підключення вихідних каскадів усіх вузлів дозволяє їм впливати на сигнали, що передаються та брати участь в управлінні обміном даними. Топологія "загальна шина" забезпечує тільки напівдуплексний обмін даними: в будь-якому циклі обміну один пристрій передає дані, а всі інші - тільки їх приймають.

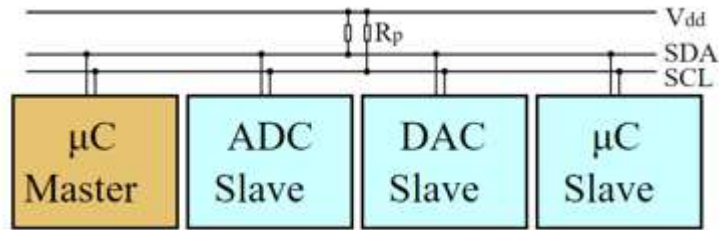


Рисунок 2.4 – приклад схеми підключення I²C з одним Master

RS-232/422/485 Під аббревіатурою RS (Recommend Standard) з цифровими позначеннями RS-232, RS-422 і RS-485 мається на увазі інтерфейси для цифрової передачі даних. Стандарт RS-232 більш відомий як звичайний COM порт комп'ютера або послідовний порт. Інтерфейси RS-422 і RS-485 широко застосовуються в промисловості для з'єднання різного обладнання,

USB (Universal Serial Bus - «універсальна послідовна шина») послідовний інтерфейс для підключення периферійних пристроїв до обчислювальної техніки. Отримав широке розповсюдження і фактично став основним інтерфейсом підключення периферії до побутової цифрової техніки.

2.2.2 Апаратні платформи ядра

Ядро системи чисельного керування станку повинно бути посередником між апаратними складовими та програмним забезпеченням. Таким чином апаратна платформа для ядра системи повинна мати змогу працювати з всіма перерахованими в розділі 2.2.1 технологіями для передачі даних та передавати їх на більш високий рівень. Відіграти цю роль може:

- ASUS Tinker board;
- Arduino;
- Raspberry Pi;
- Intel Edison;
- персональний комп'ютер.

Asus Tinker Board

Asus Tinker Board S (рис. 2.5) - це одноплатний комп'ютер (SBC) на базі ARM з чотириядерним процесором, 2 ГБ оперативної пам'яті та підтримкою 4K відео та HD аудіо. Це чудовий комп'ютер для любителів IoT, DIY, і виробників подібних систем.

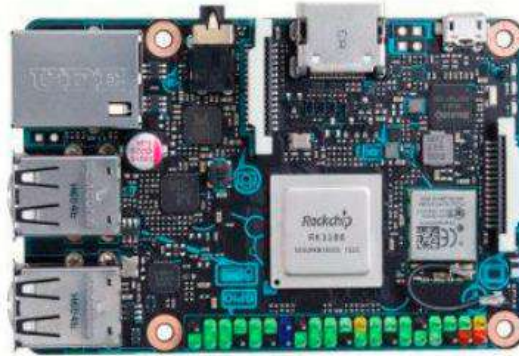


Рисунок 2.5 – Плата Asus Tinker Board

Tinker Board має стандартні варіанти підключення такі як USB, Ethernet та 40-контактний інтерфейс GPIO з підтримкою UART, ШІМ, I²C, SPI. Tinker Board оснащений підключенням DSI MIPI для дисплеїв та сенсорних екранів. Коннектор CSI MIPI призначений для підключення до сумісних камер, що забезпечують комп'ютерний зір, та багато іншого.

Tinker Board також оснащений RJ-45 Gbit для Інтернету та підключення до мережі. Інтегрований контролер Wi-Fi та Bluetooth на платі Tinker захищений металевією кришкою, щоб забезпечити мінімальні перешкоди та покращену продуктивність радіо. Вбудований роз'єм антени IPEX дозволяє легко замінити або оновити антену.

Tinker Board також має повнорозмірний вихід HDMI. Крім того, він включає чотири порти USB 2.0 для розширеного периферійного та додаткового підключення. Суттєвим недоліком даної плати є висока вартість.

Arduino

Arduino - це платформа з відкритим кодом, яка використовується для побудови проектів електроніки. Arduino складається як з фізичної програмованої плати (часто її називають мікроконтролером, рисунок 2.6), так і з частини програмного забезпечення, або IDE (інтегрованого середовища розробки), що працює на комп'ютері, і використовується для запису та завантаження комп'ютерного коду на фізичну плату.

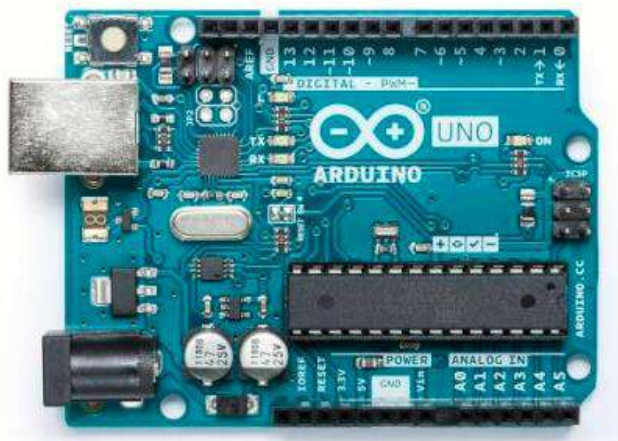


Рисунок 2.6 – Плата Arduino

Платформа Arduino стала досить популярною серед людей, які тільки починають займатися електронікою, і це не даремно. На відміну від більшості програмованих друкованих плат, Arduino не потребує окремого апаратного забезпечення (програматора) для завантаження нового коду до плати - ви можете просто використовувати кабель USB. Крім того, IDE Arduino використовує спрощену версію C ++, що полегшує навчання програмуванню. Нарешті, Arduino пропонує стандартний форм-фактор, який розбиває функції мікроконтролера на більш доступний пакет.

Arduino реалізується на базі мікроконтролерів ATMELE AVR та, якщо треба, може слугувати драйвером між низькорівневими приладами та високорівневими, але це потребує написання відповідної поведінки для кожного приладу окремо.

Raspberry Pi

Raspberry Pi - це серія одноплатних комп'ютерів, створена Фондом Raspberry Pi, британською благодійною організацією, яка має на меті навчити людей обчислювальній техніці та полегшити доступ до обчислювальної освіти.

Raspberry Pi вийшов у 2012 році, і з тих пір було випущено кілька ітерацій та варіацій. Оригінальний Pi мав однопоточний процесор на 700 МГц і лише 256 мегабайт оперативної пам'яті, а остання модель має чотири поточний процесор на 1,5ГГц з 4 гігабайтом оперативної пам'яті(Raspberry Pi 4 B+). Твердий диск відсутній, натомість використовується SD карта. Комп'ютер зображено на рисунку 2.7.

Основною ціною на Raspberry Pi завжди було \$ 35, а всі моделі - \$ 35 або менше, включаючи Pi Zero, яка коштує всього \$ 5.



Рисунок 2.7 – Плата Raspberry Pi 3b+

У всьому світі люди використовують Raspberry Pi для вивчення навичок програмування, побудови апаратних проектів, домашньої автоматизації та навіть використання їх у промислових додатках.

Raspberry Pi - це дуже дешевий комп'ютер, який працює під управлінням Linux, але він також пропонує набір шпильок GPIO (general purpose input/output - загального призначення введення / виведення), які дозволяють управляти електронними компонентами для фізичних обчислень та досліджувати Інтернет речей (IoT).

Ніжки плати здатні працювати як виходи мікроконтролера та отримувати рівень сигналу, емулювати UART, працювати з I²C та іншими технологіями передачі даних за допомогою вже існуючих драйверів та бібліотечних рішень.

Апаратна начинка з процесором архітектури ARM дозволяє запускати веб сервери, сервери баз даних та інше. Так, наприклад, платформа .NET core доступна як в Runtime, так і SDK. Найбільш популярна мова програмування – Python, бібліотечні рішення можна знайти буквально для будь яких потреб.

Intel Edison

Intel Edison - це невелика плата, трохи більша за SD-карту ($35,5 \times 25,0 \times 3,9$ мм), рисунок 2.5, що має двоядерний двопотоковий процесор Intel® Atom™ на 500 МГц та 32-розрядний мікроконтролер Intel® Quark™ на 100 МГц.

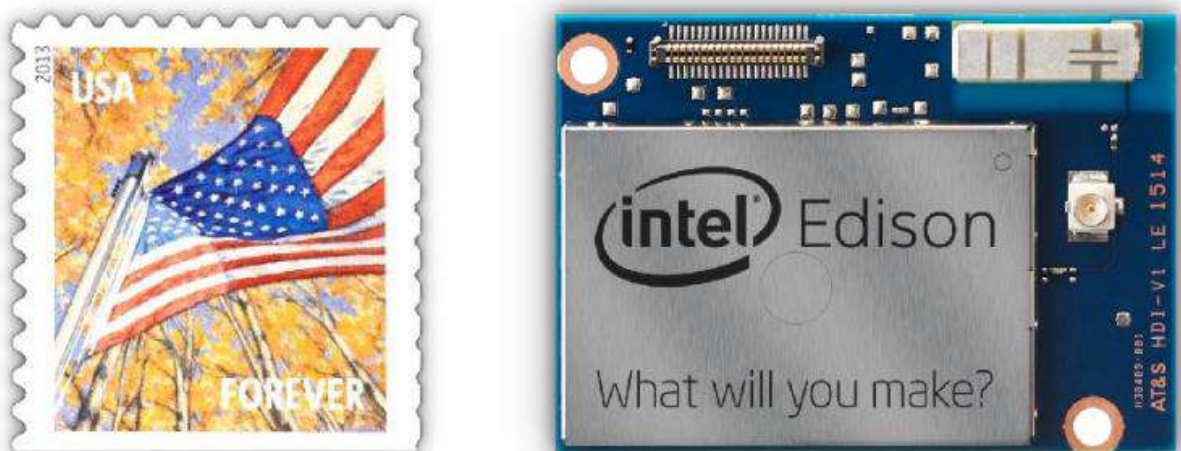


Рисунок 2.8 – Плата Intel Edison

Доступна пам'ять - 1 Гб оперативної пам'яті та 4 Гб флеш-пам'яті. На борту є WiFi та Bluetooth. Він має 1 USB-контролер OTG та всі інтерфейси, про які виробник IoT може мріяти, включаючи підтримку UART, I2C, SPI, I2S, GPIO та SD Card. Незважаючи на свої розміри, плата має:

- 20 цифрових входів/ виходів, включаючи 4 виходи з підтримкою ШІМ;
- 6 аналогових входів;
- 1 UART (RX / TX);
- 1 I2C;
- 1 ICSP 6-контактний роз'єм (SPI);
- Роз'єм USB Type-A;
- Роз'єм SD-карти;

РС

У якості ядра системи керування станком, звісно, можна використовувати персональний комп'ютер. Для розробки та прототипування, завдяки різноманітним рішенням для розробки вибору програмних інструментів, він підходить ідеально, також має величезну (порівняно з попередніми апаратними платформами) потужність, можливість використовувати x86 програмне забезпечення баз даних, тощо.

Однак слід зазначити, що підтримку таких інтерфейсів як UART, I²C, SPI, RS або простих логічних сигналів, необхідно реалізовувати за допомогою плат розширення. Також суттєвим недоліком є енерговитратність та шум.

2.2.3 Апаратні платформи для розв'язки та передачі даних

Для подальшої розробки проведемо порівняння існуючих платформ які найбільш підійдуть на роль блоку, який буде відповідати за базовий алгоритм генерації керуючих імпульсів та передачі даних, що підходять для вибраного підходу що до розробки системи числової обробки даних – DPS-контролеру.

Список найбільш підходящих та доступних платформ мікро ЕОМ:

- AVR;
- PIC;
- ПЛІС.

AVR

Мікроконтролери Гарвардської архітектури, раніше випускалися американською компанією Atmel Corporation нині Microchip Technology Inc. Аббревіатура AVR це назва модифікованого 8-бітного однокристального мікроконтролера з RISC архітектурою Гарвардського університету. Архітектура AVR була розроблена студентами Альф-Егіль Богеном та Вегардам Волланом. Назва AVR походить від імені цих людей які й розробили цю архітектуру, котра також відома як “Advanced Virtual Risc”.

У номенклатурі Microchip Technology Inc. представлений широкий спектр 8-ми, 16-ти і 32-бітних мікроконтролерів та цифрових сигнальних контролерів під маркою AVR. Відмінною особливістю AVR-контролерів є хороша спадкоємність різних сімейств. Це і програмна сумісність (безкоштовне середовище розробки AtmelStudio IDE, C++\C-компілятори від GCC), і сумісність з виводами, за периферією, за напругами живлення, за способами розробки за бібліотеками і стеками найбільш популярних комунікаційних протоколів. Номенклатура нараховує більш 500 різних контролерів з будь-якими варіаціями периферії, пам'яті, кількістю виводів, продуктивністю, діапазонами живлення і температури та т.ін.

Як акцентувалося вище мікроконтролери AVR мають у своїй основі Гарвардську RISC архітектуру і діляться на 3 стандартні сімейства: Tiny (ATtinyxxxx), Mega (ATmegaxxxx) та Xmega (ATxmegaxxxx).

Мікроконтролери сімейства AVR відносяться к класу 8-розрядних мікроконтролерів. Це означає, що переважну більшість операцій процесори проводять з восьмирозрядними двійковими числами. З цієї причини встроєна шина даних у цих контролерів також восьмирозрядна. Усі осередки й більшість регістрів мікроконтролера також восьмирозрядні.

За для обробки шістнадцятирозрядних чисел деякі внутрішні регістри можуть об'єднуватися попарно. Кожна така пара має змогу працювати як один

шістнадцятирозрядний регістр. Виняток становить пам'ять програм. Вона цілком складається з шістнадцятирозрядних осередків.

Мікроконтролери AVR виготовляються за КМОП-технологію, завдяки цьому вони мають достатньо високу швидкодію та низьке енергоживлення. Більшість команд виконується за один такт тому їх швидкодія може досягати до шістнадцяти мільйонів операцій за секунду при тактовій частоті 16 МГц.

Сімейство Tiny (ATtinyxxxx) представляє із себе спрощену архітектуру базового використання, що надає більш дешеве рішення з пропонованих Microchip.

Переваги такої архітектури наступні:

Мікроконтролери Гарвардської архітектури, раніше випускалися американською компанією Atmel Corporation нині Microchip Technology Inc. Аббревіатура AVR це назва модифікованого 8-бітного однокристального мікроконтролера з RISC архітектурою Гарвардського університету. Архітектура AVR була розроблена студентами Альф-Егіль Богеном та Вегардам Волланом. Назва AVR походить від імені цих людей які й розробили цю архітектуру, котра також відома як "Advanced Virtual Risc".

У номенклатурі Microchip Technology Inc. представлений широкий спектр 8-ми, 16-ти і 32-бітних мікроконтролерів та цифрових сигнальних контролерів під маркою AVR. Відмінною особливістю AVR-контролерів є хороша спадкоємність різних сімейств. Це і програмна сумісність (безкоштовне середовище розробки AtmelStudio IDE, C++\C-компілятори від GCC), і сумісність з виводами, за периферією, за напругами живлення, за способами розробки за бібліотеками і стеками найбільш популярних комунікаційних протоколів. Номенклатура нараховує більш 500 різних контролерів з будь-якими варіаціями периферії, пам'яті, кількістю виводів, продуктивністю, діапазонами живлення і температури та т.ін.

Як акцентувалося вище мікроконтролери AVR мають у своїй основі Гарвардську RISC архітектуру і діляться на 3 стандартні сімейства: Tiny (ATtinyxxxx), Mega (ATmegaxxxx) та Xmega (ATXmegaxxxx).

Мікроконтролери сімейства AVR відносяться к класу 8-розрядних мікроконтролерів. Це означає, що переважну більшість операцій процесори проводять з восьмирозрядними двійковими числами. З цієї причини встроєна шина даних у цих контролерів також восьмирозрядна. Усі осередки й більшість регістрів мікроконтролера також восьмирозрядні.

За для обробки шістнадцятирозрядних чисел деякі внутрішні регістри можуть об'єднуватися попарно. Кожна така пара має змогу працювати як один шістнадцятирозрядний регістр. Виняток становить пам'ять програм. Вона цілком складається з шістнадцятирозрядних осередків.

Мікроконтролери AVR виготовляються за КМОП-технологію, завдяки цьому вони мають достатньо високу швидкодію та низьке енергоживлення. Більшість команд виконується за один такт тому їх швидкодія може досягати до шістнадцяти мільйонів операцій за секунду при тактовій частоті 16 МГц.

Сімейство Tiny (ATtinyxxxx) представляє із себе спрощену архітектуру базового використання, що надає більш дешеве рішення з пропонованих Microchip.

Переваги такої архітектури наступні:

- Flash-ROM до 16Кб; SRAM до 512б; ERROM до 512б;
- число ліній вводу-виводу 4-18 (загальна кількість ліній 6-32);
- обмежений набір периферійних пристроїв.

Сімейство Mega (ATmegaxxxx) середній сегмент і має більш розширену систему переривань, а також багатий набір периферії такий як SPI, I²C, USART, компаратори, таймери та АЦП.

Зазначена архітектура наступна:

- Flash-ROM до 256Кб; SRAM до 16Кб; ERROM до 4Кб;
- число ліній вводу-виводу 23-86 (загальна 28-100);
- апаратний помножувач;
- розширена система команд й периферійних пристроїв.

Xmega (ATxmegaxxxx) високопродуктивне сімейство 8-бітних мікроконтролерів AVR на момент цієї розробки. Характерними відмінностями

даного сімейства більш вдосконалена і розширена архітектура та периферійна система. Завдяки DMA контролеру вдалося знизити навантаження на центральний процесор, усі периферійні пристрої при передачі даних можуть працювати у режимі DMA. Тому система обробки подій забезпечує взаємодію периферійних пристроїв з 100% передбачуваністю часу реакції на подію. Продуктивність достатня для роботи у режимі реального часу.

На додаток сімейство Xmeag має при собі технологію Atmel picoPower що надає змогу працювати кристалу з напругою 1.8В при споживанні струму 600нА при працюючих годинах реального часу й збереження даних в SRAM для швидкого виходу з режиму сну.

Зазначена архітектура відзначає наступна:

- Flash-ROM до 384Кб; SRAM до 32Кб; ERROM до 4Кб;
- чотирьохканальний DMA-контролер;
- інноваційна система обробки подій.

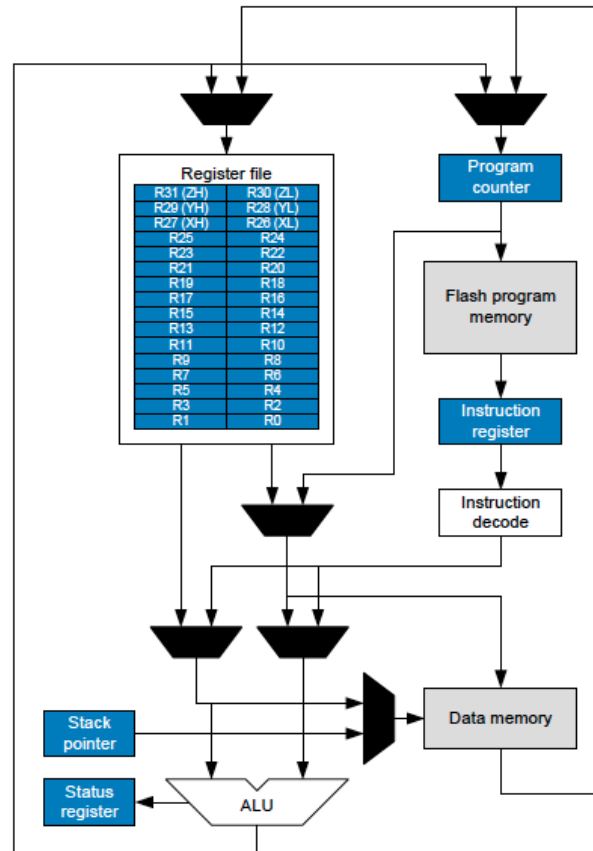


Рисунок 2.9 – блок схема архітектури AVR

В представленому дипломному проєкті використано мікроконтролер АТмега32 структурна схема якого зображена на рисунку 1.1.

PIC

Це також серія мікроконтролерів Гарвардської архітектури, являється вже власною розробкою американської компанії Microchip Technology. Початковим призначенням контролерів серії PIC було розширення можливостей вводу виводу 16-бітових мікропроцесорів CP-1600, звідти й пішла назва серії що з англ. Peripheral interface controller.

Загалом серії мікроконтролерів під маркою PIC має декілька основних видів архітектури рисунок 2.10.

	Baseline Architecture	Mid-Range Architecture	Enhanced Mid-Range Architecture	PIC18 Architecture
Pin Count	6-40	8-64	8-64	18-100
Interrupts	No	Single interrupt capability	Single interrupt capability with hardware context save	Multiple interrupt capability with hardware context save
Performance	5 MIPS	5 MIPS	8 MIPS	Up to 16 MIPS
Instructions	33, 12-bit	35, 14-bit	49, 14-bit	83, 16-bit
Program Memory	Up to 3 KB	Up to 14 KB	Up to 28 KB	Up to 128 KB
Data Memory	Up to 138 Bytes	Up to 368 Bytes	Up to 1,5 KB	Up to 4 KB
Hardware Stack	2 level	8 level	16 level	32 level
Features	- Comparator 8-bit ADC - Data Memory - Internal Oscillator	In addition to Baseline: - SPI/I²C™ - UART - PWMs - LCD 10-bit ADC - Op Amp	In addition to Mid-Range: - Multiple Communication Peripherals - Linear Programming Space - PWMs with Independent Time Base	In addition to Enhanced Mid-Range: 8x8 Hardware Multiplier - CAN - CTMU - USB - Ethernet 12-bit ADC
Highlights	Lowest cost in the smallest form factor	Optimal cost to performance ratio	Cost effective with more performance and memory	High performance, optimized for C programming, advanced peripherals

Рисунок 2.10 – чотири категорії 8-бітного сімейства PIC

Як зазначалось 8-бітний мікроконтролер PIC використовує архітектуру зі скороченим набором команд (RISC), який має окремі адресу і шину даних. Це

означає, що вони також можуть бути розділені за розміром. Всі 8-бітові мікроконтролери PIC мають 8-бітну шину даних, але програмна шина буде відрізнятися за розміром в залежності від сімейства. Це може викликати плутанину, оскільки структура пам'яті 8-бітного мікроконтролера PIC буде повідомлятися словами, розмір яких дорівнює програмній шині, а не шині даних. Більша програмна шина дозволяє пристрою розміщувати більше інформації в одному рядку команд і, завдяки чому надається змога більш ефективно її виконувати, звідси пам'ять для кожного пристрою вказується словами, а потім – байтовим посиленням.

Для прикладу, 8 Кбайт програмної пам'яті на 14-бітному основному пристрої містять 8 Кбайт програмного простору, що еквівалентно 14 Кбайтного простору в байтах.

Всі 8-бітові контролери PIC виконують команди збірки за один цикл команд, за винятком розгалужень і вибірок. Швидкість генератора в більшості пристроїв ділиться на чотири для отримання тактової частоти команд, але в удосконалених частинах є функція фазового автопідстроювання частоти (PLL), яка дозволяє підвищити частоту генератора на чотири перед поділом, щоб отримати співвідношення між генераторами один до одного, швидкості й тактової частоти інструкції.

Baseline Architecture - Базові мікроконтролери PIC використовують 12-бітне командне слово і надають необхідну кількість функцій і опцій, щоб мінімізувати витрати і виконати роботу правильно. Baseline має найпростішу архітектуру з сімейства 8-бітних, і тому з ними найпростіше працювати і розуміти, чим і користуються першопочатківці. Вони мають:

- простий набір з 33 (12-бітних) інструкцій;
- адресна програмна пам'ять об'ємом 2 тис. Слів (3 КБ) 144 байти RAM (макс.);
- дворівневий апаратний стек;
- один (8-бітний) регістр вибору файлу;
- безліч варіантів продукту і проста міграція;

- найменші доступні форм-фактори;
- базову архітектуру можна визначити за структурою номерів: 10Fxxx, 12Fxxx та 16Fxxx.

Mid-Range Architecture - середній рівень мікроконтролерів є наступним рівнем продуктивності що унаслідкує функції мікроконтролерів Baseline PIC. Ці багатофункціональні периферійні пристрої, що використовують 14-бітове командне слово, ідеально підходять для багатьох додатків, що вимагають вищого рівня вбудованого управління і більшого обсягу пам'яті.

- 35 простих інструкцій (шириною 14 біт) для вивчення;
- адресна програмна пам'ять об'ємом 8 тис. Слів (14 КБ);
- 368 байт ОЗУ (макс.);
- 8-рівневий апаратний стек;
- один (9-бітний) регістр вибору файлу;
- обробка апаратних переривань;
- високо інтегрований набір функцій з : EEPROM, ЖК-дисплеєм, сенсорні рішення mTouch™ і послідовний зв'язок;
- paradise city архітектуру можна визначити за структурою номерів: 10Fxxx, 12Fxxx та 16Fxxx.

Enhanced Mid-Range Architecture - це новітнє сімейство, вдосконалене ядро середнього рівня, яке засноване на кращих елементах ядра середнього рівня і забезпечує додаткову продуктивність й зберігаючи при цьому сумісність з мікроконтролерами PIC середнього рівня для істинної міграції продукту. Ці пристрої мають новітні функції і найнижчим енергоспоживанням серед 8-бітних ядер. Вдосконалене ядро додає більше програмної пам'яті і збільшує швидкість роботи. Вони також мають найвищу точність та внутрішніми генераторами найвищої частоти.

- 49 (14-бітних) складальних команд;
- адресна програмна пам'ять об'ємом 32 КБ (56 КБ);
- 4 КБ ОЗУ (макс.);
- 16-рівневий апаратний стек;

- два (16-бітних) регістра вибору файлу;
- апаратна обробка переривань зі збереженням контенту;
- розширений набір функцій, кілька послідовних комунікацій і можливість управління двигуном;
- Enhanced Mid-Range архітектуру можна визначити за структурою номерів: 12F1xxx й 16F1xxx.

High-End У цієї архітектури є власний префікс, а саме PIC18. Це сімейство поєднує в собі максимальний рівень продуктивності і інтеграції з простотою використання 8-бітної архітектури. Володіючи обчислювальною потужністю до 16 MIPS, мікроконтролери PIC18 оснащені розширеними периферійними пристроями, такими як CAN, USB, Ethernet, ЖК-дисплей і CTMU. Вони пропонують найбільшу кількість висновків і обсяг пам'яті в 8-бітному сімействі компонентів. Архітектура оптимізована для програмування на C.

- 83 (шириною 16 біт) інструкції по збірці;
- адресна пам'ять програм до 2 МБ;
- 4 КБ ОЗУ (макс.);
- 32-рівневий апаратний стек;
- один (8-бітний) регістр вибору файлу;
- Вбудоване апаратне множення 8x8;
- найвища продуктивність 8-бітної архітектури;

High-End архітектуру можна визначити за структурою номерів: 18Fxxx, 18FxxJxx, та 18FxxKxx.

Всі 8-бітові пристрої можна програмувати / налагоджувати за допомогою одного і того ж набору засобів розробки Microchip MPLAB® X IDE єдиної інтегрованої середовища розробки що підтримує усі мікроконтролери PIC, для певного типу мікроконтролерів є свій компілятор, для 8-бітних пристроїв MPLAB XC, є безкоштовним всі версії доступні без обмежень до часу або пам'яті.

ПЛІС

На відміну від вище описаних технологічних рішень цифрової електроніки, в основі яких була взаємодія мікропроцесорного ядра з внутрішньою периферією

(таймери, лічильники, ADC-перетворювачі), програмно інтегральні логічні схеми (ПЛІС) – це кристал який має в собі набір кілка десятків тисяч, є і більше, логічних елементів. На відміну від цифрових мікросхем, логіку роботи ПЛІС не визначають на етапі виготовлення, а задають за допомогою програмування/запису, спеціально відведеної для цього пам'яті, стан якої комує усю внутрішню логіку взаємодії логічних елементів між собою. Для цього служать програматори, й налагоджувальні середи розробки, що дозволяють створити бажану структуру цифрового пристрою у вигляді принципової електронної схеми або за допомогою спеціальних мов описання апаратури (Verilog, VHDL, AHDL). Завдяки цьому отримується висока швидкодія й гнучкість, що надає можливість, не змінюючи чип, зробити/прототипувати проекти багацького спектру задач.

Основним критерієм при виборі ПЛІС є кількість програмованих блоків. Як правило це залежить від конкретного ПЛІС кількість блоків може змінюватися у широких межах, також це впливає в значному степені на вартість пристрою. На відміну від мікроконтролеру який послідовно виконує записану у його пам'ять програму, ПЛІС виконує задачу паралельно й по можливості незалежно, тому порівнювати характеристику тактової частоти ПЛІС й мікроконтролерів не є доцільним.

На сьогодні найбільш актуальними є два види ПЛІС:

Complex Programmable Logic Device (CPLD) – це, можна сказати, класичний ПЛІС зі встроєною енергонезалежною пам'яттю, в яку загружається прошивка. Внутрішня структура побудована на матриці з логічних блоків, а кількість елементів в них лежить у межах сотень й тисяч штук. Завдяки відносній простоті коштують менше ніж наступний вид програмованої логіки. Зазвичай CPLD використовують у схемах, де потрібна висока швидкість робити та велике число виводів, й при цьому виконувати задачі не високого рівня складності.

Field-Programmable Gate Array (FPGA) – програмована користувачем вентиля матриця, однак її часто відносять к ПЛІС, є більш розвинутим пристроєм на відміну від CPLD, побудовані на логічних блоках с гнучкою комутацією й містять велике число елементів десятки або сотні тисяч. Прошивка зберігається у

зовнішній енергонезалежній пам'яті. Крім простіших логічних елементів FPGA може мати в собі готові рішення обробки сигналів, все це дозволяє реалізувати пристрій для обробки даних й інші складні пристрої.

2.3 Особливості архітектури і розробки

Розробка архітектури пристрою та його програмного забезпечення є основним та визначним для подальшого розвитку проекту, він визначає складність

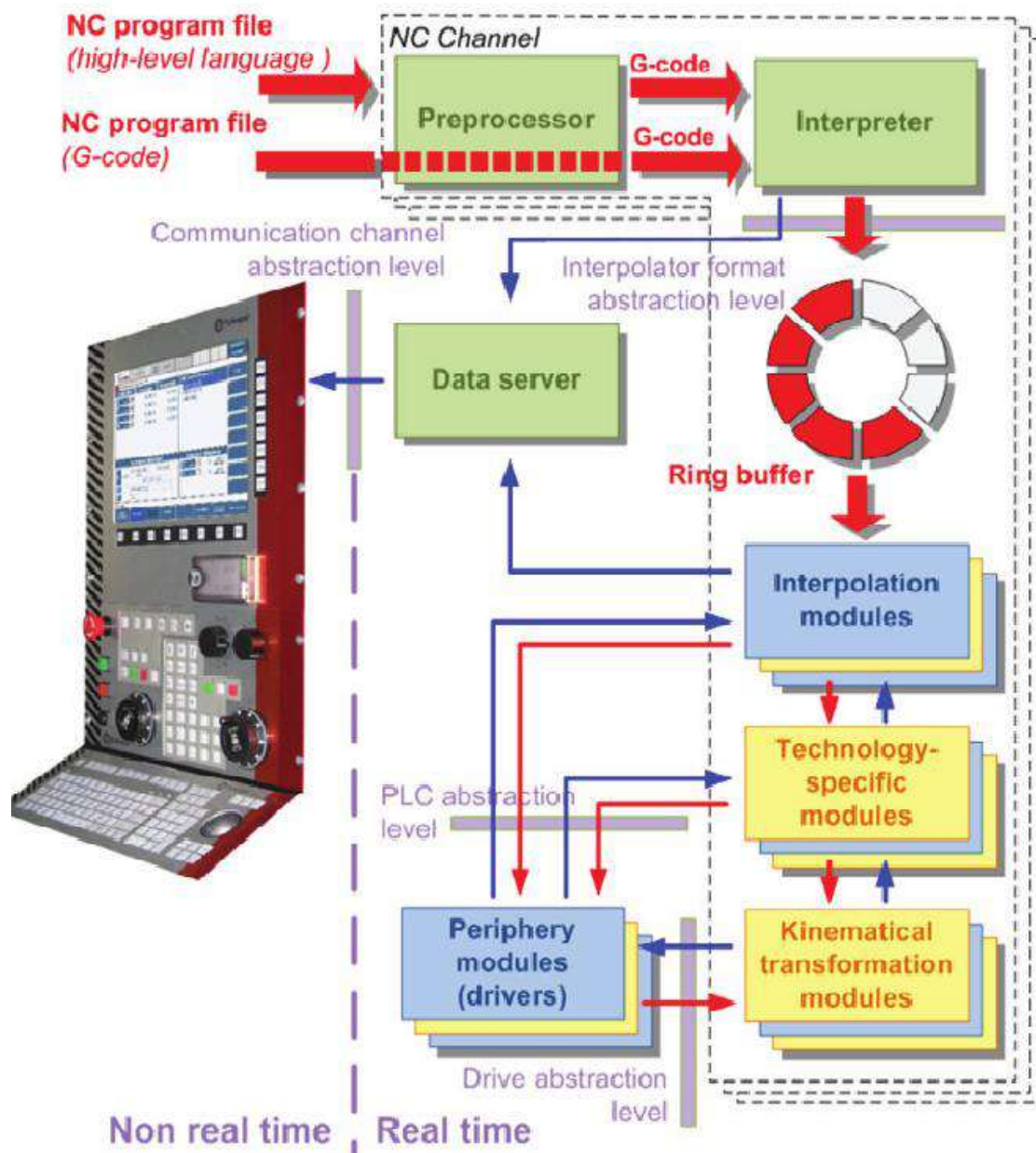


Рисунок 2.11 – приклад структури конвеєра обробки даних

подальших змін. Вибір оптимальної архітектури пристрою, як апаратної, так і програмної описано у наступних підрозділах.

2.3.1 Архітектура

Відповідно до концепції пристроїв ЧПУ на сьогоднішній день, архітектура яких системи є відкритою, виконують згідно одно-комп'ютерній моделі. Ця модель передбачає використання традиційного комп'ютера оснащеного додатковими контролерами для зв'язку з механізованими об'єктами керування. Це може бути різне виконання електронної схеми з інтегральними схемами, мікроконтролерами й ПЛІС. Таким чином зорганізується структура конвеєра обробки даних, між станком та операційною системою, приклад на рисунку 2.11.

2.3.2 Частина блоку ядра або серверна частина ЧПУ

Це основний блок, який реалізує основну логіку пристрою, важливішим чинником цього блоку є виконання операційної системи здатної підтримувати всі потрібні програмні засоби. В якості операційної системи може бути використана ОС Windows NT, однак яка не є системою реального часу і в цьому випадку вимагає відповідного розширення; наприклад, у вигляді системи RTX 4.1 американської фірми VentureCom. Система RTX модифікує шар HAL (Hardware Abstraction Layer) операційної системи Windows NT і доповнює його диспетчером потоків (threads) реального часу.

2.3.2 Частина блоку розв'язки

Розглянутий блок основна задача якого забезпечити прийом та передачу даних з серверною частиною, та на основі прийнятих даних генерувати керуючі імпульси, та збирати данні за датчиків станку.

3 ПРАКТИЧНА ЧАСТИНА

У приведеному розділі описується підхід що до процесу розробки персональної системи чисельного керування. Основна задача даної розробки це отримати базовий функціонал промислових пристроїв PCNC, а саме, керування шаговими двигунами (генерувати імпульси) на основі G-коду.

3.1 Вибір стеку технологій

На роль апаратного ядра системи керування пристроєм було обрано Raspberry PI, він підтримує перераховані у розділі 2.2.1 технології передачі даних, задовольняючи потреби чисельної потужності, має невеликі розміри, і є простим у налаштування та конфігуруванні програмних компонентів пристрою. Це дозволяє інтегрувати пристрій, без усіляких нарікань на складнощі, у PCNC систему, як якусь головну мікросхему, або плату CPU.

Питанням обробки даних на прикладному рівні, буде реалізовано за допомогою операційної системи Raspberry PI OS, для зчитування файлу керуючої програми, G-коду, зі змінного носія та робота з потоками й інтересами буде реалізовано за допомогою вже встроєних засобів інтерпретатору Python.

Оскільки напруга логічних рівнів Raspberry Pi досягає 3,3В і є максимальною для входів/виходів мікрокомп'ютера, такого рівню може бути не достатньо для керування пристроями силової електроніки драйверів або інверторів. До того ж виводи мікрокомп'ютеру GPIO не як не захищені подавши напругу, наприклад 5 В, можна вивести пристрій із ладу,

Маючи на увазі результати аналізу відкритих апаратних систем, а саме плат розв'язки, таких пристроїв як ЧПУ NC-201M(розділ 2.1), та RMHV3.1(розділ

1.3.1.2) для реалізації блоку генерування керуючих базових імпульсів було обрано мікроконтролер AVR ATmega16, оскільки він задовольняє вимогам чисельної потужності, та доступності інструментів розробки програмного забезпечення та бібліотек.

3.1 Налаштування Raspberry Pi

Даний розділ описує процес інсталяції операційної системи до пристрою Raspberry Pi, проводиться процес встановлення та налаштування операційної системи, що у подальшому буде відігравати роль ядра системи PCNC. За для зручнішої розробки та налагодження ОС, було прийнято рішення налаштування видаленої середовища розробки Visual Studio Code за допомогою SSH з'єднання, через технологію безпроводного зв'язу Wi-Fi.

3.1.1 Встановлення Raspberry OS

Серед різноманіття операційних, яких на сьогодні з десяток, вибір впав до RaspberryPi OS 32-bit Desktop ARM32 версія Linux Debian спеціально розроблена для RPi, вона має інструменти для налаштування апаратних інтерфейсів плати та вже встановлений інтерпретатор python.

Встановлення операційної системи відбувається за допомогою іншого комп'ютеру на SD накопичувач встановлюються образ системи, тому потрібно як мінімум 8 Гб пам'яті та рекомендується використовувати 10 клас SD карти з записом 10Мб/с.

Завантажуємо Raspberry Pi Imager. Наступний крок, необхідно вставити SD карту до пристрою зчитування MicroSD карт, деякі комп'ютери, зазвичай ноутбуки, вже мають у собі потрібний порт, якщо нема використовуємо USB адаптер під MicroSDcard.

Запускаємо Raspberry Pi Imager, обираємо образ представлений у розділі CHOOSE OS, та обираємо образ RaspberryPi OS 32-bit, далі необхідно обрати пристрій на який буде записана ОС кнопка CHOOSE STORAGE, обираємо



Рисунок 3.1 – інтерфейс Raspberry Pi Imager

потрібний носій, та натискаємо WRITE. Після закінчення процесу встановлення, вставляємо флеш накопичувач у слот для SD карт Raspberry Pi.

Для першого запуску необхідно підключити до плати клавіатуру, миш та монітор. Завершуємо процес інсталяції за допомогою графічного інтерфейсу. При першій же можливості налаштовуємо підключення до Wi-Fi мережі. Дізнаємось IP адресу яку надав нам DHCP за допомогою команди у терміналі:

```
>hostname -I
```

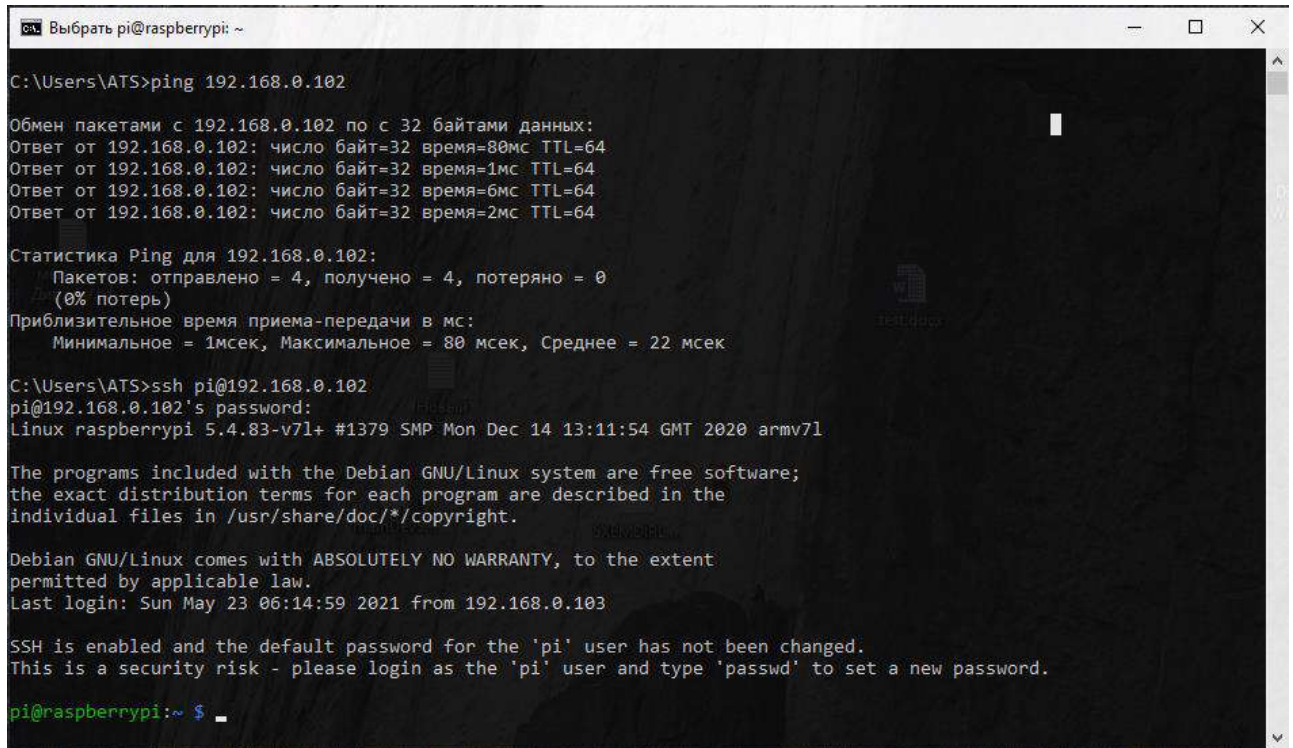
3.1.2 Налаштування віддаленої середи розробки за допомогою SSH

Використовувати мікрокомп'ютер RaspberryPi для розробки та конфігурування системи не завжди зручно, враховуючи важкість IDE, скористаємось можливостями видаленого доступу за допомогою SSH. Для початку переконаємось, що робочий комп'ютер має доступ до плати RPi через cmd.exe:

```
> ping 192.168.0.102
```

```
> ssh 192.168.0.102
```

Результатом виконання команди `ping` повинні бути 4 успішні пакети, а відповідь на запит `ssh` це введення паролю та віддалене підключення до терміналу ОС Raspbian (рисунок 3.2):



```

Выбрать pi@raspberrypi: ~
C:\Users\ATS>ping 192.168.0.102

Обмен пакетами с 192.168.0.102 по 32 байтами данных:
Ответ от 192.168.0.102: число байт=32 время=80мс TTL=64
Ответ от 192.168.0.102: число байт=32 время=1мс TTL=64
Ответ от 192.168.0.102: число байт=32 время=6мс TTL=64
Ответ от 192.168.0.102: число байт=32 время=2мс TTL=64

Статистика Ping для 192.168.0.102:
  Пакетов: отправлено = 4, получено = 4, потеряно = 0
  (0% потерь)
Приблизительное время приема-передачи в мс:
  Минимальное = 1мсек, Максимальное = 80 мсек, Среднее = 22 мсек

C:\Users\ATS>ssh pi@192.168.0.102
pi@192.168.0.102's password:
Linux raspberrypi 5.4.83-v7l+ #1379 SMP Mon Dec 14 13:11:54 GMT 2020 armv7l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Sun May 23 06:14:59 2021 from 192.168.0.103

SSH is enabled and the default password for the 'pi' user has not been changed.
This is a security risk - please login as the 'pi' user and type 'passwd' to set a new password.

pi@raspberrypi:~ $

```

Рисунок 3.2 – Результати операцій ping та ssh.

Після вдалого з'єднання завантажуюмо Visual Studio Code від компанії Microsoft за посиланням: <https://code.visualstudio.com/download>.

Завершивши процес інсталяції, для нормальної роботи через SSH з'єднання потребується завантажити розширення, а саме Remote Development та Remote – SSH(Nightly).

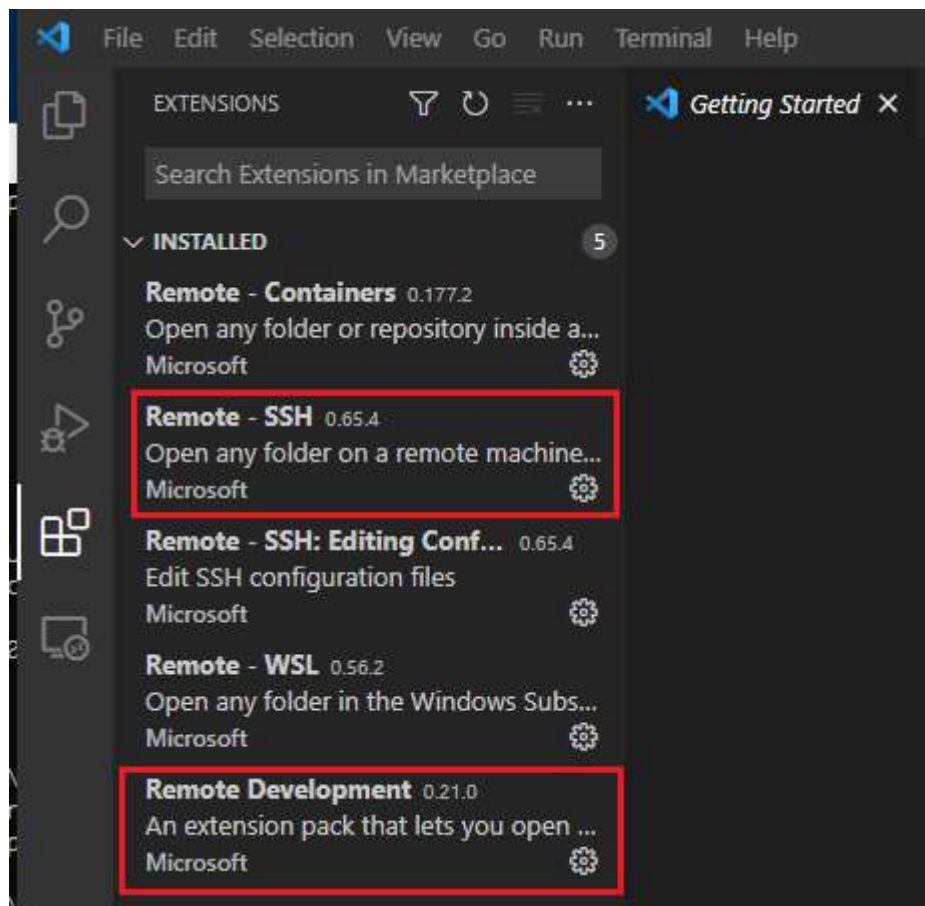


Рисунок 3.3 – Необхідні розширення для вдаленої розробки.

По завершенню встановлення розширень треба перезавантажити вікно редактору. Після цього натискаємо на іконку Remote explorer , далі у вікні remote explorer натискаємо «+» і приписуємо `pi@192.3.168.0.102`. Після введення паролю обираємо Linux та чекаємо, доки на віддалений пристрій завантажить усі бібліотеки що необхідні для роботи у режимі SSH. Після проведеної роботи IDE емулює роботу на віддаленому пристрої, завдяки цьому середі розробки доступна файлова система raspberry pi, та підтримка консольних команд. Всі наступні дії будуть проведені в консолі RPi за допомогою SSH.

3.1.3 Налаштування програмного забезпечення PyCNC

Даний підрозділ описує встановлення та налаштування програмного продукту PyCNC, який є безкоштовний високопродуктивний інтерпретатор G-коду

з відкритим вихідним кодом, що реалізую усі потрібні методи для роботи у якості ЧПУ або 3D-принтера. І може може працювати на різних платах ARM з ОС Linux, включаючи Raspberry Pi.

3.1.4 Встановлення PyCNC

Для початку, переведемо оновлення пакетів операційної системи Raspberry Pi та встановимо систему контролю версій Git. Для того щоб отримати список усіх нових пактів, які можна встановити, виконаймо:

```
> sudo apt-get upgrade
```

По завершенню завантаження зробимо безпосередньо виконання оновлення пакетів, наступною командою:

```
> sudo apt-get upgrade
```

Після завантаження оновлення пакетів для операційної системи встановимо утиліту контролю версій Git командою:

```
> sudo apt install git
```

По покінченню завантаження та встановлення Git проведемо основну конфігурацію встановленої утиліти необхідну для її роботи за допомогою команд:

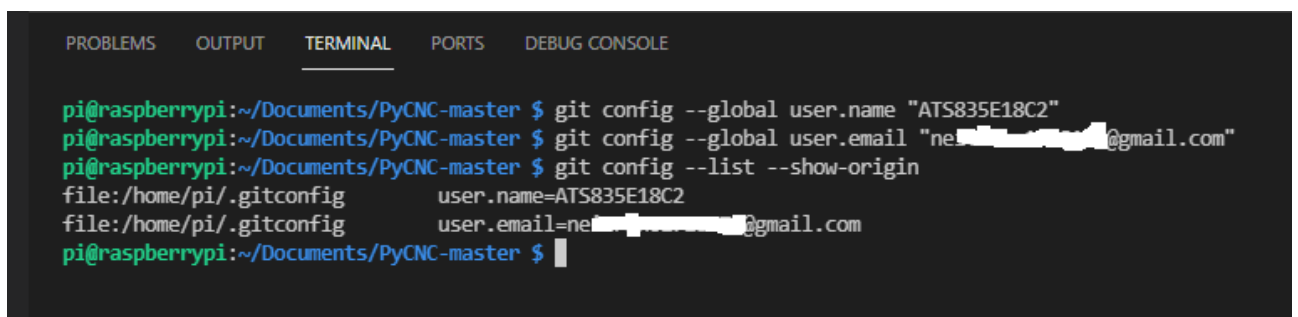
```
> git config --global user.name "name"
```

```
> git config --global user.email "email"
```

Щоб перевірити правильність проведених маніпуляцій введіть команду:

```
> git config --list --show-origin
```

повинена бути відповідь як показано на рисунку 3.4.



```

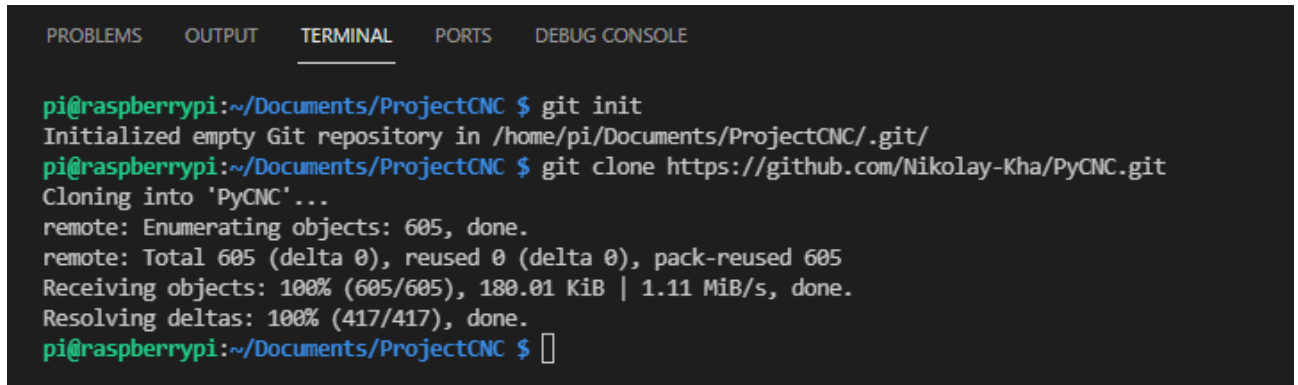
PROBLEMS  OUTPUT  TERMINAL  PORTS  DEBUG CONSOLE

pi@raspberrypi:~/Documents/PyCNC-master $ git config --global user.name "ATS835E18C2"
pi@raspberrypi:~/Documents/PyCNC-master $ git config --global user.email "ne[REDACTED]@gmail.com"
pi@raspberrypi:~/Documents/PyCNC-master $ git config --list --show-origin
file:/home/pi/.gitconfig      user.name=ATS835E18C2
file:/home/pi/.gitconfig      user.email=ne[REDACTED]@gmail.com
pi@raspberrypi:~/Documents/PyCNC-master $

```

Рисунок 3.4 – Перевірка загальної конфігурації Git

Після цього переходимо до потрібного каталогу заходимо каталогу й ініціалізуємо Git, та приступаємо к клонуванню за приведеним просиланням: <https://github.com/Nikolay-Kha/PyCNC.git>.

A screenshot of a terminal window with a dark background. At the top, there are tabs labeled 'PROBLEMS', 'OUTPUT', 'TERMINAL' (which is selected), 'PORTS', and 'DEBUG CONSOLE'. The terminal shows the following commands and output:

```
pi@raspberrypi:~/Documents/ProjectCNC $ git init
Initialized empty Git repository in /home/pi/Documents/ProjectCNC/.git/
pi@raspberrypi:~/Documents/ProjectCNC $ git clone https://github.com/Nikolay-Kha/PyCNC.git
Cloning into 'PyCNC'...
remote: Enumerating objects: 605, done.
remote: Total 605 (delta 0), reused 0 (delta 0), pack-reused 605
Receiving objects: 100% (605/605), 180.01 KiB | 1.11 MiB/s, done.
Resolving deltas: 100% (417/417), done.
pi@raspberrypi:~/Documents/ProjectCNC $
```

Рисунок 3.5 – результат клонування репозиторію

Після цього переходимо до `.cnc/config.py` рисунок 3.6 та налаштовуємо потрібну нам конфігурацію DSP та порти Raspberry Pi GPIO які будуть відповідати вихідному керуючому каналу, варіацію та можливості, програмного забезпечення, що наведено у посиланні вище з приведенного проекту автор розробив спосіб уніфікованого доступу до всіх приладів систем за допомогою можливостей Python та C++.

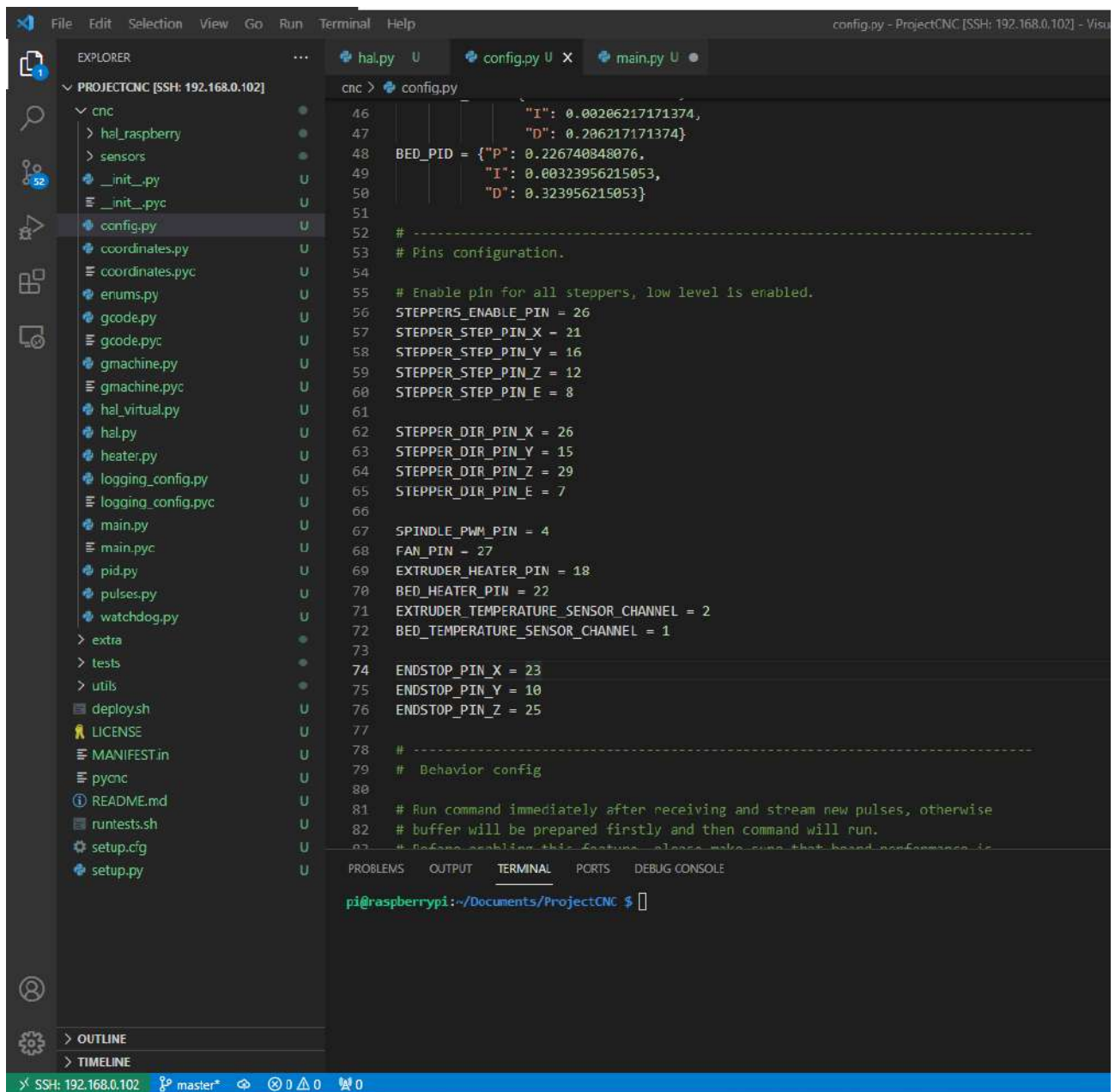


Рисунок 3.6 – Процес конфігурації прикладної програми PyCNC

Після налаштування конфігурації можна запустити програму через корень `./руснс`. Вона запуститься в інтерактивному режимі терміналу, таким чином надається можливість протестувати команди G-коду які можна вводити вручну, для того щоб запустити файл з G-кодом команд треба після назви програми ввести назву файлу G-коду наприклад: `./руснс filename`. Не виключена можливість

встановлення програми за допомогою інструментів Puthon командою: `sudo pip install`. Результат успішного запуску програми наведено на рисунок 3.7.

```

cnc > config.py
48 BED_PID = { "P": 0.226/408480/0,
49             "I": 0.00323956215053,
50             "D": 0.323956215053}
51
52 # -----
53 # Pins configuration.
54
55 # Enable pin for all steppers, low level is enabled.
56 STEPPERS_ENABLE_PIN = 26
57 STEPPER_STEP_PIN_X = 21
58 STEPPER_STEP_PIN_Y = 16
59 STEPPER_STEP_PIN_Z = 12
60 STEPPER_STEP_PIN_E = 8
61
62 STEPPER_DIR_PIN_X = 26
63 STEPPER_DIR_PIN_Y = 15
64 STEPPER_DIR_PIN_Z = 29
65 STEPPER_DIR_PIN_E = 7
66
67 SPINDLE_PWM_PIN = 4
68 FAN_PIN = 27
69 EXTRUDER_HEATER_PIN = 18
70 BED_HEATER_PIN = 22
71 EXTRUDER_TEMPERATURE_SENSOR_CHANNEL = 2
72 BED_TEMPERATURE_SENSOR_CHANNEL = 1
73
74 ENDSTOP_PIN_X = 23
75 ENDSTOP_PIN_Y = 10
76 ENDSTOP_PIN_Z = 25
77
78 # -----
79 # Behavior config
80
81 # Run command immediately after receiving and stream new pulses, otherwise
82 # buffer will be prepared firstly and then command will run
83
PROBLEMS OUTPUT TERMINAL PORTS DEBUG CONSOLE
pi@raspberrypi:~/Documents/ProjectCNC $ sudo pypy pycnc
***** Welcome to PyCNC! *****
> g1x20y20
Ok
> g2i10g10
Ok
>
Exiting...
pi@raspberrypi:~/Documents/ProjectCNC $

```

Рис 3.7 – результат успішного запуску програми PyCNC

Для взаємодії з мікроконтролером додано до файлу `main.py` програмами PyCNC свою програму `mSpiAvr.py` це процес який не вмішується основну роботу програми PyCNC й запускається як окремий процес. Основна задача цього процесу

це взаємодія з мікроконтролером через SPI для передачі результатів перетворення каналу ADC, та подальшої обробки цього каналу, реалізація реагування програми на отримані результати з мікроконтролеру була опущена, так як це не відповідає технічному завданню, і може бути розроблено окремо. Основна ціль цього процесу це реалізувати комунікацію між блоком розв'язки та блоком CPU що створює додаткове чисельне навантаження на мікроконтролер, яке потрібно урахувати під час тестування роботи блоку розв'язки.

3.2 Розробка структурної схеми пристрою

Структурна схема пристрою складається з наступних блоків:

- плата CPU;
- плата розв'язки;
- пристрій виводу;
- клавіатура.

Плата CPU виконує зчитування керуючого коду програми з носію, або з внутрішньої пам'яті, звертаючи увагу на конфігурацію системи, яка занесена до конфігурації що представляє специфіку станку, генерує керуючі імпульси які передаються блоку розв'язки.

Блок плати розв'язку не є звичайним повторювачем або підсилювачем сигналу, він постійно в реальному часі перевіряє стан датчиків, та реагує відповідною дією на отримані данні.

Драйвер крокового двигуна представляє із себе готовий високотехнологічний пристрій який приймає сигнали від блоку розв'язки, на основі яких керує подачею струму та напруги до обмоток крокового двигуна. Зазвичай драйвер керує лише одним двигуном, тому для станку у якого три осі, потрібно три таких пристрою як правило, якщо вони від одного виробника, то в них буде однаковий інтерфейс

керування, по цій причині, для спрощення, блок драйверів у структурній схемі (рисунку 3.8) позначений у множині.

Блок пристрою виводу, дисплей який призначений для візуального відображення інформації що поступає з блоку плати CPU, обмін інформацією відбувається завдяки інтерфейсу HDMI.

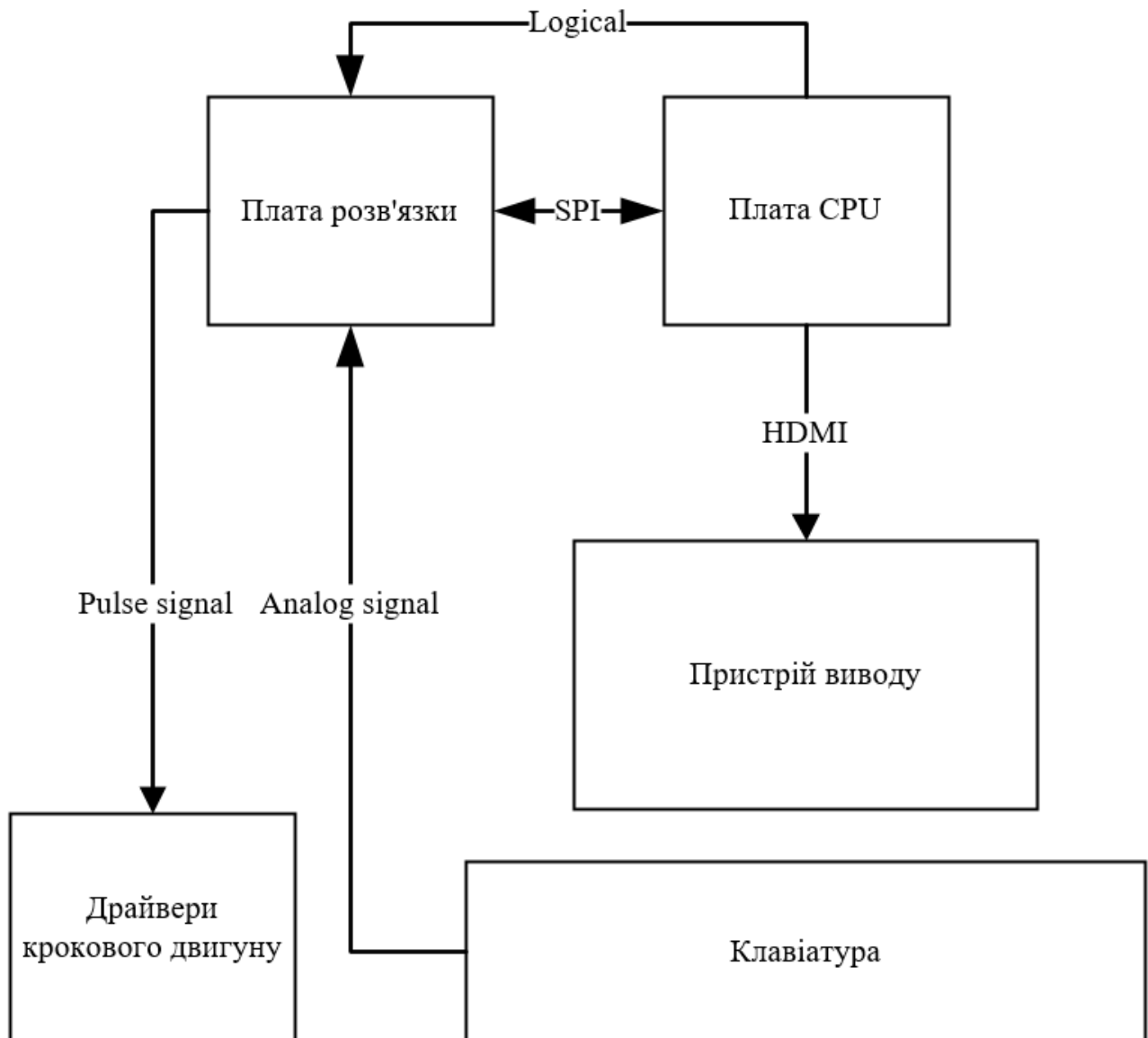


Рисунок 3.8 – Структурна схема пристрою

Блок клавіатури передає комбінацію рівня напруги до блоку плати розв'язки, які відповідають натисненим кнопкам та подіям на виконання команд.

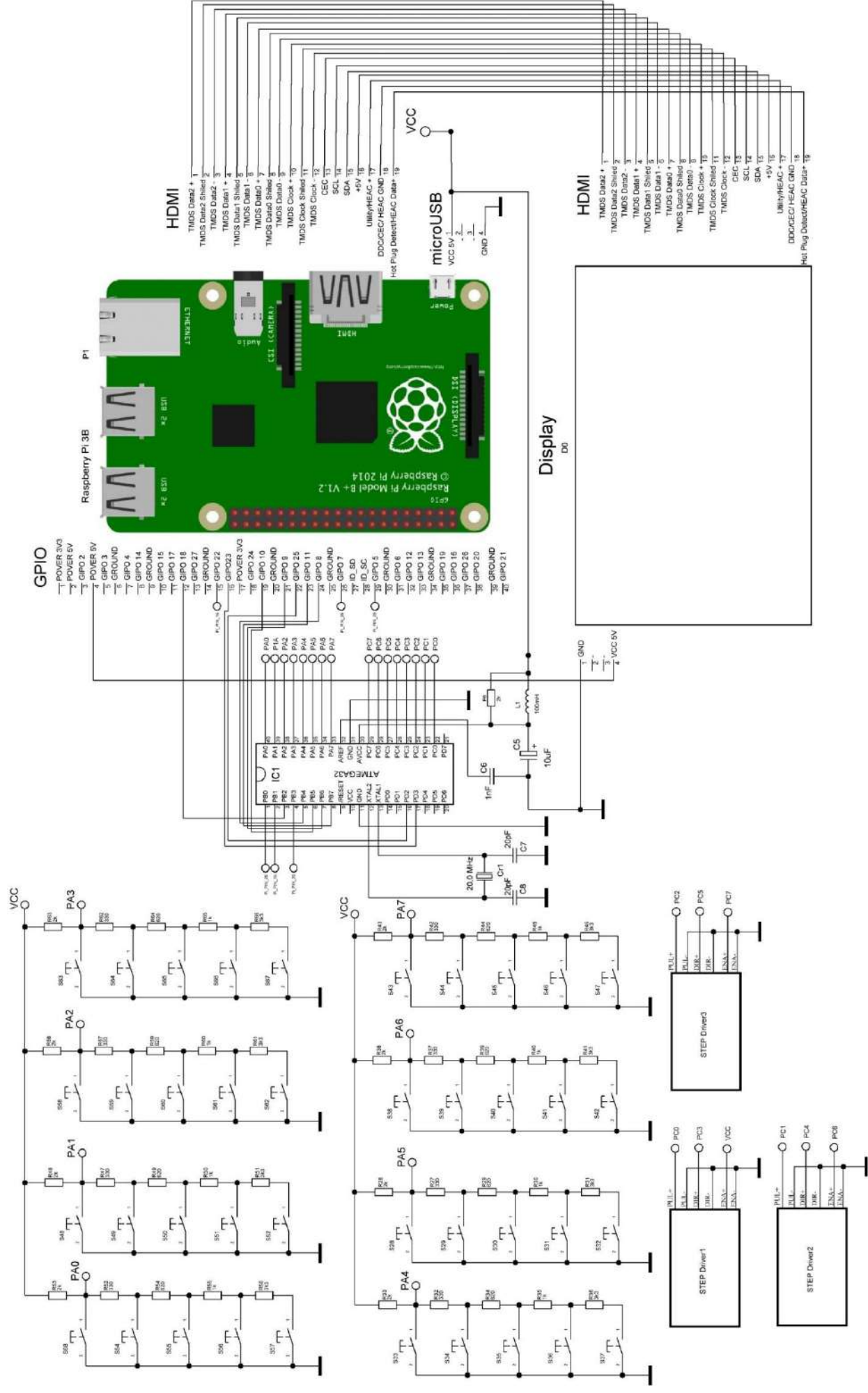
3.3 Розробка апаратної частини пристрою

Блок основної плати CPU це мікрокомп'ютер raspberry Pi 3B який зв'язаний SPI шиною з блоком розв'язки та HDMI інтерфейсом з блоком виводу.

Блок розв'язки побудовано на мікроконтролері AVR ATmega32. Кварцовий резонатор Cr1 задає максимальну частоту кристалу, що оговорюється специфікацією цього мікроконтролеру. Конденсатори C8, C7 забезпечують потрібне навантаження коливального контуру у який входить кварц це забезпечує стійкий старт коливань та передбачувану гармонію у продовж усієї роботи.

Блок пристрою виводу представляє із себе це семидюймовий IPS дисплей з'єднаний з блоком плати CPU і має роз'єм microUSB на який подається живлення 5В що потрібно для підсвічування дисплею.

Блок клавіатури побудований на послідовно з'єднаних резисторах та ключів між ними, які реалізують дільник напруги.



3.4 Програмування мікроконтролеру AVR ATmega32

При включенні живлення пристрою мікроконтролер ATmega32 починає працювати за спеціально розробленою програмою, алгоритм роботи якої наведено на рисунку 3.11.

Робота програми починається зі здійснення ініціалізації внутрішніх модулів контролера: таймерів Timer/Counter0, Timer/Counter1, Timer/Counter2, портів вводу виводу, аналого-цифрового перетворювачу (ADC), інтерфейсу SPI, та модулю переривань.

По завершенню ініціалізації починається нескінченний цикл процесу у якому відбувається перехід на функцію/процес за значенням яке має глобальна змінна processN, паралельно працює Timer/Counter2 у режимі таймеру(normal), за допомогою якого визначається час виконання процесу, якщо процес дозволяє глобальні переривання, відбувається обробка вектору TIMER2 OVF що має найвищий пріоритет з представленої периферії таймерів даного мікроконтролеру, основна задача якого виділяти час на роботу процесу, наступний процес виберетися за ситуацією коли це потрібно.

Процес комунікації Raspberry Pi через SPI інтерфейс не має високого пріоритету і не використовує SPI переривання, однак забороняє глобальні переривання на один цикли передачі байту, основна задача процесу приймати та виправляти данні до плати Raspberry Pi.

Процес обробки клавіатури виконує обробку натиснень кнопок, процес може бути перерваний лише тоді коли аналого-цифровий перетворювач завершить обробку каналу, результати натиснень відправляються до плати CPU за інтерфейсом SPI.

Процес повторювача має найбільший пріоритет, який працює у спеціальному режимі основна задача правильно забороняти переривання TIMER2 OVF, для максимальної готовності обробки переривань INT0, INT1, INT2, на вхідні імпульси

плати CPU. Припиненням процесу може послугувати відсутність сингалів з енкодерів, які обробляє AVR у векторах INT0, INT1, INT2.

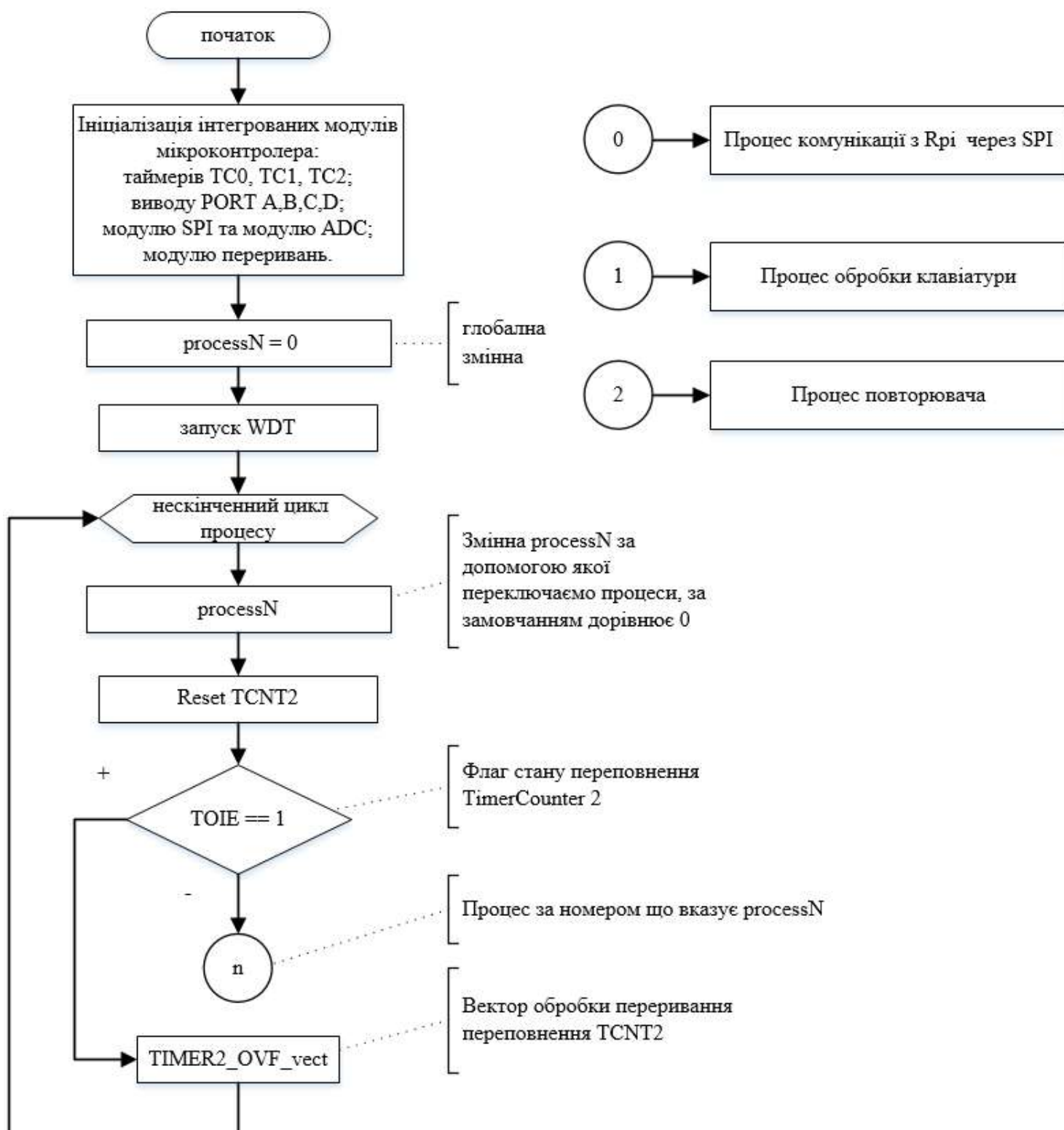


Рисунок 3.11 – Алгоритм боти основного циклу програми

За представленим алгоритмом у середовищі AtmelStudio 7.0.129 мовою C написано програму, за якою працює мікроконтролер. Цей програмний продукт було обрано з наступних причин:

- інтуїтивно зрозумілий інтерфейс з доступними інструкціями;
- вбудований компілятор мови C із безкоштовним використанням без обмежень на розмір пам'яті програми;
- наявність утіли що дозволяє відлагоджувати програму, емулюючи кристал.
- безліч бібліотек для роботи з різноманітними структурами AVR.

Текст програми наведено у додатку А.

3.5 Моделювання роботи пристрою

В цьому розділ описується тестування як блоку плати CPU та і блоку розв'язку який побудований на базі мікроконтролеру AVR ATmega32.

Тестування блоку розв'язки проводилось за допомогою програмного продукту Proteus Design Suite який дозволяє перевірити коректність роботи програмного коду мікроконтролеру. Для цього у програмі Proteus була побудована електрична схема максимально наближена до схеми, що на рисунку 3.10. Це потрібно для того щоб у повній мірі змодельовати чисельну загрузку на ядро AVR.

Тестування плати CPU проводилось за допомогою емулювання основного функціоналу станку на три осі. Даний підхід надає змогу дослідити коректну роботу програмного забезпечення PyCNC, а саме: правильність інтерпретації команд G-коду та коректність конфігураційного налаштування керування шаговими двигунами станку за допомогою яких реалізується зонне 3Д фрезерування.

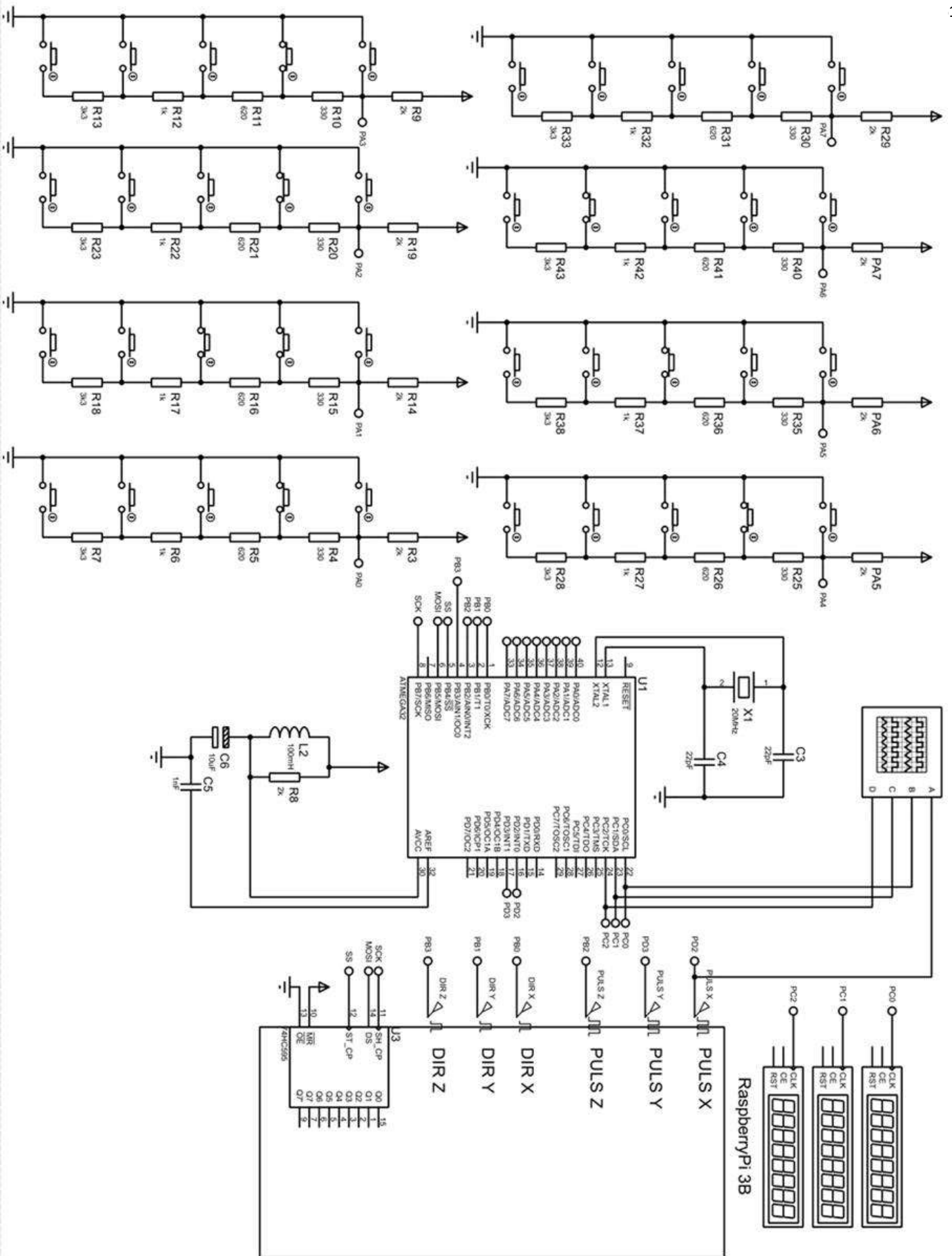
Для реалізації цього методу потрібен програмно-апаратний комплекс який надасть змогу передавати данні з розробленого пристрою (імпульси до драйверів крокових двигунів) до віртуального верстату, та надавати телеметрію стану його

пристроїв, позицію тощо й на додаток візуалізувати позиціонування конструкцій станку у віртуальному просторі. Досягнути цього можна за допомогою умовно безкоштовного програмного продукту Unity та плати Arduino Pro Micro, завдяки яким можна розробити програму емуляції станку, що надасть змогу зрозуміти коректність роботи пристрою.

3.5.1 Тестування блоку розв'язки на базі ATmega32

Методами програмного продукту Proteus Design Suite, було побудовано електричну схему рисунок 3.12. Оскільки програмний продукт не підтримує емуляцію пристрою Raspberry Pi 3B було вирішено створити його мізерну на функціонал модель за допомогою інструменту генерації імпульсів (Generate mode - > PULS) та мікросхеми 74PC595 яка є восьмирозрядним зсувним регістром з послідовним введенням – SPI, таким чином ми отримуємо потрібну нам поведінку, передбачувані вхідних дані які буде оброблювати мікроконтролер.

По результатам тесту було виявлено, що максимальна частота яку здатен видати мікроконтролер до драйверу крокового двигуна, при паралельній обробці трьох процесів що реалізує прошивка це 15kHz рисунок 3.13



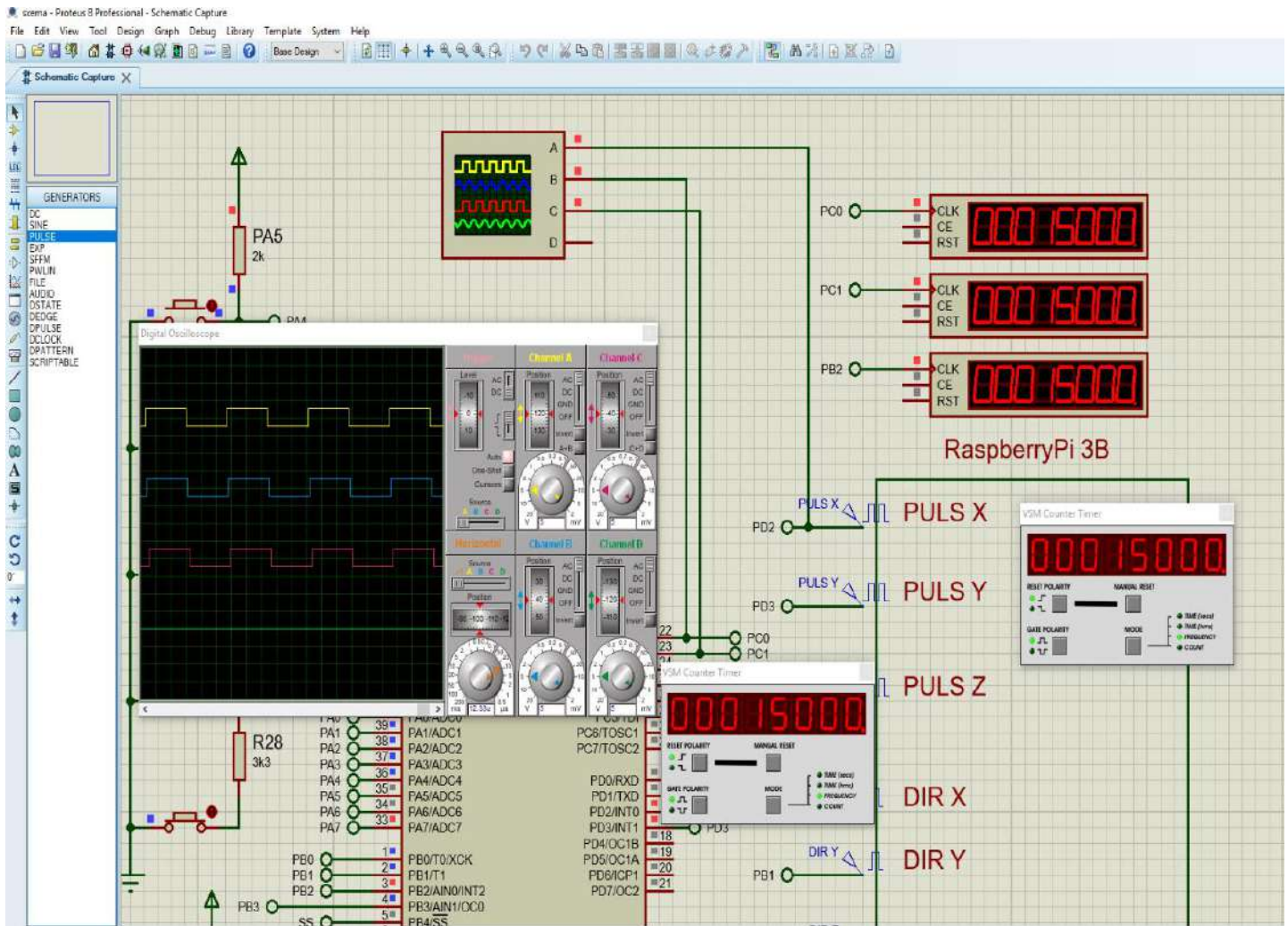


Рисунок 3.13 – результат тестування блоку розв’язки

3.5.2 Розробка віртуальної моделі станку

Оскільки Unity у собі не має засобів розробки для 3Д моделювання, скористаймося безкоштовним програмним забезпеченням Blender за посилання <https://www.blender.org/download/Blender2.92/blender-2.92.0-windows64.msi>. По закінченню встановлення програми, запускаємо 3Д редактор, додатково налаштовувати програмний продукт не потрібно, одиниця вимірювання в міліметрах (mm), стоїть за замовчуванням. Далі було знайдено у мережі інтернет креслення моделі верстату на 3 осі. За основу було взято модель креслення станку з приведенного джерела рисунок креслення приведений у рисунку 3.12

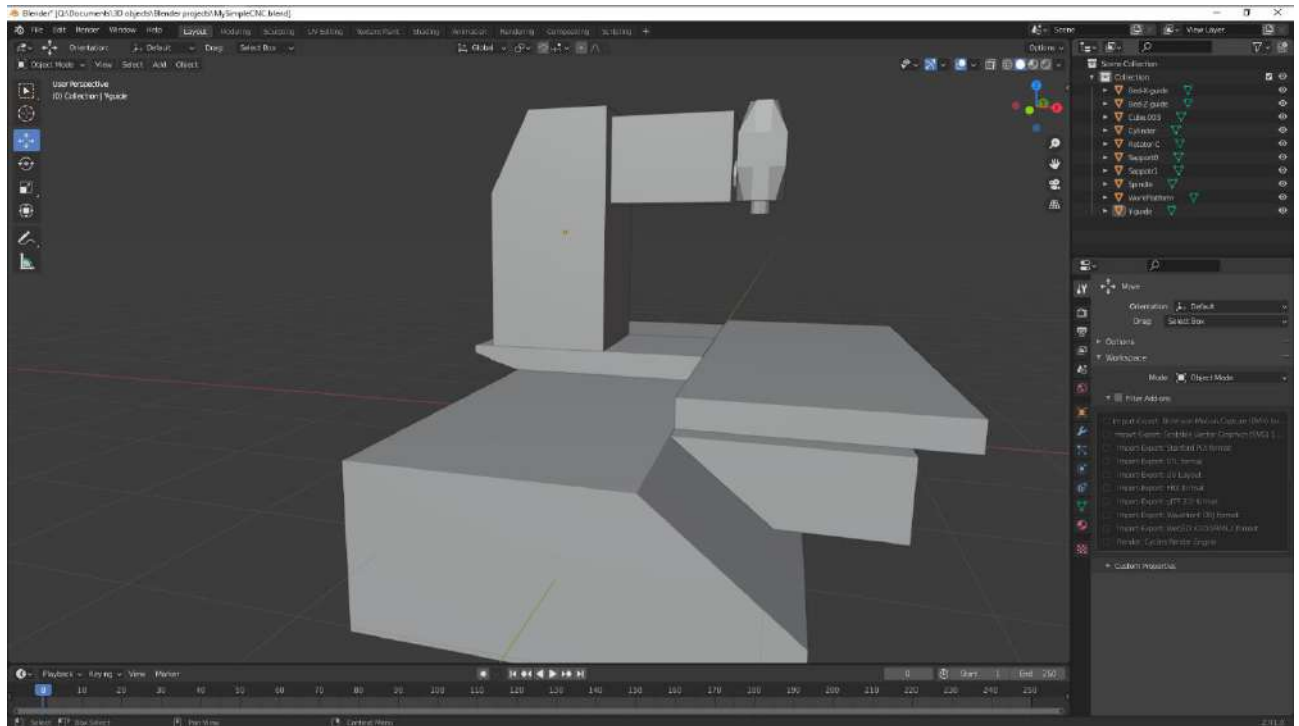


Рисунок 3.13 – результат моделювання станку

3.5.3 Створення програми емуляції станку засобами Unity

Програмний продукт Unity який можна завантажити за допомогою лаунчеру UnityHub за посиланням

<https://public-cdn.cloud.unity3d.com/hub/prod/UnityHubSetup.exe> являється міжплатформеним середовищем розробки комп'ютерних ігор, котрий надає безліч різних бібліотек для відносно швидкого процесу створення програм. Що й послугувало основним поштовхом для реалізації базової емуляції станку.

Рішенням питання підключення й передачі даних з пристроєм, який є темою даного диплому, до віртуального верстату впаде на Arduino Pro micro. Arduino Pro micro це пристрій з сімейства Arduino яке було оглянуте у розділі 2.1.2, головною причиною вибору цієї плати послугувало те, що вона може емулювати так звані HID пристрої шляхом програмного налаштування за допомогою вже готових бібліотек.

3.5.3.1 Налаштування плати Arduino.

Для того щоб запрограмувати плату Arduino Pro micro для роботи у режимі HID пристрою, треба завантажити єдину і безкоштовну спеціально розроблену середу розробки Arduino IDE за посиланням <https://downloads.arduino.cc/arduino-1.8.15-windows.zip>, та бібліотеку <https://github.com/NicoHood/HID>, яку після завантаження треба буде перемістити\розпакувати у каталог Arduino IDE за шляхом ...\\arduino-1.8.15\\libraries. Після вводу скетчу, вибираємо плату «Arduino Micro» та вибираємо порт на якому підключена потрібна нам плата. Рисунок 3.16.

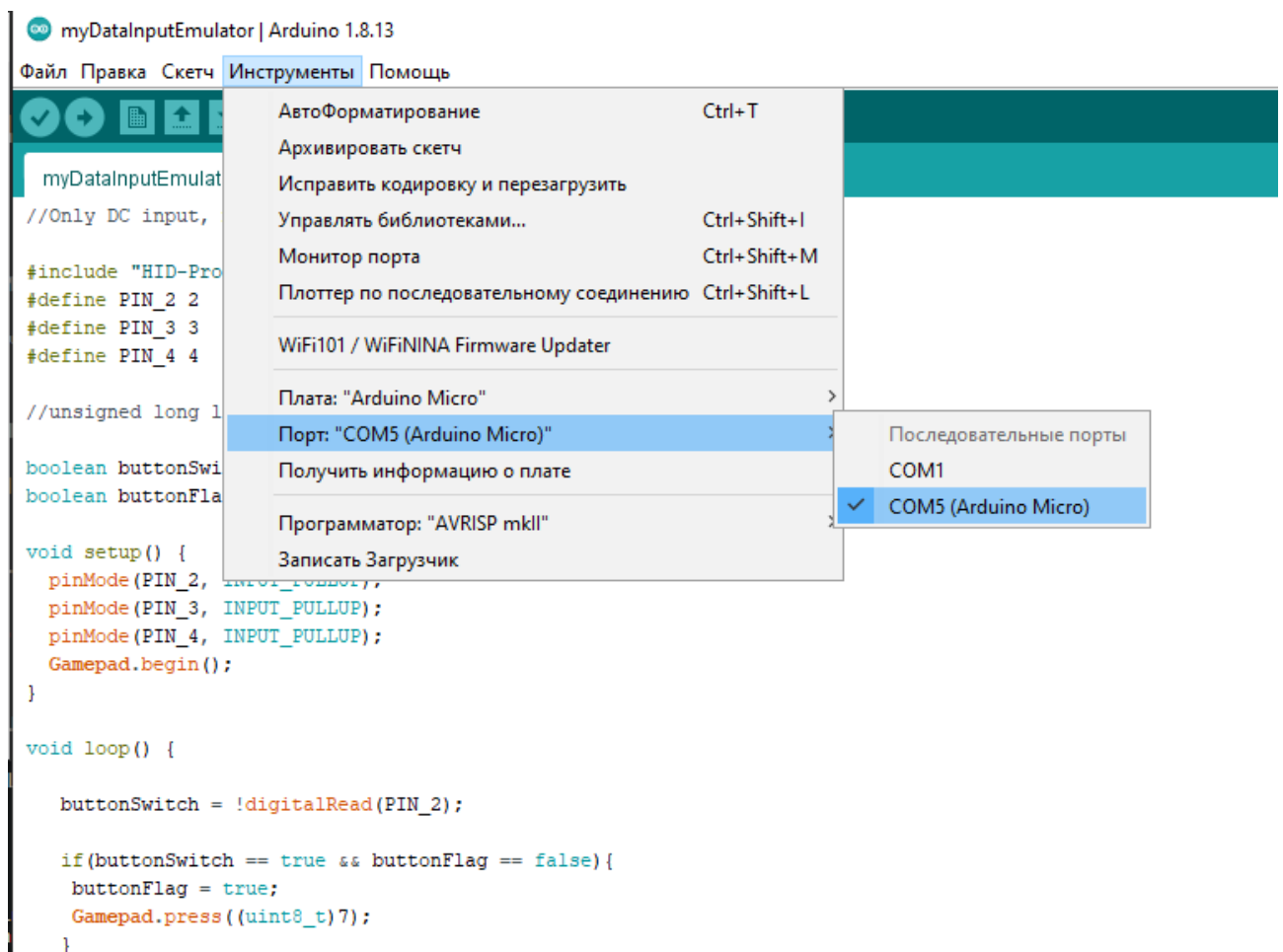


Рис 3.16 – вибір потрібної плати у IDE Arduino

По закінченню налаштування та вдалої загрузки скетчу до плати, переконаємося в тому, що плата дійсно працює так як нам треба, для цього потрібно зробити такі дії: Панель управління => Оборудование и звук =>

Устройства и принтеры у списку пристроїв повинен з'явитися Arduino micro з піктограмою ігрового контролеру. Перевіривши його властивості побачимо слідує результат рисунок 3.17.

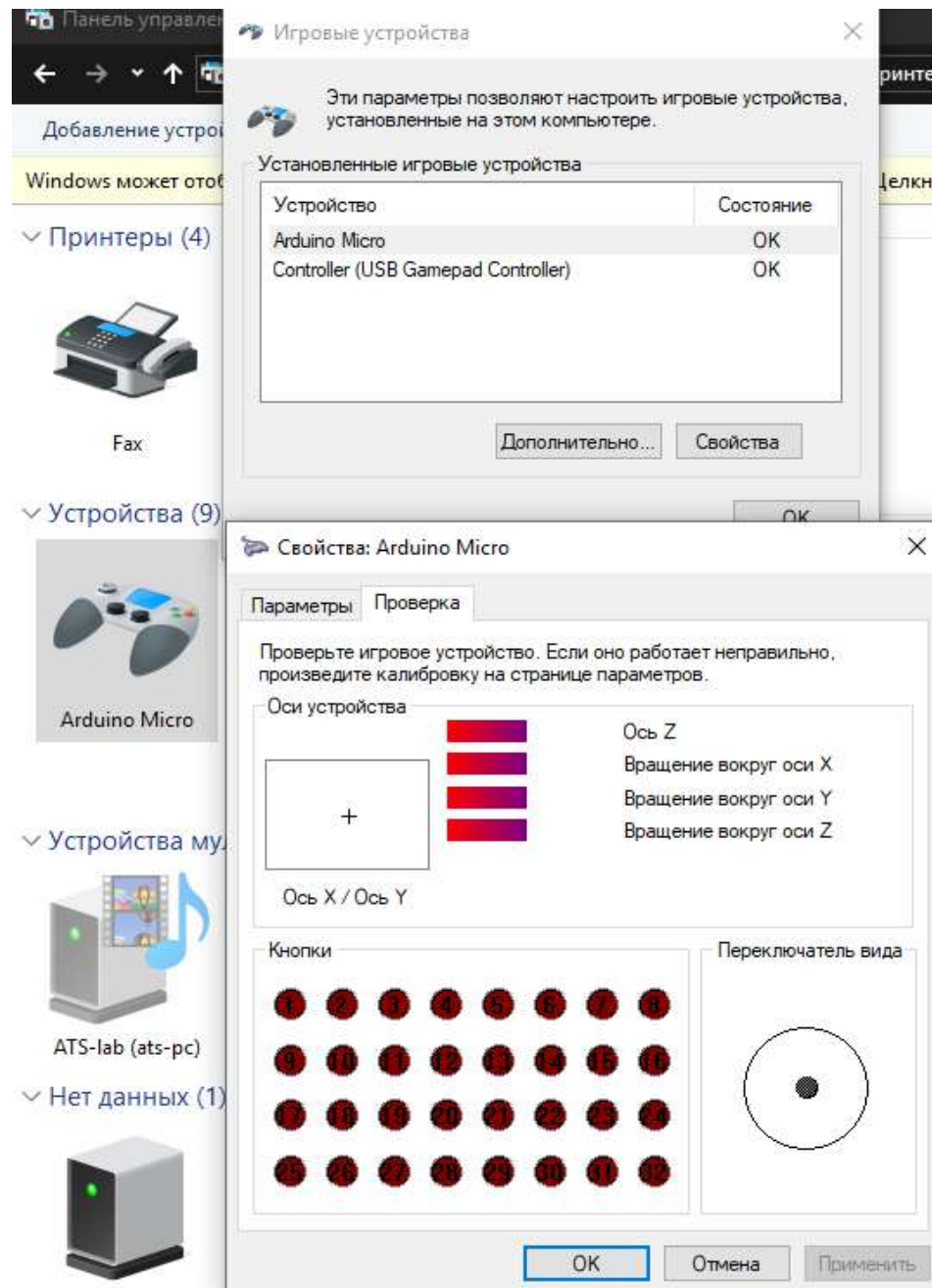


Рисунок 3.17 – результат перевірки властивостей Arduino micro

3.5.3.2 Налаштування Unity

Для того щоб середа розробки Unity побачила пристрій вводу на базі Arduino Pro micro для цього потрібно встановити додатковий плагін input System, який надає змогу бачити нестандартні пристрої вводу.

Створивши проект і запустивши програму Unity натискаємо на вкладку Windows та обираємо Package manager. У вікні Package manager у полі вводу набираємо «input» з ліва з'явиться остання версія плагіну рисунок 3.18, натискаємо кнопку install. По скінченню встановлення, у спливаючому вікні натискаємо «Yes», Unity перезавантажиться.

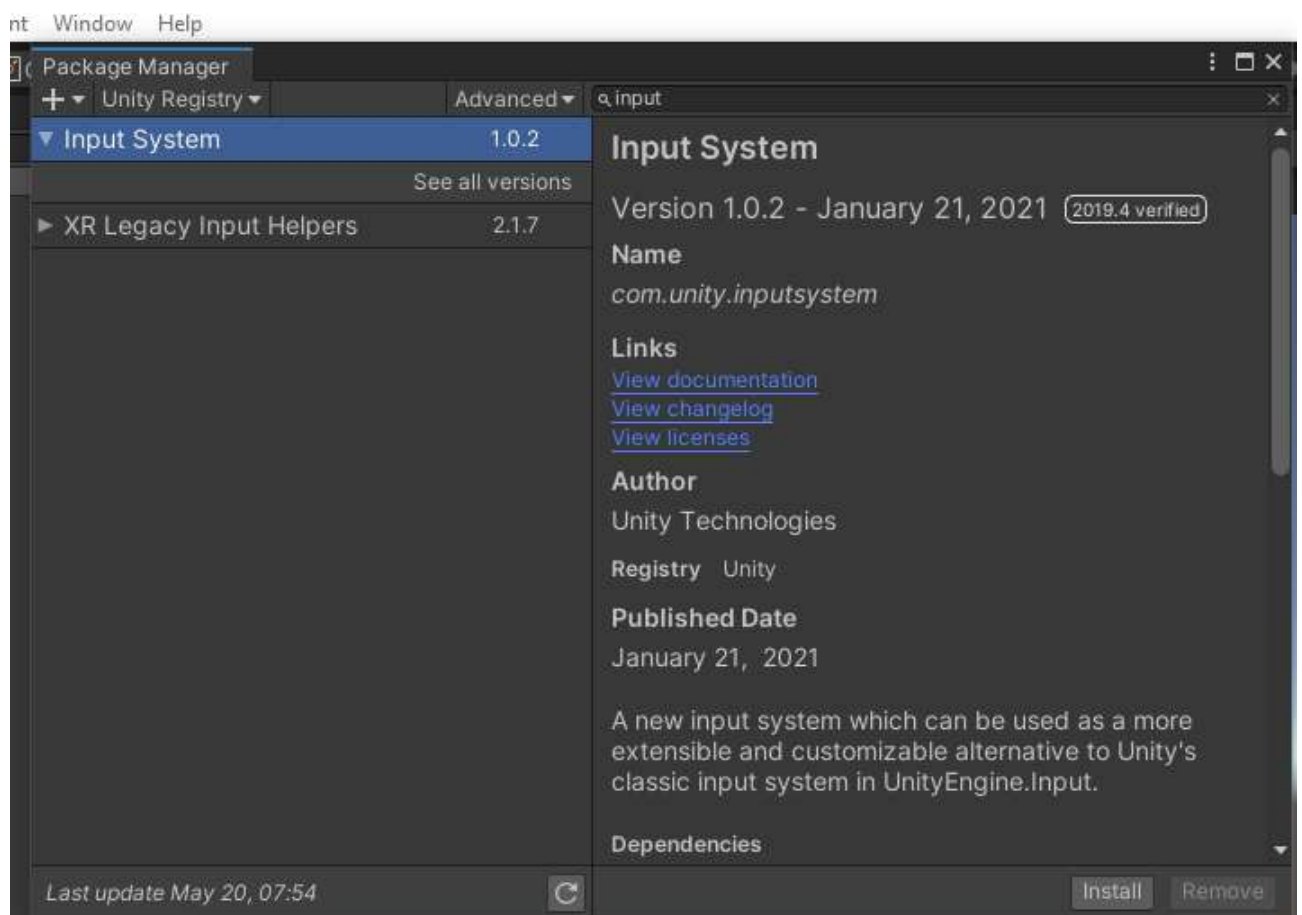


Рисунок 3.18 – Встановлення плагіну Input System

Далі створюємо об'єкт Input Actions у полі Assets ПКМ => Create => Input Actions, подвійним натиском по об'єкту, відкриваємо вікно налаштування Input Actions, та створюємо Actions Maps натискаючи на кнопку «+» у полі Action

називаємо та назначаємо усі налаштовані від Arduino кнопки у режимі зчитування Button. Після зроблених налаштувань у вікні inspector прописуємо ім'я для майбутнього фалу що генерується та натискаємо кнопку Apply.

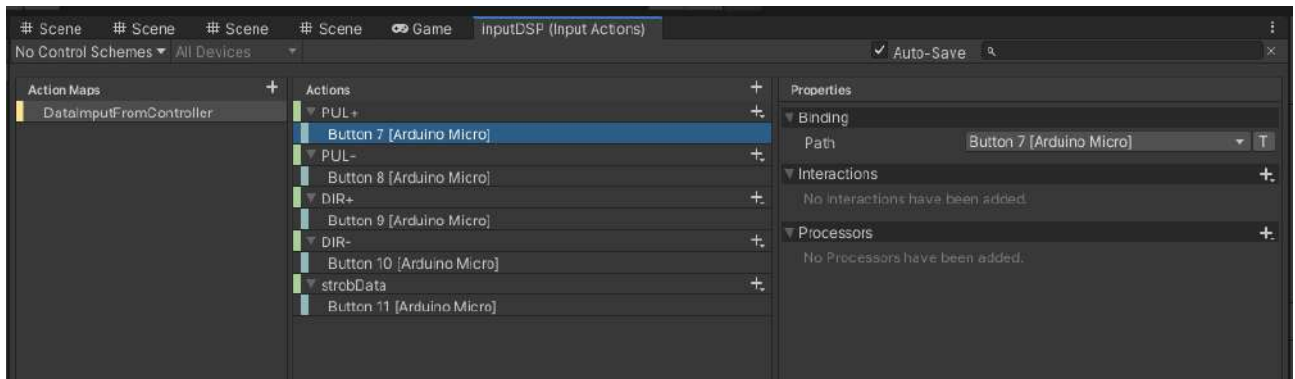


Рисунок 3.19 – Результат роботи налаштування Input System

3.5.4 Розробка програми емуляції станку за допомогою Unity

Цей розділ описує процес розробки програми емуляції базового функціоналу станку, а саме емуляцію драйверу крокового двигуна та зміщення конструкцій верстату за направляючим трьох осей. Емуляція інвертору шпинделю та процес наглядного фрезерування тіла була опущена. Основна суть даного підходу впевнитись у правильності виконання пристроєм команд G-коду на основі яких генеруються імпульси керування шаговими двигунами.

У поле Assets програми Unity загрузимо розроблену у розділі 3.6.1 модель станку, на налаштуймо ієрархію залежності позиціювання об'єктів станку шляхом їх вкладення як це показано на рисунку 3.20. Таким чином наприклад переміщуючи об'єкт CNC, об'єкти які вкладені до нього унаслідують його позиціювання – перемінюються роз з ним, як єдиний об'єкт.

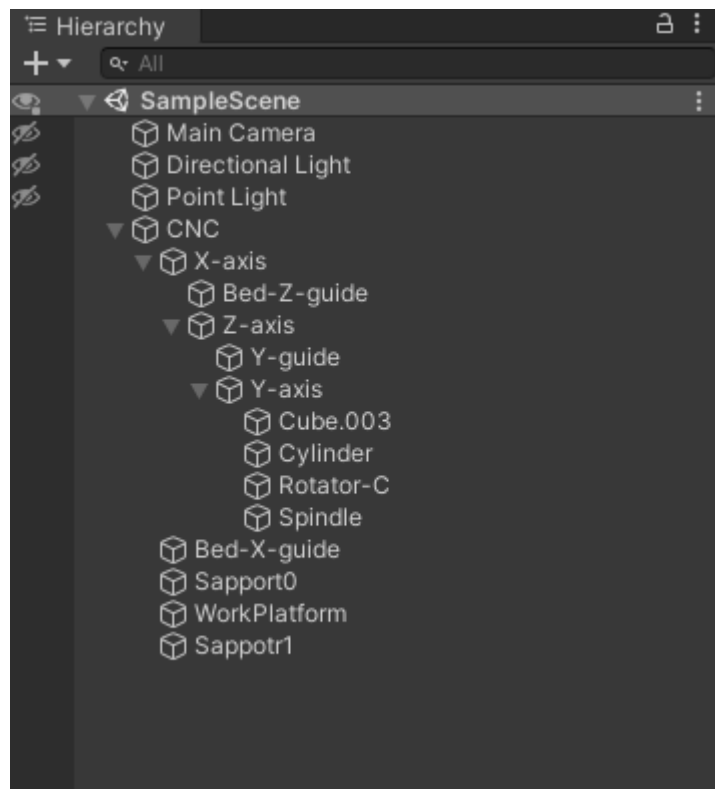


Рисунок 3.20 – налаштування залежності об’єктів моделей

Наступним кроком програмуємо логіку поведінки об’єктів у створеному проекті в Unity на мові програмування C#. Оброблювати вхідні данні з розробленого пристрою управління кроковими двигунами буде простий емулятору драйверу крокового двигуна зі стандартним інтерфейсом: PUL+, PUL-, DIR+, DIR- й передбачувано реагуватиме на вхідні данні.

3.5.5 Тестування PyCNC

Після завершення розробки програми емуляції станку, до плати Arduino мікро був підключений мікрокомп’ютер Raspberry Pi 3B із працюючим програмним забезпеченням PyCNC.

Далі у програмному забезпеченні PyCNC було запущено тестову програму G-коду, за якою на налаштованих у конфігурації програми виводах мікрокомп’ютеру Raspberry Pi генерувалися керуючі імпульси які приймав Arduino мікро та перенаправляв їх до віртуального драйверу емулятору станку.

Результат виконання дав змогу зрозуміти, що налаштування у конфігураційному файлі програми PyCNC відповідають очікуваній поведінці станку на прийняті данні з мікрокомп'ютеру.

ВИСНОВКИ

Концепція системи керування кроковими двигунами, передбачає підхід побудови на основі об'єднання різного комплексу інноваційних технічних рішень, що в значній степені змінюють ефективність функціонування системи керування, шляхом покращення алгоритмів автоматизації та апаратної складової, керування різних модулів, драйверів, датчиків, розподілених силових вузлів, на основі певних вхідних даних.

Ідея використання систем керування кроковими двигунами не нова і тому станом на 2021 рік існує велика варіація приладів для побудови такої системи. Оскільки система керування кроковими двигунами є частиною комплексу системи станку, тому компоненти з яких складатися система керування окрім функціонального призначення, відрізняються технологіями комунікації між приладами та модулями. Наприклад датчики здебільшого використовують SPI, I²C UART.

Існуючі апаратні і програмні рішення орієнтовані на використання лише певних приладів від компанії виробника, що суттєво зменшає варіативність затраченого часу але позбавляє можливості використовувати інші прилади для модернізування устаткування до специфічних робіт, або заміни компонентів на найбільш досконалих.

Під час виконання дипломної роботи була розроблена система керування кроковими двигунами, яка спроектована таким чином, що інтеграція в систему нового компонента або перехід до іншого устаткування потребує лише відповідного запису в налаштування та корекції Python скрипту, Python завдяки своїй інтерпретативності дозволяє додавати нові правила взаємодії з мінімальними зусиллями.

В результаті отримано продукт, який поєднує переваги інших подібних продуктів та вирішує їх недоліки, має низьку вартість та високий потенціал для майбутнього розвитку.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Станки с ЧПУ Устройство, программирование, ИНструментальное Обеспечение и Оснастка - Издательство ФЛИНТА, 2014 р.
2. Современный Станок с ЧПУ и CAD/CAM-система - ДМК, 2015.
3. Основы программирования станков с ЧПУ, Екатиринбург 2015 р.
4. Build Your Own CNC machine(Technology in action) – Paperback 2009.
5. Разработка управляющих програм для оборудования с ЧПУ – ЛАНЬ 2017
6. The CNC Cookbook: An Introduction to the Creation and Operation of Computer Controlled Mills, Router Tables, Lathes, and More - Scited Publications 2009.
7. AVR Microcontroller and Embedded Systems: Pearson New International Edition: Using Assembly and C Muhammad Ali Mazidi, - DeVry University 2014
8. Digital Design and Computer Architecture 2nd Edition, David Harri, Sarah L. - Harris Morgan Kaufman English Edition 2013
9. Embedded C Programming and the Atmel AVR 1st Edition by Richard H. Barnett , Sarah Cox , Larry O’Cull - Delmar Cengage Learning 2002
10. Programming the Raspberry Pi: Getting Started with Python by Simon Monk- McGraw-Hill Education Tab 2012
11. Raspberry Pi Stepper Motor Tutorial: вебсайт. Режим доступу: <https://www.rototron.info/raspberry-pi-stepper-motor-tutorial/>
12. SPI with Raspberry Pi: вебсайт. Режим доступу: <https://core-electronics.com.au/tutorials/i-spi-with-raspberry-pi.html>
13. введение в частоту приемистости шагового двигателя: вебсайт. Режим дуступу: <https://avtika.ru/chto-takoe-chastota-priemistosti-shagovogo-dvigatelya/>

14. ATmega32/L Datasheet: вебсайт. Режим
доступа:
<https://ww1.microchip.com/downloads/en/DeviceDoc/doc2503.pdf>

ДОДАТОК А

ПРОГРАМА ДЛЯ АТМЕГА32

Лістинг А – main.c

```

#define F_CPU 2000000UL
#include <avr/io.h>
#include <avr/interrupt.h>
#include <avr/pgmspace.h>

//#include <util/delay.h> //for debug
const uint8_t Temp[] PROGMEM = {0x0f, 0xff};

#define MOSI PB5
#define MISO PB6
#define SCK PB7
#define SS PB4

#define STEPER_PULS_X PC0
#define STEPER_PULS_Y PC1
#define STEPER_PULS_Z PC2

#define STEPER_DIR_X PC3
#define STEPER_DIR_Y PC4
#define STEPER_DIR_Z PC5

#define STEPER_ENA_X PC3
#define STEPER_ENA_Y PC4
#define STEPER_ENA_Z PD7

//bits responsible for the mode
#define USUALLY 0x01
#define REPITER 0x02

void CommutationWithRaspberry(void);
void keyboard(void);
void repiter(void);
void (*processAdres[3]) (void) = { CommutationWithRaspberry,
keyboard, repiter};

uint16_t adc_read(unsigned char channel);
void spi_Rpi(uint8_t data);
uint8_t uDataADC = 0;
uint8_t cannel = 0;

struct{
    uint8_t mod :1;

```

```

    uint8_t N : 2;
} Tread;

int main(void) {
    Tread.N = 0;

    WDTCSR |= (1<<WDTOE) | (1<<WDE);
    WDTCSR &=~(1<<WDE);

    MCUCR |= (1<<ISC00) | (1<<ISC10) | ((1<<ISC2));
    GICR  |= (1<<INT0) | (1<<INT1) | (1<<INT2);

    TCCR2 |= (1<<CS20);
    TIMSK |= (1<<TOIE2);

    ADMUX |= (1<<REFS0);
    ADCSRA|= (1<<ADEN);
    ADCSRA|= (1<<ADPS0) | (1<<ADPS2);

    ADCSRA|= (1<<ADEN); //разрешить работу АЦП
    ADCSRA|= (1<<ADPS0) | (1<<ADPS2); // делитель 32

    DDRA = 0xff; //input ADC

    DDRD |= (0<<PD2) | (0<<PD3);
    PORTD|= (0<<PD2) | (0<<PD3);

    DDRB|= (0<<PB0) | (0<<PB1) | (0<<PB2) | (0<<PB3);
    DDRB|= (0<<MOSI) | (0<<MISO) | (0<<SCK) | (0<<SS);
    PORTB|= (0<<PB0) | (0<<PB1) | (0<<PB2) | (0<<PB3);

    DDRC |= (1<<STEPPER_PULS_X) | (1<<STEPPER_PULS_Y) |
(1<<STEPPER_PULS_Z);
    DDRC |= (1<<STEPPER_DIR_X) | (1<<STEPPER_DIR_Y) |
(1<<STEPPER_DIR_Z);
    DDRC |= (1<<STEPPER_ENA_X) | (1<<STEPPER_ENA_Y);
    PORTC|= 0x00;
    DDRD|= (1<<STEPPER_ENA_Z);
    PORTD|= (0<<STEPPER_ENA_Z);

    asm("wdr");
    sei();

    while (1)
    {

    }
}

ISR(INT0_vect) {

```

```

        PORTC ^= (1<<STEPPER_PULS_X);
    }

    ISR(INT1_vect) {
        PORTC ^= (1<<STEPPER_PULS_Y);
    }

    ISR(INT2_vect) {
        PORTC ^= (1<<STEPPER_PULS_Z);
    }

    ISR(TIMER2_OVF_vect) {
        processAdres[Tread.N++]();
        if(Tread.mod != REPITER) { }
        asm("wdr");
    }

    ISR(TIMER0_OVF_vect) {
        PORTA ^= (1<<PA0);
        PORTA ^= (1<<PA0);
    }

    void CommutationWithRaspberry(void) {
        cli();
        spi_Rpi(uDataADC);
        if(uDataADC == REPITER) { Tread.mod |= REPITER; Tread.N =
2; }
        asm("wdr");

        sei();
    }

    void keyboard(void) {

        cli();
        uDataADC = adc_read(cannel);
        if(cannel == 8) {cannel = 0;}
        Tread.N = 0;
        asm("wdr");
        sei();
    }

    void repiter(void) {

        Tread.N = 0;
        sei();
    }

    uint16_t adc_read(unsigned char channel) {
        ADMUX&=0xf0;
        ADMUX|= channel;
        ADCSRA |= (1<<ADSC);
        while((ADCSRA & (1<<ADSC)));
    }

```

```
        Tread.N = 0;
        return ADC;
    }

void spi_Rpi(uint8_t data){
    PORTB &=~(1<<SS);

    SPDR = data;
    while(!(SPDR&(SPIF)));

    PORTB |= (1<<SS);
}
```

ДОДАТОК В

СКРИПТИ ДЛЯ СЕРВЕРНОЇ ЧАСТИНИ

Лістинг В – configure.py

```
# -----
# Hardware config.

# Maximum velocity for each axis in millimeter per minute.
MAX_VELOCITY_MM_PER_MIN_X = 24000
MAX_VELOCITY_MM_PER_MIN_Y = 12000
MAX_VELOCITY_MM_PER_MIN_Z = 600
MAX_VELOCITY_MM_PER_MIN_E = 1500
MIN_VELOCITY_MM_PER_MIN = 1
# Average velocity for endstop calibration procedure
CALIBRATION_VELOCITY_MM_PER_MIN = 300

# Stepper motors steps per millimeter for each axis.
STEPPER_PULSES_PER_MM_X = 100
STEPPER_PULSES_PER_MM_Y = 100
STEPPER_PULSES_PER_MM_Z = 400
STEPPER_PULSES_PER_MM_E = 150

# Invert axes direction, by default(False) high level means
increase of
# position. For inverted(True) axis, high level means decrease
of position.
STEPPER_INVERTED_X = True
STEPPER_INVERTED_Y = False
STEPPER_INVERTED_Z = False
STEPPER_INVERTED_E = True

# Invert zero end stops switches. By default(False) low level
on input pin
# means that axis in zero position. For inverted(True) end
stops, high level
# means zero position.
ENDSTOP_INVERTED_X = True
ENDSTOP_INVERTED_Y = True
ENDSTOP_INVERTED_Z = False # Auto leveler

# Workplace physical size.
TABLE_SIZE_X_MM = 200
TABLE_SIZE_Y_MM = 200
TABLE_SIZE_Z_MM = 220

# Mixed settings.
```

```

    STEPPER_PULSE_LENGTH_US = 2
    STEPPER_MAX_ACCELERATION_MM_PER_S2 = 3000 # for all axis, mm
per sec^2
    SPINDLE_MAX_RPM = 10000
    EXTRUDER_MAX_TEMPERATURE = 250
    BED_MAX_TEMPERATURE = 100
    MIN_TEMPERATURE = 40
    EXTRUDER_PID = {"P": 0.059161177519,
                    "I": 0.00206217171374,
                    "D": 0.206217171374}
    BED_PID = {"P": 0.226740848076,
              "I": 0.00323956215053,
              "D": 0.323956215053}

# -----
-----
# Pins configuration.

# Enable pin for all steppers, low level is enabled.
STEPPERS_ENABLE_PIN = 26
STEPPER_STEP_PIN_X = 21
STEPPER_STEP_PIN_Y = 16
STEPPER_STEP_PIN_Z = 12
STEPPER_STEP_PIN_E = 8

STEPPER_DIR_PIN_X = 20
STEPPER_DIR_PIN_Y = 19
STEPPER_DIR_PIN_Z = 13
STEPPER_DIR_PIN_E = 7

SPINDLE_PWM_PIN = 4
FAN_PIN = 27
EXTRUDER_HEATER_PIN = 18
BED_HEATER_PIN = 22
EXTRUDER_TEMPERATURE_SENSOR_CHANNEL = 2
BED_TEMPERATURE_SENSOR_CHANNEL = 1

ENDSTOP_PIN_X = 23
ENDSTOP_PIN_Y = 10
ENDSTOP_PIN_Z = 25

# -----
-----
# Behavior config

# Run command immediately after receiving and stream new
pulses, otherwise
# buffer will be prepared firstly and then command will run.
# Before enabling this feature, please make sure that board
performance is
# enough for streaming pulses(faster then real time).
INSTANT_RUN = True

```

```

    # If this parameter is False, error will be raised on command
    with velocity
    # more than maximum velocity specified here. If this parameter
    is True,
    # velocity would be decreased(proportional for all axes) to
    fit the maximum
    # velocity.
    AUTO_VELOCITY_ADJUSTMENT = True

    # Automatically turn on fan when extruder is heating, boolean
    value.
    AUTO_FAN_ON = True

```

Лістинг В – mSpiAvr.py

```

import RPi.GPIO as GPIO
import spidev

spi = spidev.SpiDev()
spi.open(0,0) # SPI0 and SPI0_CE0_N
spi.lsbfirst = False
spi.cshigh = False
spi.mode = 0b01
spi.bits_per_word = 8
spi.max_speed_hz = 4000000

avrDataByte = 0;
try:
    while True:

        txData = [0x8F, 0x00]
        rxData = spi.xfer(txData)
        for avrDataByte in range(len(rxData)):
            print (hex(rxData[avrDataByte]))
        #spi.close()

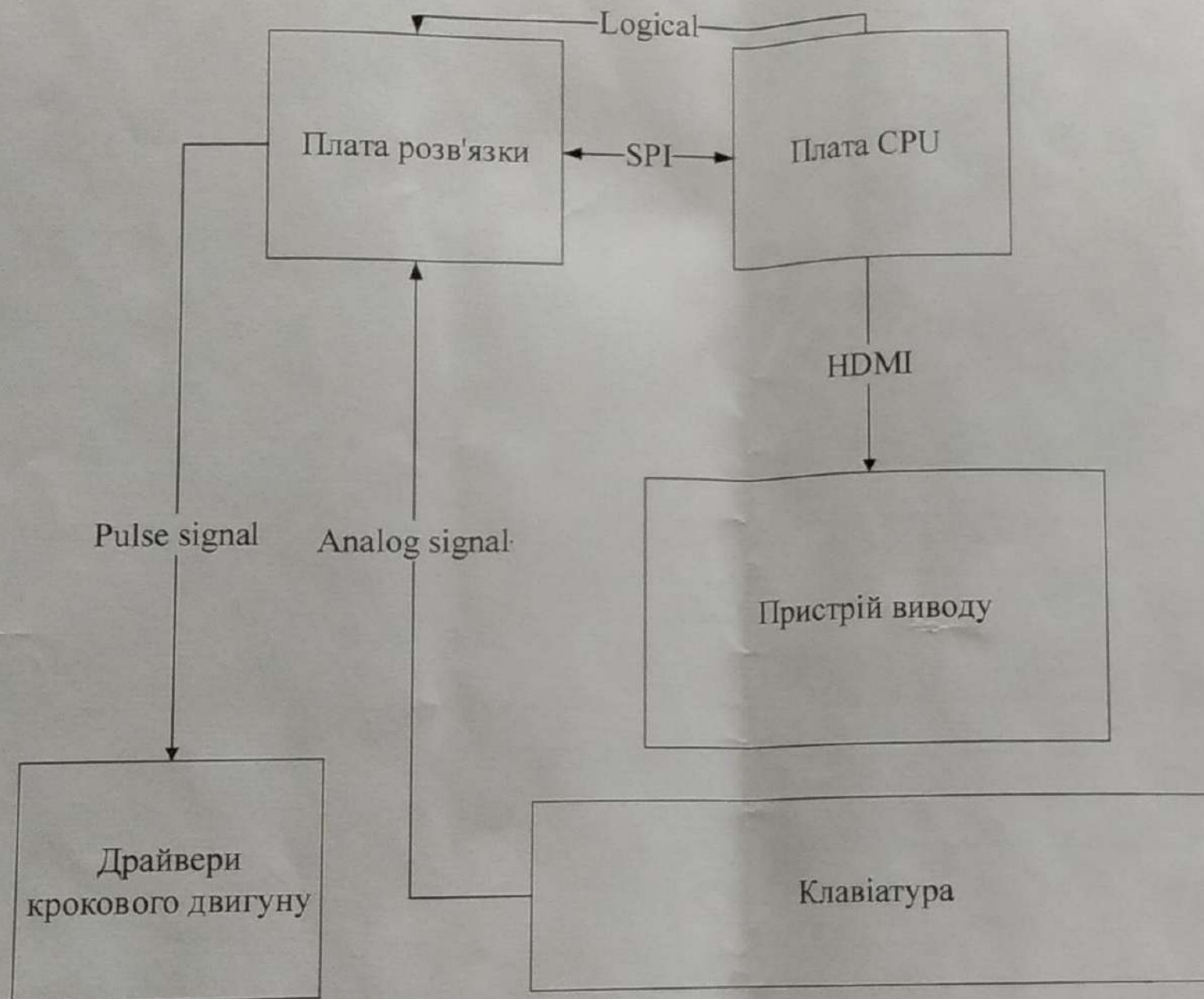
    except KeyboardInterrupt:
        print("exit pressed ctrl+c")

    except:
        print("Other Exception")

    finally:
        print("End of SPItoAVR process")

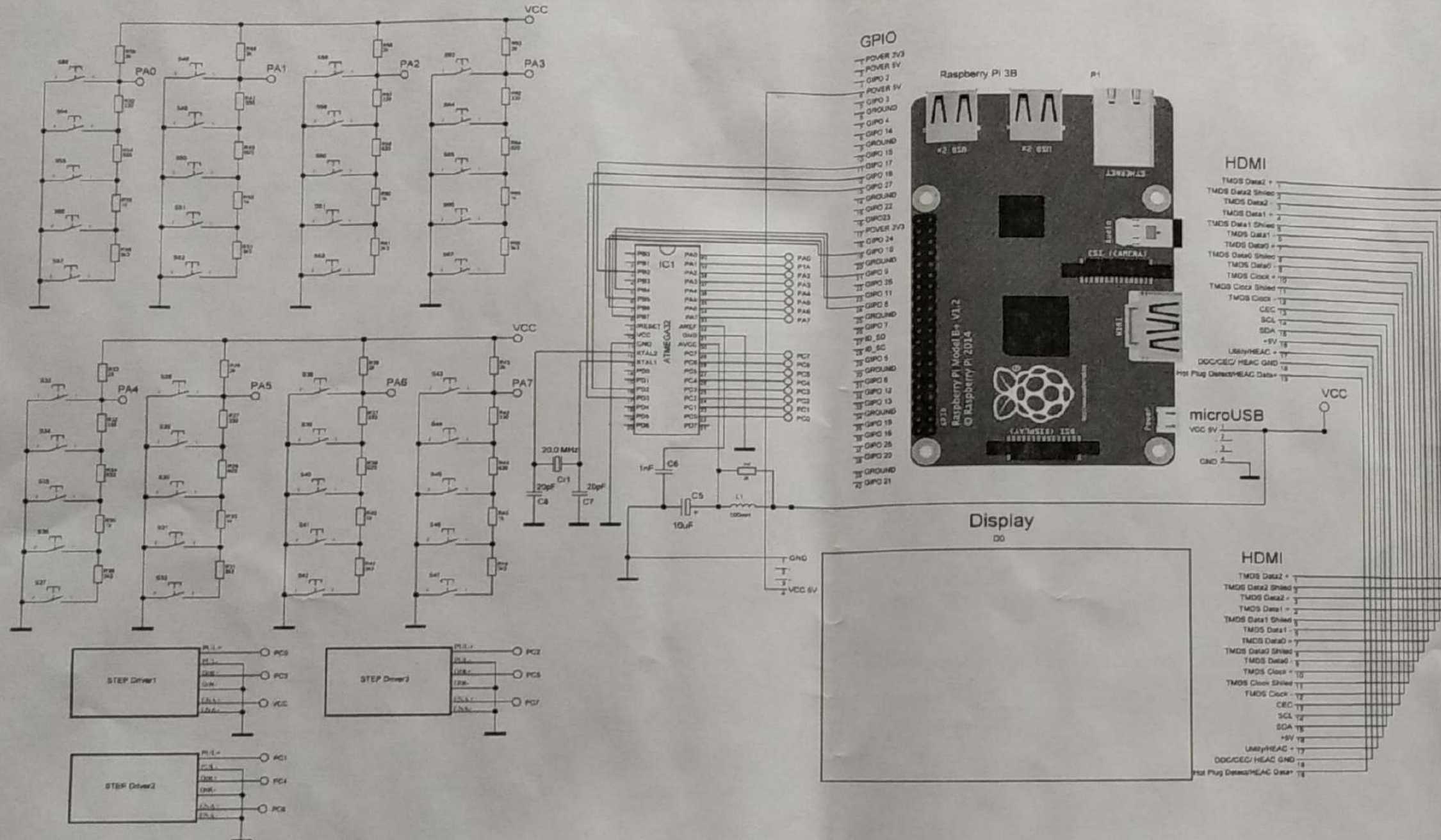
```

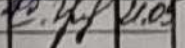
Структурна схема пристрою



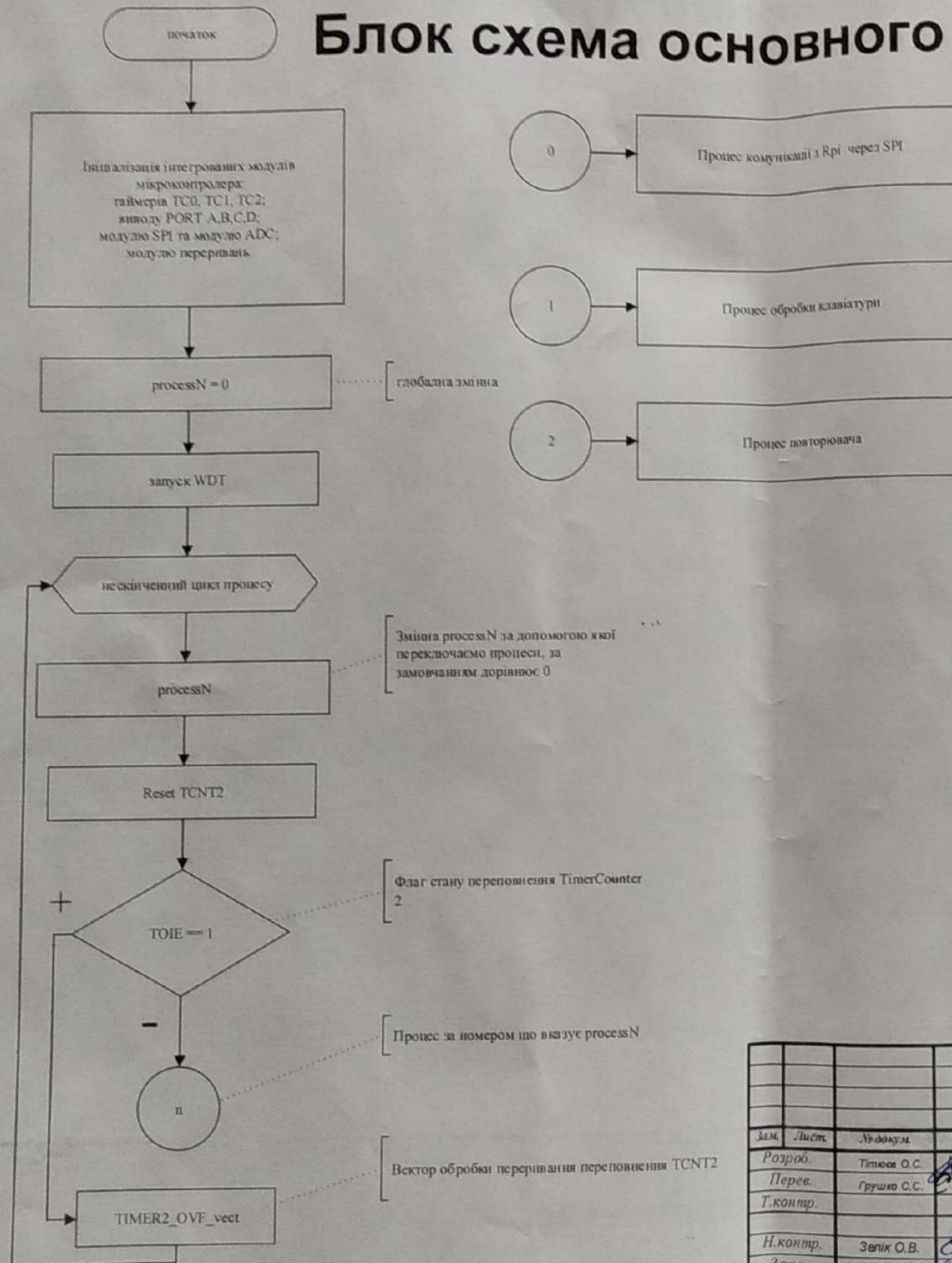
13.02070849.00052 ПЛ1					
Зам.	Лист	№ докум.	Підп.	Дата	Розробка Системи управління кроковим двигуном
Розроб.	Титюк О.С.			21.05	
Перев.	Грушко С.С.			21.05	
Т.контр.					
Н.контр.	Зелік О.В.				Структурна схема пристрою
Затв.	Куберметов Р.К.				
					Лист 1
					Листів 4
					НУ «Запорізька політехніка» КНТ-518сп

Електрична схема пристрою

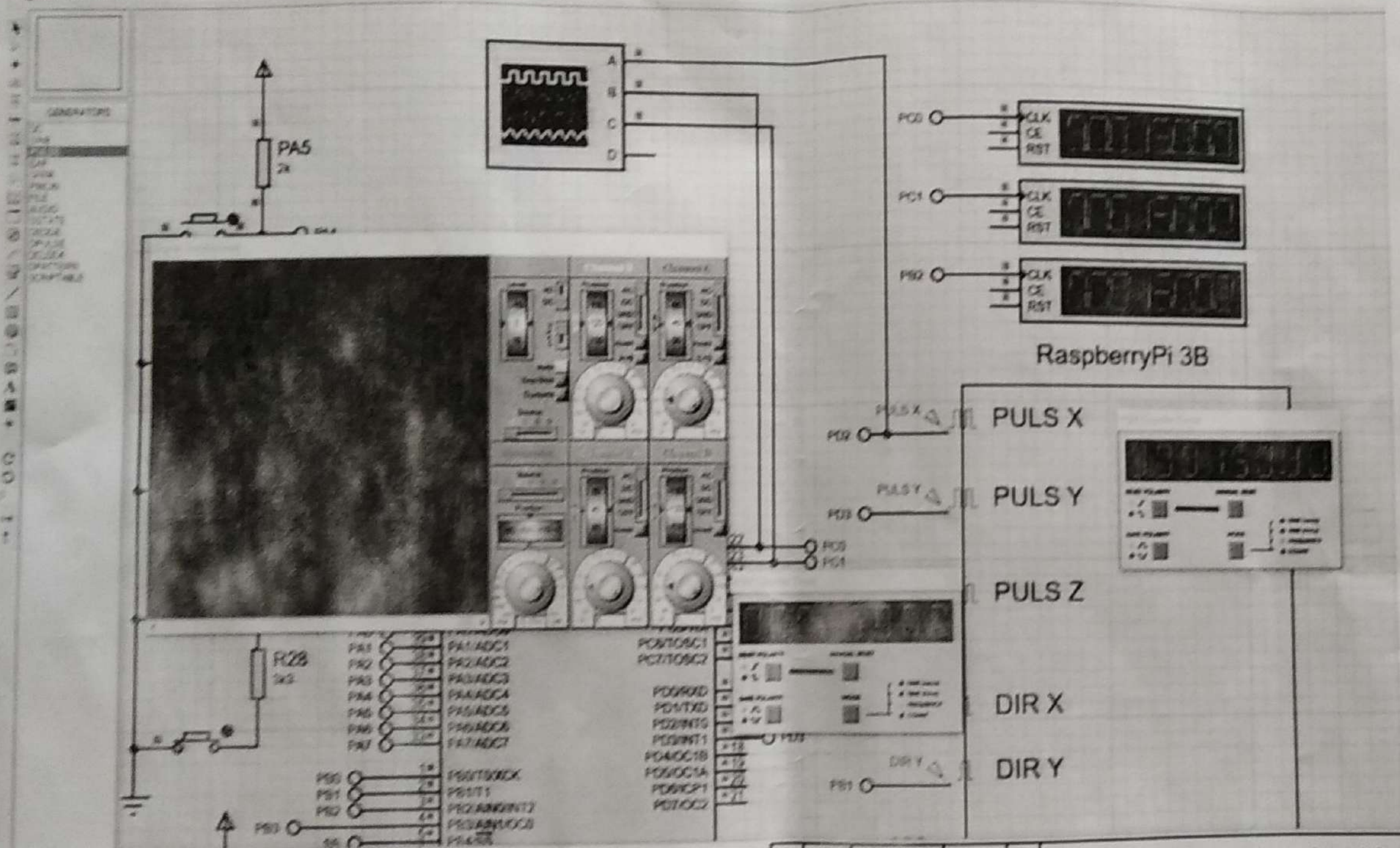


					13.02070849.00052 ПЛ2			
					Розробка системи управління кроковим двигуном	Лист	Маса	Масштаб
Зам.	Лист	№ докум.	Підп.	Дата				
Розроб.		Гітосев О.С.		21.05				
Перев.		Грушко С.С.		21.05				
Т.контр.						Лист 2	Листів 4	
Н.контр.		Зелік О.В.			Електрична схема пристрою	НУ «Запорізька політехніка» КНТ-518сп		
Затв.		Кудиметов Р.К.						

Блок схема основного алгоритму програми



13.02070849.00052 ПЛЗ							
Зам.	Лист	Автори	Підп.	Дата	Розробка Системи управління кроковим двигуном		
Розроб.	Тітюка О.С.	Грушко С.С.	21.05	21.05	Лист 3		
Перев.	Грушко С.С.				Листів 4		
Т.контр.					Блок схема основного алгоритму програми мікроконтролеру		
Н.контр.	Зелік О.В.				НУ «Запорізька політехніка»		
Затв.	Кудерметов Р.К.				КНТ-518сп		



13.02070849.00052 ПЛЧ

					Лист 4		
Зам.	Лист	Ж. Визнач.	Підп.	Дат.	Лист 4		
Розроб.	Григор'єв С.С.			21.11	Лист 4		
Перев.	Григор'єв С.С.			21.05	Лист 4		
Т. констр.					Лист 4		
Н. констр.	Зават О.В.				Лист 4		
Зам.	Григор'єв С.С.				Лист 4		

Розробка системи управління кроковим двигуном

Результат тестування пристрою

НУ «Запорізька політехніка»
 КНТ-518сп