БАКАЛАВРСЬКА РОБОТА

БР.КІ-19.00.00.000 ПЗ

Група КІ-21-1

Сеньків Богдан-Арсен

2025

Міністерство освіти і науки України
Івано-Франківський національний технічний університет нафти і газу
Факультет інформаційних технологій
Кафедра комп'ютерних систем і мереж

Сеньків Богдан-Арсен Михайлович

УДК 004.3

БАКАЛАВРСЬКА РОБОТА

Розробка багатофункціональної автономної монетниці з LCD-дисплеєм на базі мікроконтролера ATmega328

Комп'ютерна інженерія
(назва освітньої програми)

123 - Комп'ютерна інженерія
(шифр і назва спеціальності)

Робота містить результати власних досліджень, використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело:

Здобувач освітнього ступеня *Сеньків Б.М.*
(підпис, ініціали та прізвище здобувача)

Науковий керівник *Бабчук С.М., к.т.н., доцент*
(підпис, прізвище, ім'я, по батькові, науковий ступінь, вчене звання керівника)

Допущено до захисту
Завідувач кафедри КСМ

д.т.н.,проф. *С.І. Мельничук*
(посада) (підпис) (дата) (ініціали та прізвище)

Івано-Франківськ – 2025 рік

Івано-Франківський національний технічний університет нафти і газу

(повне найменування вищого навчального закладу)

Факультет *Інформаційних технологій*

Кафедра *Комп'ютерних систем і мереж*

Освітній ступінь *бакалавр*

Спеціальність *123 – Комп'ютерна інженерія*

(шифр і назва)

ЗАТВЕРДЖУЮ

Завідувач кафедри КСМ

(С.І. Мельничук)

« 05 » *травня* 2025 року

З А В Д А Н Н Я

НА БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТОВІ

Сенькову Богдану-Арсену Михайловичу

(прізвище, ім'я, по батькові)

1. Тема проекту (роботи*)* *Розробка багатофункціональної автономної монетниці з LCD-дисплеєм на базі мікроконтролера ATmega328*

керівник проекту (роботи) *Бабчук С. М., к.т.н., доцент*

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу від *05.05.2025 № 275/7*

2. Строк подання студентом роботи *12 червня 2025 р*

3. Вихідні дані до роботи *Матеріали і результати отримані під час проходження переддипломної практики, методичні вказівки, технічна література.*

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) *1. Аналіз предметної області та існуючих аналогів лічильників монет і банкнот. 2. Розробка апаратних рішень для багатофункціональної автономної монетниці з lcd-дисплеєм на базі мікроконтролера ATmega328. 3. Розробка програмного забезпечення та перевірка працездатності*

багатофункціональної автономної монетниці з lcd-дисплеєм на базі мікроконтролера ATmega328 .

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

_____

6. Консультанти розділів роботи

7. Дата видачі завдання *29 січня 2025 р.*

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
1	*Аналіз предметної області, існуючих пристроїв та їх компонентів*	*Лютий, 2025*	
2	*Розробка та збір системи*	*Квітень, 2025*	
3	*Розробка програмного забезпечення та перевірка працездатності розробленої системи*	*Травень, 2025*	
4	*Оформлення пояснювальної записки*	*Червень, 2025*	

Студент _____ *Сеньків Б.М.*
(підпис) (прізвище та ініціали)

Керівник роботи _____ *Бабчук С.М.*
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Метою даної бакалаврської роботи є розробка багатофункціональної автономної монетниці з LCD-дисплеєм на базі мікроконтролера ATmega328.

В першому розділі роботи проведений аналіз предметної області та існуючих аналогів лічильників монет і банкнот.

В другому розділі було розроблено структурну схему багатофункціональної автономної монетниці, обґрунтовано та вибрано необхідні компоненти та розроблено принципову електричну схему підключення елементів системи.

В третьому розділі було розроблено необхідне програмне забезпечення та здійснено перевірку працездатності розробленої системи. Встановлено, що створена багатофункціональна автономна монетниця виконує покладені на неї функції.

Ключові слова: МОНЕТОПРИЙМАЧ, КУПЮРОПРИЙМАЧ, МОНЕТНИЦЯ, ATMEGA328, LCD-ДИСПЛЕЙ.

ABSTRACT

The purpose of this bachelor's thesis is to develop a multifunctional autonomous coin counter with an LCD display based on the ATmega328 microcontroller.

The first section of the work analyzes the subject area and existing analogs of coin and banknote counters.

In the second section, we developed a block diagram of a multifunctional autonomous coin counter, substantiated and selected the necessary components, and developed a basic electrical circuit for connecting the system elements.

In the third section, we developed the necessary software and tested the performance of the developed system. It has been established that the created multifunctional autonomous coin box performs the functions assigned to it.

Keywords: COIN ACCEPTOR, BILL ACCEPTOR, COIN DISPENSER, ATMEGA328, LCD DISPLAY.

ЗМІСТ

с.

ПЕРЕЛІК ОСНОВНИХ ПОЗНАЧЕНЬ І СКОРОЧЕНЬ

АЦП – аналого-цифровий перетворювач;

COM-порт - серійний порт комп’ютера для зв’язку з мікроконтролером;

EEPROM (*Electrically Erasable Programmable Read-Only Memory)* - енергонезалежна пам’ять, що програмується і стирається електрично;

I2C - послідовна шина даних для зв'язку інтегральних схем;

LCD (*Liquid Crystal Display*) - рідкокристалічний дисплей;

USB (*Universal Serial Bus*) – універсальна послідовна шина.

ВСТУП

Готівка, незважаючи на розвиток безготівкових платежів, все ще відіграє важливу роль у фінансових операціях. Проте, підрахунок великої кількості банкнот і монет вручну може бути складним процесом, забирати багато часу та призводити до помилок. Лічильники банкнот і монет допомагають зробити цей процес швидшим, точнішим та надійнішим, спрощуючи роботу з готівкою та підвищуючи безпеку обліку коштів.

Актуальність теми. Розробка багатофункціональної автономної монетниці з LCD-дисплеєм на базі мікроконтролера ATmega328 є актуальною через потребу в автоматизованих рішеннях для обліку та зберігання готівкових коштів у побуті. Часто після покупок залишаються дрібні монети і банкноти, які складно впорядкувати та підрахувати вручну. Логічним вирішенням є накопичення цих коштів та автоматизація процесу їх підрахунку. Саме тому є доцільною розробка простого і доступного пристрою, який об’єднає в собі функції лічильника банкнот і монет, а також забезпечить їхнє зберігання, виконуючи роль скарбнички.

Об’єктом дослідження є процес автоматизованого обліку готівкових коштів.

Предметом дослідження є системи підрахунку та обліку монет і банкнот.

Метою даної дипломної роботи є розробка багатофункціональної автономної монетниці з LCD-дисплеєм на базі мікроконтролера ATmega328, що забезпечить автоматичне визначення номіналу, підрахунок, зберігання та виведення інформації про кількість та суму готівки. Основні завдання роботи включають:

– аналіз існуючих аналогів;

– розробка апаратної частини, що включає в себе систему сенсорів для визначення номіналу монет та банкнот, дисплей для відображення інформації та інші необхідні компоненти;

– розробка програмного забезпечення для мікроконтролера ATmega328, яке забезпечуватиме збір та обробку даних із сенсорів, а також збереження цих даних;

– проведення перевірки працездатності розробленої системи.

Методи дослідження. Під час проведення дослідження існуючих систем для підрахунку банкнот і монет було використано метод порівняльного аналізу; для створення багатофункціональної автономної монетниці з LCD-дисплеєм на базі мікроконтролера ATmega328 використано метод швидкого прототипування.

Практичне значення отриманих результатів полягає в тому, що розроблена багатофункціональна автономна монетниця дозволить зменшити час підрахунку монет та банкнот, зменшить вплив людського фактору на процес обліку, підвищить зручність зберігання готівки.

Таким чином, впровадження розробленої багатофункціональної автономної монетниці з LCD-дисплеєм на базі мікроконтролера ATmega328 не лише підвищить зручність та інтерес до накопичення коштів, але й може зробити значний внесок у розвиток побутових фінансових операцій.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ІСНУЮЧИХ АНАЛОГІВ ЛІЧИЛЬНИКІВ МОНЕТ І БАНКНОТ

1.1 Аналіз предметної області

Лічильники монет та купюр є системати, які призначені для автоматизації процесу підрахунку з можливістю сортування готівки в умовах великих обсягів готівкових операцій.

Такі системи широко використовуються в банківських установах, супермаркетах, транспортних компаніях, ігрових автоматах та інших сферах, де потрібна висока швидкість і точність підрахунку коштів [1].

Основна функція лічильників монет і купюр полягає в автоматизації процесу підрахунку та перевірки справжності грошей. Це дозволяє значно скоротити час, витрачений на ручний підрахунок, зменшити ймовірність помилок, а також мінімізувати ризики шахрайства. Сучасні пристрої здатні розпізнавати не лише номінал, але й інші характеристики, такі як валюта, справжність та стан купюри чи монети.

Для побутового використання, такі системи можуть бути особливо корисні для людей, які часто працюють із готівкою або бажають організувати свої фінанси. Такі пристрої дозволяють швидко та точно підраховувати гроші, зменшувати ризик помилок і заощаджувати час.

Технології, що використовуються в лічильниках для домашнього використання, зазвичай базуються на простих і доступних сенсорах.

1.2 Аналіз існуючих аналогів лічильників монет та банкнот

На сьогоднішній день на ринку представлена велика кількість різноманітних за своїми характеристиками, функціональністю та ціновою категорією лічильників купюр та монет. Вартість таких пристроїв варіюється від 3500 грн до 138 600 грн, залежно від виробника та моделі.

Лічильники, що використовують сучасні технології, перевіряють автентичність, зношеність та пошкодження грошей та можуть мати автоматичні системи для сортування за номіналами та пакування. Це дозволяє забезпечити не тільки точний підрахунок, але й високу надійність при роботі з великими обсягами готівки.

Базові моделі лічильників можуть бути оснащені тільки функцією підрахунку, тому є значно дешевшими.

Лічильник монет Native C8 (рис.1.1) є одним із сучасних пристроїв для автоматизованого підрахунку монет, що забезпечує зручний і швидкий спосіб сортування та підрахунку монет для малого бізнесу і приватного використання .

Рисунок 1.1 – Native C8

Цей лічильник дозволяє користувачам сортувати монети за номіналом та підраховувати загальну суму за короткий час. Він оснащений інтуїтивним дисплеєм, що відображає кількість монет та загальну суму. Native C8 є легким у використанні, забезпечує високу точність і швидкість, а також має функцію автоматичного зупинення при переповненні відсіків [2].

Орієнтовна вартість лічильника Native C8 на українському ринку становить близько 15 000 грн.

Cassida C550 (рис.1.2) є високоякісним сучасним банківським обладнання, яке швидко і точно перераховує металеві гроші, уникнувши помилок, які можуть виникнути через людський фактор. Використовується в установах середнього та великого рітейлу, а також у розважальних центрах, супермаркетах, транспортних та інших компаніях, які потребують точного підрахунку великого обсягу монет.

Рисунок 1.2 – Cassida C550

Особливістю цього лічильника є конструкція, яка забезпечує безперешкодний доступ до тракту, що дає змогу користувачеві оперативно розв'язувати дрібні проблеми та швидко проводити техобслуговування. Також його дозатор у завантажувальному бункері керує завантаженням машини за допомогою трьох швидкостей подавання монет, що перешкоджає «захлинанню» апарату та має зручне кріплення мішка для оброблених монет [3].

На українському ринку вартість Cassida C550 починається від 22 131 грн.

CTcoin Pelican 309 (рис.1.3) є високонадійним продуктивним сортувальником монет професійного банківського обладнання, призначеного для високоякісної обробки великого обсягу монет.

Рисунок 1.3 – CTcoin Pelican 309

Відмінною рисою даного сортувальника монет є унікальна технологія сенсора CTcoin 5-го покоління з функцією перевірки сплаву монет, яка забезпечує найвищу точність перерахунку та перевірки монет: відбраковування іноземних монет, підроблених монет, жетонів тощо. Особливістю даного пристрою є безліч різноманітних рахувань і підрахункових звітів, функції пам'яті та коди помилок для легкого вирішення проблем [4].

Його вартість починається від 138 600 грн.

У таблиці 1.1 представлено порівняння цих трьох лічильників монет.

Таблиця 1.1 – Порівняння лічильників монет

Характеристика	**Назва пристрою**		
	Native C8	**Cassida C550**	**CTcoin Pelican 309**
Місткість завантажувального бункера (кількість монет)	500	3000 (можливе розширення до 11000)	До 3000 монет
Швидкість (монет/хв)	250	2300	1100
Кількість приймальних кишень	8	2	10
Функція сортування за номіналом	+	-	+
Режими роботи	Перерахунок, Фасування, Сортування	Фасовка, Звичайний рахунок	Рахунок, Калькуляція за номіналом, Фасування, Сортування

Кінець таблиці 1.1

Характеристика	Назва пристрою		
	Native C8	**Cassida C550**	**CTcoin Pelican 309**
Споживана потужність	<20 Вт	< 60 Вт	33 Вт
Габарити (мм)	345 x 306 x 260	260 × 350 × 350	705 x 333 x 471
Вага (кг)	3,9	8.8	28
Вартість (грн)	15000	22100	138 600

Cassida C550 виділяється швидкістю перерахунку до 2300 монет/хв і підходить для великих обсягів, але не має функції сортування за номіналом.

CTcoin Pelican 309 має велику кількість приймальних кишень, високу швидкість (1100 монет/хв), найкращу точність та детекцію за допомогою функції перевірки сплаву, але є найбільш громіздким та дорогим варіантом.

Native C8 – компактний і дешевий варіант з базовою місткістю та функцією сортування, підходить для малого бізнесу та домашнього використання.

Спільним недоліком Cassida C550, CTcoin Pelican 309 та Native C8 є їх висока вартість (від 15000 грн до 138600 грн).

Портативний лічильник банкнот PRO 15 (рис.1.4) є найдешевшим ручним портативним лічильником банкнот, який може працювати від батарейок [5].

Рисунок 1.4 – PRO 15

Пристрій працює з будь-якими валютами і призначений для швидкого перерахунку готівки в невеликому обсязі. Головна перевага цього лічильника - мобільність і можливість скористатися в будь-якій ситуації.

Орієнтовна лічильник банкнот PRO 15 на українському ринку становить близько 3 500 грн.

Лічильник банкнот BCASH 9500T (рис.1.5) є високопродуктивним лічильником банкнот, призначений для ефективної та точної обробки готівки у комерційних та фінансових установах.

Рисунок 1.5 – BCASH 9500T

Особливістю цього лічильника є перевірка справжності банкнот і калькуляція по номіналах (можливість вручну вказати номінал) [6].

На українському ринку вартість лічильник банкнот BCASH 9500T починається від 12 500 грн.

Лічильник банкнот Cassida Xpecto (рис.1.6) є багатофункціональним лічильником банкнот професійного класу з автоматичним визначенням номіналу. Він може автоматично розпізнавати чотири основних валюти з можливістю розширення цього списку ще на десять валют та перевіряє справжності за чотирма ознаками: видимим чином банкноти, ультрафіолетовим, магнітним та інфрачервоним міткам [7].

Рисунок 1.6 – Cassida Xpecto

Вартість лічильник банкнот Cassida Xpecto починається від 38 610 грн.

У таблиці 1.2 представлено порівняння вищевказаних трьох лічильників банкнот.

Таблиця 1.2 – Порівняння лічильників банкнот

Характеристика	**Назва пристрою**		
	PRO 15	**BCASH 9500T**	**Cassida Xpecto**
Ємність завантажувальної кишені, бакнот	100	500	500
Швидкість рахунку, банкнот/хв	600	900, 1200, 1500	800, 1000, 1200
Ємність приймальної кишені, банкнот	100	250	200
Старт	ручний	ручний та автоматичний	ручний та автоматичний
Валюти	Усі	більшість основних валют	гривні, євро, долари, польські злоті (можна додати ще 10 валют)
Габарити, мм	75 x 190 x 102	310x250x180	265x240x230
Вага, кг	0,45	7,2	5,6
Вартість, грн	3 500	12 500	38 610

Кінець таблиці 1.2

Характеристика	Назва пристрою		
	PRO 15	**BCASH 9500T**	**Cassida Xpecto**
Види детекцій	-	за розміром; за оптичною щільністю; за магнітними мітками; ультрафіолетова (УФ); інфрачервона (ІЧ).	ультрафіолетова (UV); магнітна (MG); інфрачервона (IR); детекція за образом (2 CIS); детекція за щільністю (DD); детекція половинчастих банкнот; детекція за розміром (SD)
Функції	рахунок; підсумовування.	автоматичний старт; підсумовування.	автоматичне розпізнавання номіналу; сортування за номіналом; сортування за лицем; сортування за орієнтацією; фасування за кількістю; зчитування серійних номерів (USD, EUR); звіт по прорахованим банкнотам; автовизначення валюти; мультивалютний депозит.

PRO 15 є найбільш доступним і компактним лічильником банкнот, але має мінімальні функції та низьку швидкість обробки. Його перевагою над іншими лічильниками є автономний режим роботи, що дозволяє брати пристрій із собою.

BCASH 9500T на відміну від попереднього лічильника, може не лише визначати кількість купюр, а і детально перевіряти їх справжність. Також він може робити калькуляцію по номіналах, якщо ввести їх вручну.

Cassida Xpecto є найдорожчим пристроєм серед розглянутих лічильників банкнот. Його основною перевагою є розширений набір функцій, таких як автоматичне розпізнавання номіналу, сортування за різними критеріями та мультивалютна підтримка.

Спільним недоліком PRO 15, BCASH 9500T та Cassida Xpecto є їх висока вартість (від 3 500 грн до 38 610 грн).

Досліджено, що лічильники монет та купюр включають кілька основних компонентів:

– процесор (або мікроконтролер);

– інтерфейс користувача (екран, кнопки);

– сенсори;

– механізм прийому.

Процесор виконує обробку даних, отриманих від сенсорів. Він аналізує характеристики монет чи купюр, підраховує їх кількість та взаємодіє з іншими частинами пристрою. Процесор також відповідає за керування роботою лічильника.

Інтерфейс користувача забезпечує взаємодію між пристроєм і користувачем. Екран відображає інформацію про кількість порахованих одиниць, загальну суму та інші важливі дані. Кнопки дозволяють користувачеві вибирати режими роботи, змінювати налаштування та виконувати інші дії.

Сенсори ідентифікують монети або купюри, визначаючи їх номінал, розмір, матеріал, колір та інші параметри. Вони допомагають відрізнити справжні гроші від підробок і виявляють можливі дефекти. Сенсори можуть бути різних типів, таких як оптичні, магнітні або ультразвукові.

Механізм прийому направляє монети або купюри у зону обробки. Він складається з ременів, роликів, шестерень та інших рухомих елементів, які забезпечують точне переміщення грошей.

Лічильники монет, зазвичай, розпізнають монети за допомогою:

– магнітної індукції;

– оптичного розпізнавання;

– вимірювання ваги;

– аналізу діаметру та товщини.

У детекторах монет використовують різні датчики та їх комбінації. Найпоширенішими датчиками є:

– оптичні;

– електромагнітні;

– удару.

Розпізнавання купюр є складнішим процесом через велику кількість можливих захисних елементів. Тому лічильники банкнот, як правило, розпізнають банкноти за допомогою:

– ультрафіолетового випромінювання;

– інфрачервоного випромінювання;

– магнітного розпізнавання;

– оптичного сканування;

– аналізу електропровідності.

Основні типи датчиків, що використовуються в купюроприймачах:

– оптичні;

– магнітні;

– індукційні;

– ємнісні;

– ультразвукові.

Висновки до розділу

Проведений аналіз показав, що на українському ринку є досить широкий вибір лічильників монет та банкнот, які варіюються за ціною, якістю та функціоналом. Проте немає пристрою, який б об’єднував обидві функції, та допомагав користувачам з накопиченням їх коштів.

Встановлено, що всі існуючі лічильники є дуже дорогими (від 3 500 грн до 138 600 грн) і тому вони є недоступними для широкого кола користувачів.

Тому метою бакалаврської роботи є створення зручної у використанні недорогої багатофункціональної монетниці, яка об’єднає в собі функції як лічильника банкнот так і лічильника монет, забезпечить зберігання банкнот і монет.

2 РОЗРОБКА АПАРАТНИХ РІШЕНЬ ДЛЯ БАГАТОФУНКЦІОНАЛЬНОЇ АВТОНОМНОЇ МОНЕТНИЦІ З LCD-ДИСПЛЕЄМ НА БАЗІ МІКРОКОНТРОЛЕРА ATMEGA328

2.1 Розробка структурної схеми

На рисунку 2.1 зображена розроблена структурна схема багатофункціональної автономної монетниці.

Рисунок 2.1 – Структурна схема

Мікроконтролер є центральним елементом, який обробляє сигнали від датчиків монет і купюр. Він зберігає дані про загальну суму накопичень у пам'яті й забезпечує взаємодію з іншими компонентами.

Датчик кольору та оптичний датчик забезпечують розпізнавання введених коштів. Купюроприймач також містить двигун постійного струму, який затягує банкноти в середину монетниці після успішного розпізнавання номіналу.

За допомогою кнопок вмикається пристрій, калібрується монето- та купюроприймач, очищається пам'ять, переглядається статистика кількості монет та інша службова інформація.

Дисплей призначений для виводу інформації про загальну кількість накопичених коштів, заряд акумулятора, номінал введеної купюри чи монети, а також додаткових функцій.

Акумулятор забезпечує автономну роботу пристрою, живлячи мікроконтролер, датчики, дисплей та інші компоненти. Він дозволяє використовувати монетницю без підключення до мережі живлення та інформує користувача про рівень заряду через дисплей.

2.2 Визначення способу розпізнавання монет та купюр

Основною функцією монетоприймача та купюроприймача є визначення номіналу грошей.

У таблиці 2.1 наведено характеристики українських обігових монет різних номіналів, що перебувають в обігу [8].

Таблиця 2.1 – Характеристики українських монет різних номіналів

Параметри	**Номінал монети**					
	10 грн	**5 грн**	**2 грн**	**1 грн**	**50 коп**	**1 грн (старого зразка)**
Метал	цинковий сплав із гальванічним покриттям нікелем		низьковуглецева сталь з гальванопокриттям нікелем		низьковуглецева сталь з гальвано-покриттям латунню	алюмініє ва бронза
Діаметр, мм	23,5	22,1	20,2	18,9	23	26
Товщина, мм	2,3	2,1	1,8	1,7	1,55	1,85
Вага, г	6,4	5,2	4	3,3	4,2	7,1

Найпростішим методом визначення номіналу монети є вимірювання її діаметра, оскільки кожен номінал має унікальний розмір. Крім того, монети різних номіналів мають достатню різницю діаметра, що дозволяє їх розрізнити.

У нашій розробці розмір монети буде визначатись оптичним методом за допомогою інфрачервоного світлодіода та фотодіода. Принцип роботи представлений на рисунку 2.2. Світло інфрачервоного діапазону буде

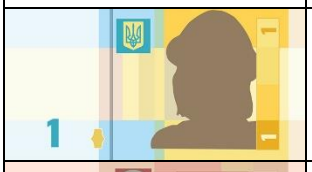
проходити крізь щілину, де розташована монета, а фотодіод фіксуватиме її розмір за інтенсивністю поглиненого світла, що дозволить визначити її розмір.

Рисунок 2.2 – Схема оптичного визначення номіналу монети

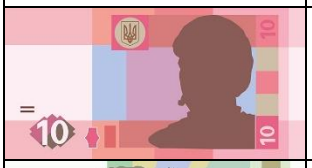
Різні монети закривають різну частину щілини. Відповідно від розміру монети залежить величина сигналу від датчику.

У таблиці 2.2 наведено характеристики українських банкнот різних номіналів, що перебувають в обігу [9].

Таблиця 2.2 - Характеристики українських банкнот різних номіналів

Зображення	Номінал	Розміри (мм)	Основний колір	Опис	
				Аверс	Реверс
	1	118 × 63	Жовто-синій	Володимир Великий	Місто Володимира (Київ)
	2		Жовто-коричневий	Ярослав Мудрий	Софійський собор (Київ)
	5		Синій	Богдан Хмельницький	Іллінська церква (с. Суботів)

Кінець таблиці 2.2

Зображення	Номінал	Розміри (мм)	Основний колір	Опис	
				Аверс	Реверс
	10	124 × 66	Червоний	Іван Мазепа	Панорама Києво-Печерської Лаври
	20	130 × 69	Зелений	Іван Франко	Львівський оперний театр
	50	136 × 72	Фіолетовий	Михайло Грушевський	Будинок Центральної Ради (Київ)
	100	142 × 75	Жовто-зелений	Тарас Шевченко	Київський національний університет імені Тараса Шевченка
	200	148 × 75	Рожевий	Леся Українка	Замок Любарта (Луцьк) та лелеки що летять
	500	154 × 75	Бежевий	Григорій Сковорода	Києво-Могилянська академія
	1000	160 × 75	Блакитний	Володимир Вернадський	Будівля Президії Національної академії наук України

Визначення номіналу банкноти є доволі складним завданням, яке потребує цілого комплексу датчиків. Одним із найпростіших методів є аналіз кольору, оскільки кожен номінал має унікальний колір, який можна ідентифікувати.

У нашій розробці (рис.2.3) ми будемо використовувати датчик кольору для визначення номіналу купюри. Датчик сканувати поверхню банкноти, визначаючи її основний колір.

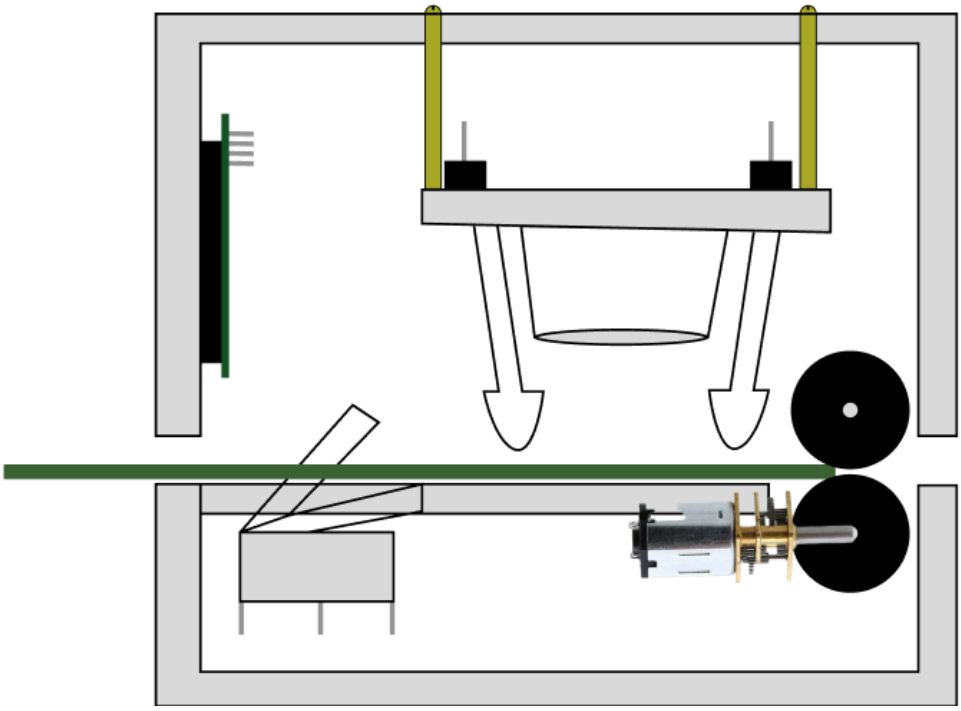


Рисунок 2.3 – Схема купюроприймача

Кожен номінал банкноти відповідає конкретному діапазону кольорів у моделі RGB, який буде порівнюватися з еталонними значеннями.

2.3. Вибір апаратного забезпечення

Для реалізації багатофункціональної автономної монетниці було обрано необхідне апаратне забезпечення, яке наведене в таблиці 2.3 [10-25].

Таблиця 2.3 – Вартість апаратного забезпечення

Назва компонента	**Вартість, грн**	**Кількість**	**Сума, грн**
Arduini Nano	177	1	177
LCD-дисплей I2C (10x2)	129	1	129
Датчик кольору	197	1	197
Мікромотор з редуктором N20	167	1	167
UPS 18650 5V	160	1	160
Акумулятор 18650 3500мА	250	1	250
Інфрачервоний світлодіод	5	1	5

Кінець таблиці 2.3

Назва компонента	**Вартість, грн**	**Кількість**	**Сума, грн**
Фотодіод	5	1	5
Транзистор 50N024	67	1	67
Резистор (220Ом, 2МОм, 10кОм)	5	7	35
Діод	4	1	4
Конденсатор 100nF	2,50	1	2,50
Кнопка	2	2	4
Мікровимикач	45	1	45
Провідники (20шт) мама-мама	25	1	25
Провідники (20шт) мама-тато	25	1	25
Корпус з фанери з порізкою та гравіюванням	75	1	75
Загальна вартість			**1372,50**

Таблиця вартості (табл.2.3) наочно показує структуру витрат на компоненти, необхідні для реалізації багатофункціональної автономної монетиці. Загальна сума витрат становить 1372,50 грн.

Найбільше коштів було витрачено на UPS 5v з акумулятором 18650 (410 грн) та плату Arduino Nano (177 грн). Arduino Nano є головним елементом, оскільки він виконує обробку даних, зберігає дані в пам’ять та забезпечує зв’язок із датчиками. Джерело живлення, хоча й є дорогим, але додає системі зручності та автономності.

До інших важливих компонентів відносяться датчик кольору GY-31 TCS230 (197 грн), мікромотор з редуктором N20 (167 грн) та інфрачервоний світлодіод з фотодіодом (10 грн). Вони забезпечують функції точного вимірювання даних, що є основою роботи системи. Витрати на провідники, резистори, транзистор, конденсатор, діод, кнопки, мікровимикач є відносно невеликими, що позитивно впливає на загальний бюджет.

Створено принципову електричну схему (рис.2.4) у середовищі Fritzing, яка відображає взаємозв’язки між компонентами розробленого пристрою.

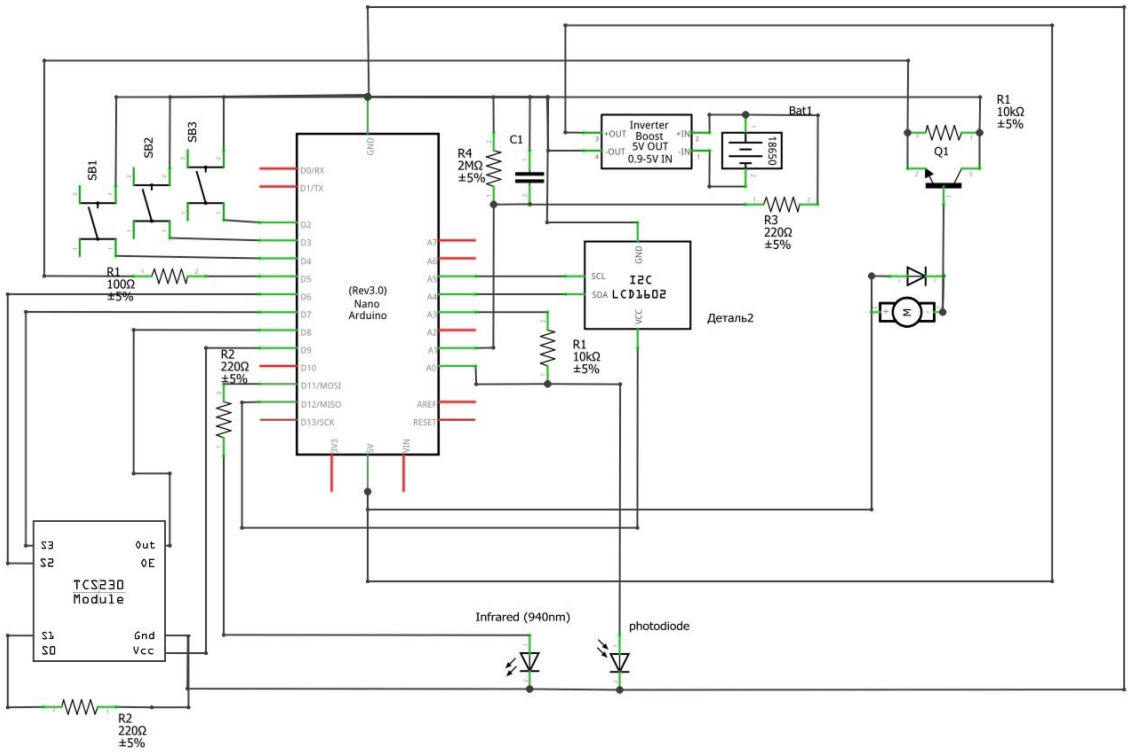


Рисунок 2.4 – Принципова електрична схема

Схема зображена на рисунку 2.4 ілюструє підключення елементів, таких як датчики, кнопки, рідкокристалічний дисплей та контролер Arduino Nano.

Висновки до розділу

В даному розділі було здійснено розробку структурної схеми багатофункціональної автономної монетниці а також було обрано компоненти для неї. Крім того, було створено принципову електричну схему багатофункціональної автономної монетниці, яка відображає взаємозв’язки між її компонентами.

3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ТА ПЕРЕВІРКА ПРАЦЕЗДАТНОСТІ БАГАТОФУНКЦІОНАЛЬНОЇ АВТОНОМНОЇ МОНЕТНИЦІ З LCD-ДИСПЛЕЄМ НА БАЗІ МІКРОКОНТРОЛЕРА ATMEGA328

3.1 Розробка програмного забезпечення

Для забезпечення функціональності пристрою його необхідно запрограмувати. Для цього в Arduino IDE було розроблено ПЗ, яке реалізує зчитування даних із монетоприймача та купюроприймача, виводить ці дані на LCD дисплей, керування кнопками, а також режим глибокого енергозбереження з індикацією заряду акумулятора.

Функціонал багатофункціональної автономної монетниці:

– вимірювання діаметра монет і його прив'язка до номіналу кожної монети;

– вимірювання інтенсивності кожного з трьох кольорів r,g,b купюри та їх прив'язка до номіналу кожної банкноти;

– обчислення загальної суми монет і банкнот у монетниці. статистика за кількістю монет і банкнот кожного номіналу;

– всі налаштування і накопичена сума зберігаються в незалежну пам'ять і не скидаються при відключенні або збої живлення;

– режим глибокого енергозбереження;

– калібрування монетниці для будь-якого числа номіналів монет і банкнот;

– скидання накопиченої суми грошей у монетниці;

– контроль залишку заряду батареї.

Для роботи цього проекту потрібні кілька бібліотек, які додаються на початку коду:

#include <LowPower.h> // бібліотека для керування режимами енергозбереження

#include <EEPROMex.h> // бібліотека для роботи з внутрішньою пам'яттю

#include <LCD_1602_RUS.h> // бібліотека для LCD дисплея

#include <Wire.h> // бібліотека для роботи з I2C інтерфейсом

Всі ці бібліотеки попередньо установлені через Менеджер бібліотек Arduino IDE.

Wire.h та LCD_1602_RUS.h дозволяють працювати з LCD-дисплеєм через інтерфейс I2C та виводити кирилицю. LowPower.h дозволяє перевести мікроконтролер в режим зниженого енергоспоживання, а EEPROMex.h забезпечує більшу функціональність для роботи з внутрішньою пам'яттю, дозволяючи зберігати і читати дані, а також керувати ресурсами пам'яті ефективніше.

На початку коду оголошено піни для підключення компонентів до плати Arduino:

#define button 2 // кнопка «прокинутися для монетоприймача»

#define banknote_button 4 // кнопка «прокинутися для купюроприймача»

#define calibr_button 3 // прихована кнопка калібрування скидання

#define disp_power 12 // живлення дисплея

#define LEDpin 11 // живлення ІЧ діода

#define IRpin A3 // живлення фотодіода

#define IRsens A0 // сигнал фотодіода

#define VoltmetrBatt A1 // вольтметр батареї

// Контакти підключення датчика кольору TCS3200 до Arduino та призначення інших пінів детектора банкнот

#define S2 6 // S2 і S3 вибір фільтра (червоний, зелений, синій)

#define S3 7 // S2 і S3 вибір фільтра (червоний, зелений, синій)

#define sensorOut 8 // вказуємо пін входу частоти від датчика кольору

#define sensor_color_power 9 // живлення датчика кольору

#define motor_power 5 // управління живленням мотора,

Після цього оголошено перелік змінних та їхній тип даних, які використовуються в програмі для роботи з монетами та банкнотами і вольтметром:

int coin_signal; // змінна - максимальне значення сигналу монети, що пролетіла, використовується для всіх номіналів, не зберігаємо

int sens_signal, last_sens_signal; // значення рівня сигналу монети від фотодатчика в динаміці

int coin_signal_min[coin_amount]; // тут зберігається нижнє значення межі діапазону сигналу для кожного розміру (номіналу) монет

int coin_signal_max[coin_amount]; // тут зберігається верхнє значення межі діапазону сигналу для кожного розміру (номіналу) монет

int coin_quantity[coin_amount]; // зберігаємо кількість монет кожного номіналу

int banknote_quantity[banknote_amount]; // кількість банкнот кожного номіналу

int banknote_signal_R[banknote_amount]; // тут зберігається значення частоти червоного кольору для кожної банкноти

int banknote_signal_G[banknote_amount]; // тут зберігається значення частоти зеленого кольору для кожної банкноти

int banknote_signal_B[banknote_amount]; // тут зберігається значення частоти синього кольору для кожної банкноти

int empty_signal_B = 200; // зберігаємо фонове значення частоти синього кольору купюроприймача

byte empty_signal; // зберігаємо фонове значення сигналу детектора монет (пустий_сигнал)

byte N = 10; // створюємо змінну «N», порядковий номер номіналу монети або банкноти. При N = 10 - купюра не визначена

unsigned long standby_timer; // таймер очікування (перехід в сон)

unsigned long reset_timer; // таймер очищення пам'яті, він так само використовується для примусового виходу з калібрування

float summ_money = 0; // сума монет і банкнот у скарбниці

// Зберігання частоти зчитування з Out датчика кольору. Створюємо змінну ... frequency для кожного кольору

int redFrequency = 0; // значення частоти для червоного кольору

int greenFrequency = 0;// значення частоти для зеленого кольору

int blueFrequency = 0;// значення частоти для синього кольору

boolean sleep_flag = true; // прапорець sleep_flag = true (прапор сну)

boolean coin_flag = false;// Прапорець для фіксації прольоту монети

У цьому фрагменті коду налаштовується програмний вольтметр для вимірювання напруги батареї за допомогою резистивного дільника (r3 і r4), підключеного до мікроконтролера.

// r3 і r4 резистори дільника напруги

const float r3 = 215.0; // 220 Ом,

const float r4 = 2050000.0; // 2.0~8.0 МОм

float VCC = 0.0; // напруга живлення мікроконтролера Vcc (безпосередньо на піні Vcc)

float battery_voltage = 0.0; // напруга на батареї

float max_battery_voltage = 4.20; // виставляємо максимальне значення напруги на батареї при повній зарядці (4,10-4,25 в)

float min_battery_voltage = 2.60; // задаємо мінімальну допустиму робочу напругу на батареї (2,60 в)

float battery_charge; // залишок заряду батареї у відсотках (0~100%)

const float vcc_const = 1.097; // 1.0 -- 1.2 константа(const) калібрування напруги

Також створено структуру MyStruct, яка визначає тип даних для зберігання частот трьох кольорів (червоного, зеленого та синього).

```
struct MyStruct {
  int redFrequency; // значення частоти для червоного кольору
  int greenFrequency; // значення частоти для зеленого кольору
  int blueFrequency; // значення частоти для синього кольору
};
```

Потім оголошено функції для сну, читання внутрішньої опорної напруги та обробник переривань:

```
void good_night();// Функція сну
float readVcc();// Функція читання внутрішньої опорної напруги
void wake_up(); // Функція - обробник переривання
```

Далі йде налаштування параметрів системи, де вказано валюту, число номіналів монет та банкнот та їхню вартість, час відображення на дисплеї номіналу монети, яка пролетіла і час бездіяльності, через який система піде в сон.

```
#define coin_amount 6    // число номіналів монет, які потрібно розпізнати
#define banknote_amount 7      // число номіналів банкнот, які потрібно розпізнати
#define coin_amount_calibration 5  //  число монет одного номіналу для калібрування
```

float coin_value[coin_amount] = {1.0, 10.0, 0.50, 5.0, 2.0, 1.0 }; // вартість, номінал монет у порядку зменшення діаметра

float banknote_value[banknote_amount] = {10.0, 20.0, 50.0, 100.0, 200.0, 500.0, 1000.0}; // номінали банкнот

String currency = "UAH"; // валюта UAH - Українська гривня

int displ_coin_value_time = 2000; // час відображення на дисплеї номіналу монети, що пролетіла

int stb_time = 30000; // час бездіяльності, через який система піде в сон

Функція setup() ініціалізує необхідні компоненти для роботи пристрою. Спочатку налаштовується серійний монітор на швидкість 38400 бод для виведення інформації.

void setup() {

Serial.begin(38400); // відкриваємо порт для зв'язку з ПК для налагодження і задаємо швидкість роботи монітора порту

delay(500); // зупиняємо виконання програми на пів секунди

void(resetFunc) (void) = 0;//оголошуємо функцію reset з адресою 0 (для програмного ресету)*

analogReference(DEFAULT); // опорною напругою вважати напругу живлення мікроконтролера Vcc=5 вольт, (режим за замовчуванням, можна не задавати, на піні AREF=Vcc=5 в)

Далі налаштовуються піни мікроконтролера. Вихідні піни використовуються для живлення, а вхідні для прийняття сигналу.

pinMode(banknote_button, INPUT_PULLUP);// кнопка в купюроприймачі

pinMode(button, INPUT_PULLUP);// кнопка монетоприймача

pinMode(calibr_button, INPUT_PULLUP);// кнопка калібрування

pinMode(disp_power, OUTPUT); // живлення дисплея

pinMode(LEDpin, OUTPUT); // живлення ІЧ діода

pinMode(IRpin, OUTPUT); // живлення фотодіода

pinMode(S2, OUTPUT); // комбінація S2 і S3 активує червоні, зелені або сині фотодіоди датчика кольору

pinMode(S3, OUTPUT);

pinMode(sensor_color_power, OUTPUT); // живлення датчика кольору

pinMode(motor_power, OUTPUT); // управління живленням мотора,

// Встановлення sensorOut як входу

pinMode(sensorOut, INPUT); // вхід для вимірювання частоти з датчика кольору

Наступним кроком було подано живлення на дисплей і діоди датчика монетоприймача:

digitalWrite(disp_power, HIGH); // живлення дисплея

digitalWrite(LEDpin, HIGH); // живлення ІЧ діода

digitalWrite(IRpin, HIGH); // живлення фотодіода

Потім підключається апаратне переривання, зчитується фоновий сигнал з фотодіоду та ініціалізуються LCD-дисплей:

//апаратне переривання №0, використовується цифровий PIN D2, куди під'єднана кнопка (або контакти) прокинутися монетоприймача

attachInterrupt(0, wake_up, CHANGE);

//апаратне переривання №1, використовується цифровий PIN D3, куди під'єднана кнопка прокинутися купюроприймача

attachInterrupt(1, wake_up, CHANGE);

empty_signal = analogRead(IRsens); // зчитати порожній (опорний/фоновий) сигнал empty_signal

lcd.init(); // ініціалізація дисплея

```
lcd.backlight(); //  Вмикаємо підсвітку LCD дисплея
```

Далі відбувається обробка кнопки калібрування. Якщо її було натиснуто під час запуску, то на дисплей виводиться «Сервіс» та інформації про процес калібрування:

```
if (!digitalRead(calibr_button)) {   // якщо під час запуску натиснуто кнопку КАЛІБРУВАННЯ
  for (byte i = 0; i < coin_amount; i++) {
    coin_quantity[i] = EEPROM.readInt(i * 2); // зчитуємо кількість монет кожного номіналу, якщо вони були збережені раніше
  }
  // Виведення на дисплей «Сервіс» та інформації про процес калібрування
  lcd.clear();
  lcd.setCursor(5, 0);
  lcd.print(L"Сервіс");
  delay(500);
  reset_timer = millis(); //скинути таймер
```

Якщо кнопка калібрування натиснута більше 10 секунд, то обнуляється кількість банкнот та монет і виводиться відповідне повідомлення на дисплей «Память очищена»:

```
  while (1) {
    if (millis() - reset_timer > 10000) {// якщо кнопка все ще утримується і минуло 10 секунд,
      // то очистити кількість монет у пам'яті
      for (byte i = 0; i < coin_amount; i++) {
```

```
        coin_quantity[i] = 0; // Пам'ять ОБНУЛЕНА! Кількість монет у скарбничці всіх номіналів - нуль
        EEPROM.writeInt(i * 2, 0);  // заповнюємо всі комірки пам'яті з адреси 0 по .... нулями
      }
      // а також очистити кількість банкнот у пам'яті
      for (byte i = 0; i < banknote_amount; i++) {
        banknote_quantity[i] = 0; // Пам'ять ОБНУЛЕНА! Кількість банкнот у скарбничці всіх номіналів - нуль
        EEPROM.writeInt(60 + i * 2, 0); // заповнюємо всі комірки пам'яті з адреси 60 по .... нулями
      }
      lcd.clear();
      lcd.setCursor(1, 0);
      lcd.print(L"Память очищена");
      delay(200);
    }
```

Якщо кнопка калібрування була відпущена раніше ніж за 10 секунд після вмикання, то відбувається процес калібрування монетоприймача та виводиться відповідне повідомлення на дисплей «Калібровка»:

```
    if (digitalRead(calibr_button)) {  // якщо відпустили кнопку, перейти до калібрування
      lcd.clear();
      lcd.setCursor(0, 0);
      lcd.print(L"Калібровка");
      delay(500);
      reset_timer = millis(); // скинути таймер
      break;
```

```
  }
 }
```

Наступним кроком вимірюється максимальне та мінімальне значення фонового сигналу з фотодіода протягом 3 секунд і виведення їх на дисплей:

```
 empty_signal = analogRead(IRsens);      // зчитати порожній (опорний/фоновий) сигнал empty_signal
 last_sens_signal = empty_signal;    // значення last_sens_signal = порожньому (опорному/фоновому) сигналу empty_signal
 while (1) {// цикл на 3 секунди для заміру і виведення меж фону на дисплей
  lcd.setCursor(0, 0); lcd.print(L"Замір Фону");
  sens_signal = analogRead(IRsens); // зчитати фотодатчик
  if (sens_signal < empty_signal) {
   empty_signal = sens_signal;  // фіксуємо мінімальне значення фону
  }
  if (sens_signal > last_sens_signal) {
   last_sens_signal = sens_signal; // фіксуємо максимальне значення фону
  }
  if (millis() - reset_timer > 3000) break; // виходимо з режиму виміру фону після закінчення часу 3 сек
 } // кінець циклу заміру меж фону
```

Далі виконується виведення мінімального та максимального значень фону сенсора на LCD-дисплей і в COM-порт для подальшого аналізу. Після цього значення last_sens_signal оновлюється, відображення триває 3 секунди, і система переходить до етапу калібрування, про що повідомляється на екрані.

```
sens_signal = empty_signal  ; // виводимо межі фону
    lcd.setCursor(11, 0);
    lcd.print(sens_signal);    lcd.print("-");    lcd.print(last_sens_signal);    // виводимо на дисплей min і max значення фону
    Serial.print(empty_signal);                    Serial.print("-");
Serial.println(last_sens_signal); // виводимо в порт min і max значення фону
    last_sens_signal = empty_signal; // скидаємо змінну last_sens_signal у мінімальне значення фону
    delay(3000);// час відображення фону, після чого переходимо безпосередньо до калібрування
    lcd.clear();
    lcd.setCursor(0, 0); lcd.print(L"Калібровка");
```

Потім йде цикл самого процесу калібрування монетоприйомника. Послідовно перебираються номінали монет та проводиться калібрування монети того самого номіналу 5 разів.

```
  while (1) {
   for (byte i = 0; i < coin_amount; i++) { // послідовний перебір номіналів монет, розмір, яких калібрується
     for (byte k = 0; k < coin_amount_calibration; k++) { // повторення калібрування монети того самого номіналу (k разів)
      // виведення на дисплей
      lcd.setCursor(0, 1); lcd.print(coin_value[i]);  // відобразити ціну монети, розмір якої калібрується
      lcd.setCursor(6, 1); lcd.print(currency);    // відобразити валюту
      lcd.setCursor(14, 1); lcd.print(coin_amount_calibration - k); // відобразити кількість спроб калібрування монети того самого номіналу, що залишилася
      if (coin_amount_calibration - k == 9 ) {
```

lcd.setCursor(15, 1);

lcd.print(" "); // очистити сегмент 16 від нуля (кількість спроб - двозначне число)

}

empty_signal = analogRead(IRsens); // зчитати порожній (опорний/фоновий) сигнал empty_signal

last_sens_signal = empty_signal; // значення last_sens_signal = порожньому (опорному/фоновому) сигналу empty_signal

Наступним кроком йде цикл сканування фотодатчика й аналізується отримане значення. Якщо нічого не робити 1 хвилину, то система вийде з циклу калібрування програмним перезавантаженням.

while (1) { // нескінченний цикл сканування фотодатчика й аналізу отриманого значення

// якщо нічого не робити 60 секунд, то система вийде з циклу калібрування програмним перезавантаженням

if (millis() - reset_timer > 60000) resetFunc(); //викликаємо функцію reset

Зчитується значення з фотодіода. Якщо значення перевищило фонове на 5 та потім впало до фонового значить, що монета пролетіла. Тоді фіксується максимальне та мінімальне значення сигналу для цього номіналу монети та виводиться на дисплей.

sens_signal = analogRead(IRsens); // зчитати фотодатчик

if (sens_signal > last_sens_signal) last_sens_signal = sens_signal; // якщо поточне значення більше за попереднє, то запам'ятовуємо його

if (sens_signal - empty_signal > 5) coin_flag = true; // якщо значення перевищило фонове на 5, вважати, що пролітає монета, ставимо прапор в ІСТИНА

if (coin_flag && (abs(sens_signal - empty_signal)) < 4) { // якщо потім значення впало, повернулося майже до фонового, то монета точно пролетіла

coin_signal = last_sens_signal; // зафіксували максимальне значення сигналу монети, що пролетіла

```
if (k < 1) {
  coin_signal_max[i] = coin_signal;
  coin_signal_min[i] = coin_signal;
}
```

Далі обробляється сигнал під час прольоту монети: він оновлює мінімальне та максимальне значення сигналу для кожного номіналу, підраховує кількість монет та скидає прапорець фіксації (coin_flag). Також оновлюється таймер reset_timer для контролю часу бездіяльності, і на дисплей виводиться зафіксований сигнал поточної монети

if (coin_signal > coin_signal_max[i]) coin_signal_max[i] = coin_signal; // фіксуємо максимальне значення

if (coin_signal < coin_signal_min[i]) coin_signal_min[i] = coin_signal; // фіксуємо мінімальне значення

coin_quantity[i]++; // додаємо до загальної кількості монет даного номіналу ще одну

coin_flag = false; // переводимо прапорець фіксації прольоту монети знову в 0

reset_timer = millis(); //скинути таймер програмного reset для примусового виходу з калібрування

```
            lcd.setCursor(11,  0);  lcd.print(coin_signal);  lcd.print("      ");//
виводимо у верхній правий кут дисплея сигнал за монету
              break; // вихід з циклу
            }
          }
        }
      }
      break;// Вихід з циклу калібрування
    }
```

Для контролю, виводяться отримані статистичні значення меж діапазонів розпізнавання монет у послідовний порт.

```
    Serial.println("Statistical boundaries of coin recognition ranges");
    for (byte i = 0; i < coin_amount; i++) {
     Serial.println("coin_signal_max~min[" + String(i) + "]: " +
String(coin_signal_max[i]) + "~" + String(coin_signal_min[i]));
    }
```

Наступним кроком розраховується значення верхніх і нижніх меж діапазонів для кожного номіналу монети, розширених до їх змикання:

```
    coin_signal_max[0] = 1023; // верхня межа монети з максимальним
діаметром обмежена цифрою 1023
    coin_signal_min[coin_amount - 1] = empty_signal + 15 ; // нижня межа
монети з мінімальним діаметром на 15 одиниць вища за фонове значення
    for (byte i = 0; i < coin_amount; i++) {
     if (i > 0) {
      coin_signal_max[i] = (coin_signal_min[i - 1] - 1); //Верхня межа
діапазону номіналу монети
```

```
        }
        if (i < coin_amount - 1) { // поки змінна last_sens_signal ВІЛЬНА, використовуємо її для обчислень меж діапазонів!
          last_sens_signal = (coin_signal_min[i] - coin_signal_max[i + 1]) / 2; // величина збільшення межі діапазону, округлена до цілого значення
          coin_signal_min[i] = coin_signal_min[i] - last_sens_signal; // нижня межа діапазону номіналу монети
        }
      }
```

Далі записуємо як масив даних розраховані, розширені межі діапазонів розпізнавання монет у пам'ять і кількість монет:

```
      for (byte i = 0; i < coin_amount; i++) {
        EEPROM.updateInt(i * 2, coin_quantity[i]); // оновлює байт даних у комірці за кількість монет, якщо її старий вміст відрізняється від нового
        EEPROM.writeInt(20 + i * 2, coin_signal_min[i]); // послідовно записали мінімальне значення за кожну монету в ПАМ'ЯТЬ у клітинку з 20 по ...
        EEPROM.writeInt(40 + i * 2, coin_signal_max[i]); // послідовно записали максимальне значення за кожну монету в ПАМ'ЯТЬ у комірку з 40 по ...
      }
    } // дужка кінець циклу калібрування, кнопку «Калібрування» раніше було натиснуто
```

Так само зчитується з пам'яті верхнє і нижнє значення діапазону сигналу для кожного номіналу монет і кольору банкнот:

```
    for (byte i = 0; i < coin_amount; i++) {
```

```
coin_quantity[i] = EEPROM.readInt(i * 2);// кількість монет кожного номіналу
coin_signal_min[i] = EEPROM.readInt(20 + i * 2); // зчитує значення нижньої межі діапазону
coin_signal_max[i] = EEPROM.readInt(40 + i * 2); // зчитує значення верхньої межі діапазону
summ_money += coin_quantity[i] * coin_value[i];  // рахуємо загальну суму монет, як добуток номіналів монет на їхню кількість
}
// під час старту системи зчитати з пам'яті значення кольору банкнот для подальшої роботи, а також їхню кількість у скарбничці
for (byte i = 0; i < banknote_amount; i++) {
banknote_quantity[i] = EEPROM.readInt(60 + i * 2);// кількість банкнот кожного номіналу
//banknote_signal_R[i] = EEPROM.readInt(80+i * 2); // частота інтенсивності червоного кольору банкноти
//banknote_signal_G[i] = EEPROM.readInt(100+i * 2); // частота інтенсивності зеленого кольору банкноти
//banknote_signal_B[i] = EEPROM.readInt(120+i * 2); // частота інтенсивності синього кольору банкноти
summ_money += banknote_quantity[i] * banknote_value[i]; // до суми вже підрахованих раніше монет,
// додаємо загальну суму банкнот, як добуток номіналів банкнот на їхню кількість
}
```

Для купюроприймача вводяться еталонні значення частот для кожного номіналу банкноти, так як процес калібрування ще не написано:

banknote_signal_R[0] = 38; banknote_signal_G[0] = 54; banknote_signal_B[0] = 43; // значення частоти для кожного кольору R,G,B для 10 гривневої банкноти

banknote_signal_R[1] = 44; banknote_signal_G[1] = 45; banknote_signal_B[1] = 40; //значення частоти для кожного кольору R,G,B для 20 гривневої банкноти

banknote_signal_R[2] = 42; banknote_signal_G[2] = 46; banknote_signal_B[2] = 34;// значення частоти для кожного кольору R,G,B для 50 гривневої банкноти

banknote_signal_R[3] = 37; banknote_signal_G[3] = 42; banknote_signal_B[3] = 35; // значення частоти для кожного кольору R,G,B для 100 гривневої банкноти

banknote_signal_R[4] = 39; banknote_signal_G[4] = 47; banknote_signal_B[4] = 33; // значення частоти для кожного кольору R,G,B для 200 гривневої банкноти

banknote_signal_R[5] = 39; banknote_signal_G[5] = 48; banknote_signal_B[5] = 39; // значення частоти для кожного кольору R,G,B для 500 гривневої банкноти

banknote_signal_R[6] = 50; banknote_signal_G[6] = 53; banknote_signal_B[5] = 39; // значення частоти для кожного кольору R,G,B для 1000 гривневої банкноти

} // Кінець виконання функції void setup

Після налаштування у функції setup(), код переходить до основного циклу програми в loop(), де відбувається безперервне виконання вимірів і обробка даних.

Якщо монетниця була в режимі сну, то робимо паузу в пів секунди, обчислюємо кількість секунд з моменту запуску програми та вимірюємо внутрішню опорну температуру.

void loop() {

// якщо if sleep_flag = true, то виконуємо те, що в дужках {...} нижче

if (sleep_flag) { // прапор сну, все, що в дужках, виконується ТІЛЬКИ після прокидання або старту (RESET)

delay(500);// пауза пів секунди

// Звернення до функції TimeToString часу активної роботи МК і до функції readVcc() читання внутрішньої опорної напруги.

stime = TimeToString(millis() / 1000); //звернення до функції TimeToString з параметрами millis()/1000, сумарний час з моменту запуску програми

VCC = readVcc(); // звернення до функції readVcc()читання внутрішньої опорної напруги. Змінна VCC = значенню напруги живлення Vcc

// вимірювання напруги на акумуляторній батареї та розрахунку залишку заряду акумуляторної батареї

battery_charge = BattCharg(); //звернення до функції BattCharg()

Після цього ініціалізуємо дисплей та виводимо на нього назву монетниці, накопичену суму грошей, символ валюти та залишок заряду батареї.

lcd.init();//ініціалізація дисплея тільки після сну і старту, займає час, тому в середині самої функції виведення DisplayOutput() не застосовуємо.

DisplayOutput(); // виконується функція виведення на дисплей стандартної інформації

Тоді зчитуємо фоновий сигнал з фотодіода та змінюємо прапорець сну на false.

empty_signal = analogRead(IRsens); // зчитати датчик (зчитати порожній (опорний/фоновий) сигнал empty_signal)

sleep_flag = false; // прапорець сну

}

last_sens_signal = empty_signal; // максимальне значення сигналу останньої монети, що пролетіла, скидаємо до рівня фону empty_signal

Далі працюємо в нескінченному циклі безперервного циклічного опитування датчиків монет і купюр доки час таймера відходу в сон не вийшов і потім «break».

while (true) {

Далі йде блок сканування купюроприймача, розпізнавання і підрахунку банкнот.

Якщо вставлено купюру, тобто замкнено кнопку купюроприймача, то визначаємо колір купюри у вигляді частоти за кожен колір R, G, B.

if (!digitalRead(banknote_button)) { // якщо вставлена купюра (замкнута кнопка купюроприймача)

//Звертаємося до функції «Banknote_RGB_Frequency» вимірювання інтенсивності частот за кожен колір R,G,B

MyStruct F_RGB; // оголошуємо структуру

F_RGB = Banknote_RGB_Frequency();

Якщо купюру було вийнято, тобто кнопка розімкнута, то вимикаємо живлення з датчика кольору.

// зняти живлення з датчика кольору, якщо забрали купюру

if (digitalRead(banknote_button)) { // якщо була вийнята банкнота (розімкнулася кнопка)

digitalWrite(sensor_color_power, 0); // LOW, зняти живлення з датчика кольору

standby_timer = millis(); // скинути таймер

```
        break;// 0-й Break примусового виходу з головного циклу циклу while (true) і переходу на void loop()
      }
```

Якщо, як і раніше, вставлено купюра (замкнута кнопка купюроприймача), то визначаємо її номінал, виходячи з отриманих раніше частот за кожен колір R,G,B.

```
      while (!digitalRead(banknote_button)) {// якщо, як і раніше, вставлена купюра (замкнута кнопка купюроприймача)
        // Виходячи з отриманих раніше частот за кожен колір R,G,B, визначаємо колір банкноти та її номінал
        for (byte i = 0; i < banknote_amount; i++) {
         // обчислюємо абсолютне (модуль) значення різниці отриманих частот із нашими значеннями з пам'яті
          if (abs(redFrequency - banknote_signal_R[i]) < 7
            && abs(greenFrequency - banknote_signal_G[i]) < 7
            && abs(blueFrequency - banknote_signal_B[i]) < 7)
          { // якщо виконалися всі три умови за трьома частотами, то запам'ятовуємо номінал купюри N = i, точніше номер купюри
            N = i;
            break; // примусово виходимо з циклу for, щоб не перебирати інші комбінації
          }
        }
        break;//примусово виходимо з циклу while(!digitalRead(banknote_button)
      } // дужка while (!digitalRead(banknote_button)).
```

Якщо купюра не розпізнана, то виводимо «ERROR». Протягом часу stb_time буде прийнято кілька спроб визначити номінал купюри (при 15 сек це 3 рази) і якщо банкнота, залишаючись у купюроприймачі, не буде впізнана, то система піде в сон. Купюра протягнута не буде.

```
if (N == 10) {
  lcd.clear();
  lcd.setCursor(5, 0); lcd.print("ERROR");// виведення на дисплей
  // якщо час таймера вийшов, спимо (stb_time час бездіяльності, через який система піде в сон )
  if (millis() - standby_timer > stb_time) {
   good_night(); // звернення до функції сну
  }
```

Якщо купюра була розпізнана, то виводимо її номінал на дисплей, але поки вона не протягнута і не підрахована.

```
else  { значить, купюра, була розпізнана
  DisplayOutput();// звертаємося до функції виведення на дисплей інформація
```

Робимо паузу в 3 секунди перед увімкненням мотора, даємо час забрати банкноту, якщо не хочемо відправити її в монетницю. Якщо протягом цього часу купюру забрали, знімаємо живлення з датчика кольору.

```
  delay(3000);
  if (digitalRead(banknote_button))  { // якщо була вийнята банкнота (розімкнулася кнопка)
    digitalWrite(sensor_color_power, 0); // LOW, зняти живлення з датчика кольору
```

```
standby_timer = millis();  // скинути таймер
break;// примусовий вихід з головного циклу while (true) і переходу на void loop()
}
```

Якщо не забрали банкноту, то подаємо живлення на датчик кольору та увімкаємо мотор для протягання купюри в монетницю.

```
digitalWrite(sensor_color_power, HIGH); // подати живлення на датчик кольору
digitalWrite(motor_power, HIGH); // увімкнути мотор для протягання купюри в скарбничку
```

Якщо вставлена банкнота розпізнана і її не забрали з купюроприймача, потрапляємо в цикл while (1). Якщо кнопка купюроприймача розімкнулася, перевіряємо, куди пройшла купюра.

```
while (1) {
if (digitalRead(banknote_button)) { //Якщо кнопка розімкнулася, перевіряємо, куди пройшла купюра. Всередину скарбнички чи ні, її вилучили.
if (pulseIn(sensorOut, LOW) < empty_signal_B - 30) {//якщо кнопка розімкнулася і частота з датчика кольору менша за фонове значення (empty_signal_B)
```

Якщо купюра точно в монетниці, рахуємо її і виводимо суму на дисплей.

```
summ_money += banknote_value[N];  // до суми грошей у скарбничці додаємо ціну банкноти
banknote_quantity[N]++;  // для розпізнаного номера купюри додаємо кількість
```

DisplayOutput();// звертаємося до функції виведення на дисплей інформація.(умови «N»!= 10(не дорівнює))

Даємо мотору із запасом попрацювати ще 1,5 секунди та вимикаємо його з датчиком кольору.

delay(1500);

digitalWrite(motor_power, LOW); // вимикаємо мотор протягання купюри

digitalWrite(sensor_color_power, LOW); // зняти живлення з датчика кольору

break;//виходимо примусово з циклу while (1), оскільки купюра протягнута в скарбничку і врахована в загальній сумі грошей.

} //дужка умови if (pulseIn(sensorOut, LOW) < empty_signal_B - 30)

Якщо купюру забрали, під датчиком немає банкноти, чорний колір та кнопка розімкнута, то не зараховуємо її. Вимикаємо мотор з датчиком кольору. Також чекаємо пів секунди, як захист від брязкоту під час вилучення купюри.

digitalWrite(motor_power, LOW); // вимикаємо мотор протягання купюри

digitalWrite(sensor_color_power, LOW); // зняти живлення з датчика кольору

delay(500);//пауза, як захист від брязкоту під час вилучення купюри, щоб не спрацювало помилково if (!digitalRead(banknote_button))

break;//виходимо примусово з циклу while (1), оскільки купюру забрали, і вона не врахована! Купюроприймач порожній.

}

if (millis() - standby_timer > stb_time + 10000) { //Тут це потрібно, щоб мотор не працював вічно, якщо ролики не захопили купюру.

good_night(); // звернення до функції сну. Після прокидання, програма продовжить працювати з цього місця.

N = 10;

break; // після прокидання, break використовується для примусового виходу з циклу while(1)

}

} //дужка циклу while (1)

standby_timer = millis(); // скинути таймер

} // дужка else {купюра була розпізнана, варіанти: (1) і підрахована; (2) але, її забрали назад і вона не врахована}

// варіант (3) - купюра не розпізнана, «ПОМИЛКА»=«ERROR», пропустили else { ... }, і потрапляємо на 3-й Break

break;// 3-й Break використовується для примусового виходу з ГОЛОВНОГО циклу while (true) і переходу на void loop()

} // дужка якщо банкнота вставлена if (!digitalRead(banknote_button)) {

Далі перейдемо до блоку опитування датчика монет, фіксації прольоту та підрахунку монет. Алгоритм такий самий, як під час калібрування.

Зчитуємо сигнал з фотодіода, якщо поточне значення більше за попереднє та перевищило фоновий сигнал на 5 та потім впало майже до фонового, вважаємо що монета пролетіла.

sens_signal = analogRead(IRsens); // зчитати датчик

if (sens_signal > last_sens_signal) last_sens_signal = sens_signal; // якщо поточне значення більше за попереднє

if (sens_signal - empty_signal > 5) coin_flag = true; // якщо значення перевищило фонове на 5, вважати, що пролітає монета, ставимо прапор в ІСТИНА

if (coin_flag && (abs(sens_signal - empty_signal)) < 4) { // якщо значення потім упало, повернулося майже до фонового, то монета точно пролетіла

Далі починаємо порівнювати сигнал із діапазонами монет, що зберігаються в пам'яті.

```
for (byte i = 0; i < coin_amount; i++) {
 if ( last_sens_signal >= coin_signal_min[i] && last_sens_signal <= coin_signal_max[i]) {  // !!! і ось тут якщо сигнал потрапляє в діапазон, то вважаємо монетку розпізнаною
 summ_money += coin_value[i];// до суми тупо додаємо ціну монетки
 coin_quantity[i]++;  // для розпізнаного номера монетки додаємо кількість
 N = i;// змінну N використовуємо для запам'ятовування порядкового номера номіналу монети
```

Виводимо на дисплей номінал монети, що пролетіла, накопичену суми грошей, символ валюти та залишок заряду батареї.

```
 DisplayOutput();//виведення на дисплей
 break;// примусовий вихід з циклу, for (byte i = 0; i < coin_amount; i++). Так як монету розпізнано, не треба перебирати інші межі.
 }  дужка циклу for (byte i = 0; i < coin_amount; i++)
} // кінець циклу for (byte i = 0; i < coin_amount; i++) перебору і порівняння сигналу монетою, що пролетіла, зі збереженими в пам'яті
 coin_flag = false; // переводимо прапорець фіксації прольоту монети знову в 0 для фіксації нового прольоту монети
 standby_timer = millis();  // скинути таймер, millis() кількість мілісекунд з моменту початку виконання програми
 break; примусовий вихід з циклу while (true) і переходу на void loop()
}
```

if (millis() - standby_timer >= displ_coin_value_time && millis() - standby_timer < displ_coin_value_time + 3) {

// Звертаємося до функції виведення на дисплей інформації: назви скарбнички, накопиченої суми грошей, символ валюти, залишок заряду батареї

DisplayOutput();

}

Якщо нічого не робили, час таймера вийшов, спимо (stb_time час бездіяльності, через який система піде в сон).

if (millis() - standby_timer > stb_time) {

good_night(); // звернення до функції сну. Після прокидання програма продовжить працювати з цього місця

break; // примусовий вихід з циклу while (true) і переходу на void loop()

}

Наступним кроком реалізовано виведення статистики за кількістю монет кожного номіналу і виведення службової інформації. Для цього перевіряємо чи натиснута кнопка прокинутись.

while (!digitalRead(button)) { // натиснута кнопка «прокинутись»

Якщо кнопка прокинутись затиснута більше 10 секунд, то виводимо статистику.

if (millis() - standby_timer > 10000) { // якщо контакти «прокинутись» замкнуті > 10 секунд, то виведення статистики

Якщо одночасно натиснуті кнопки прокинутись і калібрування, то калібруємо батарею, про що виводимо відповідну інформацію.

```
if (!digitalRead(calibr_button)) {
  lcd.clear();
  lcd.setCursor(0, 0); lcd.print(L"Калібрування");
  lcd.setCursor(0, 1); lcd.print(L"батареі");
  // напруга батареї під час повної зарядки мінус 1%, приймається, як верхня межа, що відповідає 100% рівню заряду
  max_battery_voltage = battery_voltage - (battery_voltage - min_battery_voltage) / 100;
  delay(3000);
  lcd.clear();
  lcd.setCursor(0, 0); lcd.print(L"Завершена");
  lcd.setCursor(0, 1); lcd.print("MaxBattV="); lcd.print(String(max_battery_voltage, 3) + " V");// напруга батареї верхня межа
  standby_timer = standby_timer - stb_time; // щоб після наступної паузи delay (5000) відразу з калібрування йшов у сон
  delay(5000);
  break;
 }
   lcd.clear();
```

Перше виведемо кількість банкнот в монетниці. Виведемо номінали банкнот зверху і статистику кількості банкнот знизу.

Так як кількість номіналів банкнот більша 4, виведення буде в два етапи по 5 секунд.

```
for (byte i = 0; i < 4; i++) { // відобразити на дисплеї статистику для перших 4-х банкнот
  lcd.setCursor(i * 4, 0); lcd.print(banknote_value[i], 0); // зверху номінал банкноти цілі числа
```

```
        lcd.setCursor(i * 4, 1); lcd.print(banknote_quantity[i]); // знизу їхня кількість
      }
      delay(5000);
      if (banknote_amount > 4) { // якщо число номіналів банкнот більше ніж 4, виведення другим етапом інших
        lcd.clear();
        for (byte i = 4; i < banknote_amount; i++) { // відобразити на дисплеї статистику для інших монет
          lcd.setCursor(i * 5 - 20, 0); lcd.print(banknote_value[i], 0); // зверху номінал банкноти цілі числа
          lcd.setCursor(i * 5 - 20, 1); lcd.print(banknote_quantity[i]); // знизу їхня кількість
        }
      }
      delay(5000);
      lcd.clear();
```

Далі виведемо кількість монет в скарбничці. Виведемо номінали монет зверху і статистику кількості монет знизу.

Так як кількість номіналів монет більша 4, виведення буде в два етапи по 5 секунд.

```
      for (byte i = 0; i < 4; i++) { // відобразити на дисплеї статистику для перших 4-х монет
        lcd.setCursor(i * 4, 0); lcd.print(coin_value[i], 1); // зверху номінал монети (округлений до десятих)
        lcd.setCursor(i * 4, 1); lcd.print(coin_quantity[i]); // знизу їхня кількість
      }
```

```
delay(5000);
if (coin_amount > 4) { // якщо число номіналів монет більше ніж 4, виведення другим етапом решти
  lcd.clear();
  for (byte i = 4; i < coin_amount; i++) { // відобразити на дисплеї статистику для інших монет
    lcd.setCursor(i * 5 - 20, 0); lcd.print(coin_value[i], 2); // зверху номінал монети (округлений до десятих)
    lcd.setCursor(i * 5 - 20, 1); lcd.print(coin_quantity[i]); // знизу їхня кількість
  }
}
delay(4000);
standby_timer = millis() ;  //скинути таймер
```

Якщо кнопка «прокинутися», як і раніше, була натиснута під час виведення статистики монет, то виводимо на дисплей службову інформацію: версію прошивки, напругу живлення мікроконтролера, напруги на батареї, сумарний час роботи МК, рівень фонового сигналу та результати калібрування монет.

```
while (!digitalRead(button)) {
  stime = TimeToString(millis() / 1000); //звернення до функції TimeToString з параметрами millis()/1000, сумарний час з моменту запуску програми
  VCC = readVcc(); // Звернення до функції readVcc()читання внутрішньої опорної напруги. Змінна VCC = значенню напруги живлення Vcc
```

Вимірюємо напругу на акумуляторній батареї та розраховуємо залишок заряду акумуляторної батареї, виводимо цю інформацію на дисплей.

battery_charge = BattCharg(); //звернення до функції BattCharg()

lcd.clear();

lcd.setCursor(0, 0); lcd.print("Version V3.1"); //версія прошивки

lcd.setCursor(0, 1); lcd.print("time="); lcd.print(stime); // сумарний час роботи МК

delay(5000);

lcd.setCursor(0, 0); lcd.print("Vcc="); lcd.print(VCC, 3); lcd.print(" volt "); //вивід напруги живлення Ардуіно на піні Vcc

lcd.setCursor(0, 1); lcd.print("Vbatt="); lcd.print(battery_voltage, 3); lcd.print(" volt "); // виведення напруги на батареї Vbatt

delay(5000);

lcd.setCursor(0, 0); lcd.print("BatCharge="); lcd.print(battery_charge); lcd.print("%"); // відсоток рівня заряду батареї

Також виводимо на дисплей рівень фонового сигналу.

lcd.setCursor(0, 1); lcd.print("empty_signal="); lcd.print(String(empty_signal)); lcd.print(" "); // виведення рівня фонового сигналу

delay(5000);

Далі виводимо результати калібрування монет на дисплей.

for (byte i = 0; i < coin_amount; i += 2) { // відобразяться на дисплеї межі діапазонів розпізнавання для кожного номіналу монет

lcd.clear();

lcd.setCursor(0, 0);

lcd.print(coin_value[i]); lcd.print(":"); // номінал монети

lcd.print(coin_signal_max[i]); lcd.print("~"); lcd.print(coin_signal_min[i]); // та її діапазон

```
        lcd.setCursor(0, 1);
        if (i < coin_amount - 1) {
          lcd.print(coin_value[i + 1]); lcd.print(":"); // номінал монети
          lcd.print(coin_signal_max[i + 1]); lcd.print("~");
lcd.print(coin_signal_min[i + 1]); // та її діапазон
        }
        delay(5000); // з паузою в 5 секунд, послідовно відображаються
результати калібрування за кожну монету
       standby_timer = millis() ;  //скинути таймер
       break;
     } // дужка якщо кнопка «прокинутися» як і раніше натиснута
   } // дужка if (millis() - standby_timer > 5000) (коли якщо контакти
«прокинутися» замкнуті > 5 секунд, то виведення статистики )
    } // дужка while (!digitalRead(button)) (коли натиснуто кнопку
«прокинутися» )
  } // дужка циклу while (true) (безперервного циклічного сканування
датчиків монет і купюр)
} //дужка циклу void loop() { }
```

В кінці коду реалізуються використані функції.

Першою реалізовується функція для сну. Функція good_night() реалізує перехід мікроконтролера в режим низького енергоспоживання (сплячий режим). Перед цим виконуються операції збереження даних у постійну пам’ять (EEPROM) і вимкнення енергоспоживаючих компонентів.

```
void good_night() {
  // перед тим як піти спати, записуємо в EEPROM нові отримані дані
про кількість монет за адресами починаючи з 0-го
  for (byte i = 0; i < coin_amount; i++) {
```

```
    EEPROM.updateInt(i * 2, coin_quantity[i]); // EEPROM.update обновляет байт данных
    // та же запись EEPROM.write, но лучше, заменяет значение в ячейке, если её старое содержимое отличается от нового
    }
    for (byte i = 0; i < banknote_amount; i++) {
    EEPROM.updateInt(60 + i * 2, banknote_quantity[i]); // EEPROM.update оновлює байт даних
    }
    sleep_flag = true;
    // вимкнути живлення з дисплея і всіх датчиків
    digitalWrite(disp_power, LOW);
    digitalWrite(LEDpin, LOW);
    digitalWrite(IRpin, LOW);
    digitalWrite(motor_power, LOW);
    digitalWrite(sensor_color_power, LOW);
    delay(100);
    LowPower.powerDown(SLEEP_FOREVER, ADC_OFF, BOD_OFF);
  }
```

Наступна функція wake_up() відповідає за повернення мікроконтролера до активного стану після пробудження зі сплячого режиму (ця функція - обробник переривання). Спрацьовує під час замикання кнопок купюроприймача і монетоприймача (замикання контактів) «уві сні», а також в активному режимі роботи скарбнички.

```
  void wake_up() {
    // повертаємо живлення на дисплей і датчик
    digitalWrite(disp_power, HIGH);
    digitalWrite(LEDpin, HIGH);
```

```
  digitalWrite(IRpin, HIGH);
  standby_timer = millis() - (displ_coin_value_time + 4) ;   // не обнуляємо таймер, а скидаємо таймер, до значення
  // «таймера тривалості часу виведення номіналу монет», щоб не було повторного виведення інформації на дисплей після прокидання
  }
```

Функція readVcc() обчислює напругу живлення (Vcc) мікроконтролера, використовуючи внутрішню опорну напругу 1.1 В як еталон.

```
  float readVcc() {
    byte i;
    float result = 0.0;
    float tmp = 0.0;
    for (i = 0; i < 5; i++) {
  #if defined(__AVR_ATmega32U4__) || defined(__AVR_ATmega1280__) || defined(__AVR_ATmega2560__)
      ADMUX = _BV(REFS0) | _BV(MUX4) | _BV(MUX3) | _BV(MUX2) | _BV(MUX1);
  #elif defined (__AVR_ATtiny24__) || defined(__AVR_ATtiny44__) || defined(__AVR_ATtiny84__)
      ADMUX = _BV(MUX5) | _BV(MUX0);
  #elif defined (__AVR_ATtiny25__) || defined(__AVR_ATtiny45__) || defined(__AVR_ATtiny85__)
      ADMUX = _BV(MUX3) | _BV(MUX2);
  #else
      // works on an Arduino 168 or 328
      ADMUX = _BV(REFS0) | _BV(MUX3) | _BV(MUX2) | _BV(MUX1);
  #endif
      delay(3); // Wait for Vref to settle
```

```
    ADCSRA |= _BV(ADSC); // Start conversion
    while (bit_is_set(ADCSRA, ADSC)); // measuring
    uint8_t low  = ADCL; // must read ADCL first - it then locks ADCH
    uint8_t high = ADCH; // unlocks both
    tmp = (high << 8) | low;
    tmp = (vcc_const * 1023.0) / tmp;
    result = result + tmp;
    delay(5);
  }
  result = result / 5;
  return result;
}
```

Функція BattCharg() обчислює залишок заряду акумуляторної батареї, виражений у відсотках, на основі виміряної напруги. Вона включає кілька етапів, таких як усереднення показань АЦП, обчислення напруги на батареї та інтерполяція для визначення заряду.

```
float BattCharg() {
  // зчитуємо точне значення напруги з піна A1 5 разів, де знаходиться наш вольтметр з дільником напруги
  battery_voltage = 0.0;  // напруги на акумуляторній батареї
  for (byte i = 0; i < 5; i++) {
    battery_voltage = battery_voltage + analogRead(VoltmetrBatt); // сума 5 вимірів
    delay(10);
  }
```

Розраховуємо напругу на батареї у вольтах як середньоарифметичне 5 вимірювань з урахуванням опорного VCC і резисторів R3 і R4.

*battery_voltage = ((battery_voltage / 5) * VCC / 1024) / (r4 / (r3 + r4)); // підсумкова формула розрахунку напруги батареї вольт*

Відображаємо залишку заряду у відсотках від повної ємності. Інтерполюємо вручну за графіком розряду літієвого акумулятора.

```
battery_charge = battery_voltage * 100;
if (battery_charge > 410)
  battery_charge = map(battery_charge, max_battery_voltage * 100, 410, 100, 95);
else if ((battery_charge <= 410) && (battery_charge > 405) )
  battery_charge = map(battery_charge, 410, 405, 95, 76);
else if ((battery_charge <= 405) && (battery_charge > 360) )
  battery_charge = map(battery_charge, 405, 360, 76, 29);
else if ((battery_charge <= 360) && (battery_charge > 340) )
  battery_charge = map(battery_charge, 360, 340, 29, 16);
else if ((battery_charge <= 340) && (battery_charge > 310) )
  battery_charge = map(battery_charge, 340, 310, 16, 5);
else if (battery_charge <= 310)
  battery_charge = map(battery_charge, 310, min_battery_voltage * 100, 5, 0);
battery_charge = constrain(battery_charge, 0, 100); //заряд батареї визначений областю допустимих значень 0~100% constrain
return battery_charge;// «return» припиняє обчислення у функції BattCharg() і повертає значення battery_charge у місце виклику функції
}
```

Функція char * TimeToString(unsigned long t) розраховує сумарний час активної роботи МК та перетворює його в зручний формат години, хвилини, секунди.

```
// t is time in seconds = millis()/1000; Час t у секундах від початку початку запуску програми
char * TimeToString(unsigned long t)
{
 static char str[12];
 long h = t / 3600;
 t = t % 3600; //  % повертає залишок від ділення одного цілого (int) операнда на інший
 int m = t / 60;
 int s = t % 60;
 sprintf(str, "%04ld:%02d:%02d", h, m, s); // виведення сумарного часу у форматі (h, m, s)
 return str;// повертає stime, припиняє обчислення у функції та повертає значення (str) з перерваної функції в викликаючу TimeToString(millis()/1000)=str
}
```

Функція DisplayOutput() виводить на дисплей назву скарбнички або номінал розпізнаної монети/банкноти та суму накопичених коштів, символу валюти, залишку заряду батареї.

```
void DisplayOutput()
{
 lcd.clear();
 if (N == 10) { // стан прапорця coin_flag = false
  lcd.setCursor(1, 0); lcd.print(L"Smart копілка"); //
 }
```

// якщо «N» не дорівнює 10 і прапорець фіксації прольоту монети в скарбничку - не істина, а false

else if (N != 10 && !coin_flag) { // якщо N стала відмінна від 10 і coin_flag = false, то було впізнано купюру!

lcd.setCursor(3, 0); lcd.print(print_banknote_value[N]);// вивести номінал розпізнаної банкноти

lcd.setCursor(3, 0);

if (N == 0 && print_banknote_value[N] == "RU") lcd.print(L"10 гривень");

if (N == 1 && print_banknote_value[N] == "RU") lcd.print(L"20 гривень");

if (N == 2 && print_banknote_value[N] == "RU") lcd.print(L"50 гривень");

if (N == 3 && print_banknote_value[N] == "RU") lcd.print(L"100 гривень");

if (N == 4 && print_banknote_value[N] == "RU") lcd.print(L"200 гривень");

if (N == 5 && print_banknote_value[N] == "RU") lcd.print(L"500 гривень");

if (N == 6 && print_banknote_value[N] == "RU") lcd.print(L"1000 гривень");

}

Якщо пролетіла монета і вона була розпізнана, виводимо її номінал на дисплей.

else {// в іншому випадку, пролетіла монета і розпізнана (coin_flag = true), виводимо її номінал на дисплей.

lcd.setCursor(4, 0); lcd.print(print_coin_value[N]);

lcd.setCursor(4, 0);

```
    if (N == 0 && print_banknote_value[N] == "RU") lcd.print(L"1 гривня");
    if (N == 1 && print_banknote_value[N] == "RU") lcd.print(L"10 гривень");
    if (N == 2 && print_banknote_value[N] == "RU") lcd.print(L"50 копійок");
    if (N == 3 && print_banknote_value[N] == "RU") lcd.print(L"5 гривень");
    if (N == 4 && print_banknote_value[N] == "RU") lcd.print(L"2 гривні");
    if (N == 5 && print_banknote_value[N] == "RU") lcd.print(L"1 гривня");
    // кінець
   }
   lcd.setCursor(0, 1); lcd.print(summ_money); //сума монет, якщо потрібно відобразити цілі числа, то lcd.print(summ_money,0) або ((int)summ_money)
   lcd.print(" " + currency); // виведення символів валюти
   if (round(battery_charge) < 10) {
    lcd.setCursor(14, 1);
   }
   else {
    lcd.setCursor(13, 1);
   }
   if (round(battery_charge) > 99) {
    lcd.setCursor(12, 1);
   }
   lcd.print(battery_charge, 0); lcd.print(L"%"); // відсоток заряду батареї
  }
```

Функція Banknote_RGB_Frequency призначена для визначення інтенсивності трьох основних кольорів (червоного, зеленого і синього) за допомогою сенсора кольору.

```
MyStruct Banknote_RGB_Frequency() {
```

digitalWrite(sensor_color_power, HIGH); // подати живлення на датчик кольору

delay(3000);// пауза, даємо час подати купюру до упору, до валиків.

while (1) { // цикл визначення частоти за кожен колір

// цикл виконається ОДИН раз, оскільки в кінці його стоїть оператор «break»

// зациклиться з безперервним виведенням частот, якщо буде натиснута кнопка «калібрування»

redFrequency = 0; greenFrequency = 0; blueFrequency = 0; // обнуляємо частоту

for (byte i = 0; i < 5; i++) { //зчитує 5 разів частоту інтенсивності кожного кольору

Встановленням S2 і S3 низького стану активуємо червоні фотодіоди, щоб отримати дані для червоного кольору.

digitalWrite(S2, LOW);

digitalWrite(S3, LOW);

// Читання вихідної частоти для червоного кольору

redFrequency = redFrequency + pulseIn(sensorOut, LOW); // сума 5-ти вимірів

delay(100);// пауза

Встановленням S2 і S3 високого стану активуємо зелені фотодіоди, щоб отримати дані для зеленого кольору.

digitalWrite(S2, HIGH);

digitalWrite(S3, HIGH);

// Читання вихідної частоти для зеленого кольору

greenFrequency = greenFrequency + pulseIn(sensorOut, LOW); //

```
delay(100);
```

Встановленням S2 низького стану і S3 високого стану активуємо сині фотодіоди, щоб отримати дані для синього кольору.

```
digitalWrite(S2, LOW);
digitalWrite(S3, HIGH);
// Читання вихідної частоти для синього кольору
blueFrequency = blueFrequency + pulseIn(sensorOut, LOW); //
delay(100);
} //дужка циклу for
redFrequency = redFrequency  / 5 ; // середнє арифметичне червоного кольору
greenFrequency = greenFrequency  / 5 ; // середнє арифметичне зеленого кольору
blueFrequency = blueFrequency  / 5 ; // середнє арифметичне синього кольору
```

Так як алгоритм калібрування купюроприймача ще не написано, то для його налаштування потрібно вставити банкноту і натиснути кнопку калібрування. Після цього на дисплей виводяться виміряні частоти за кожен фільтр *R,G,B* , які необхідно буде ввести в програму як еталонні значення для кожної банкноти.

```
if (digitalRead(calibr_button) or digitalRead(banknote_button)) break ;
// Для контролю, якщо натиснуто кнопку «калібрування», виводимо постійно значення частоти червоного, зеленого і синього кольору
lcd.clear(); lcd.setCursor(0, 0);
lcd.print("R" + String(redFrequency) + " G" + String(greenFrequency) + " B" + String(blueFrequency));
```

blueFrequency = redFrequency + greenFrequency + blueFrequency ; // і виводимо контрольну суму

lcd.setCursor(0, 1); lcd.print("S" + String(blueFrequency));

standby_timer = millis(); //скидаємо таймер, щоб, якщо до цього довго тримали кнопку «калібрування», не зупинився мотор,

//зі спливом часу бездіяльності standby_timer(таймера сну), коли він ще протягує купюру в скарбничку

} //дужка циклу while (1) визначення інтенсивності частоти за кожен колір R,G,B

return (MyStruct) {

redFrequency, greenFrequency, blueFrequency

};

}

Увесь код програмного забезпечення наведено у додатку А.

3.2 Перевірка працездатності розробленої системи

Після підключення всіх датчиків, підключення живлення, та завантаження програми було прийняте рішення створення корпусу для монетниці, який складається з двох частин: монетоприймача (рис 3.1) і купюроприймача (рис 3.2 – 3.3).

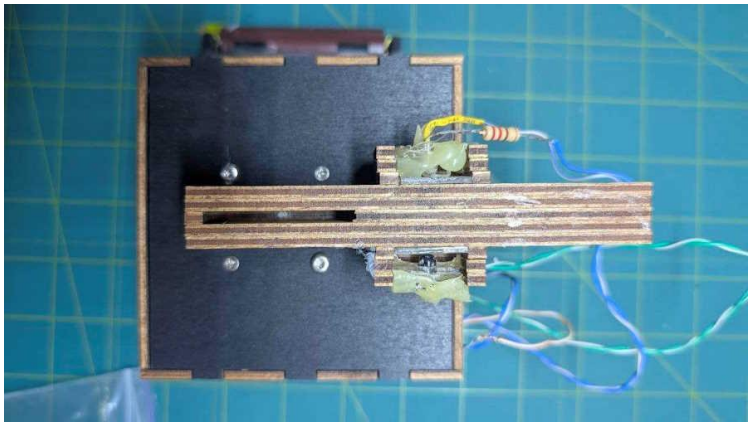


Рисунок 3.1 – Монетоприймач

Рисунок 3.2 – Купюроприймач

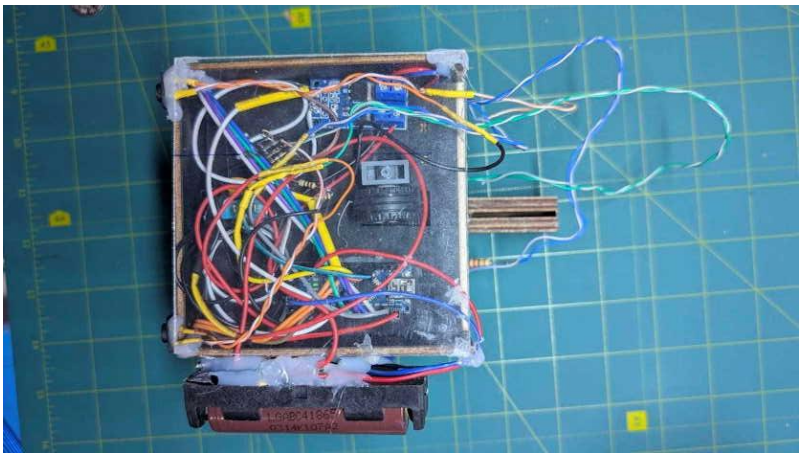


Рисунок 3.3 – Нижня частина купюроприймача

В монетоприймачі (рис.3.1) знаходиться: інфрачервоний світлодіод, фотодіод та резистор на 220 Ом.

В купюроприймачі (рис.3.2) знаходиться: датчик кольору TCS230, мікровимикач легкого ходу, мікродвигун n20 з колесом та LCD дисплей.

Знизу монетниці (рис.3.3) розмістилися: плата Arduino Nano, UPS 5v з акамулятором 18650, транзистор, резистори та інші дрібні компоненти.

Перед запуском необхідно провести налаштування в скетчі програми. Потрібно ввести кількість номіналів монет та банкнот, вартість номіналу монет в порядку зменшення їх діаметра, кількість монет одного номіналу для

калібрування, вартість номіналу банкнот, валюту, час відображення на дисплеї монети яка пролетіла та час бездіяльності через який система піде в сон.

На рисунку 3.4 зображено налаштування монетниці для української гривні.

```
//-------НАЛАШТУВАННЯ---------
#define coin_amount 6    // число номіналів монет, які потрібно розпізнати
#define banknote_amount 7    // число номіналів банкнот, які потрібно розпізнати
#define coin_amount_calibration 5  //  число монет одного номіналу для калібрування
float coin_value[coin_amount] = {1.0, 10.0, 0.50, 5.0, 2.0, 1.0 };  // вартість, номінал монет у порядку зменшення діаметра
float banknote_value[banknote_amount] = {10.0, 20.0, 50.0, 100.0, 200.0, 500.0, 1000.0};  // номінали банкнот
String currency = "UAH"; // валюта UAH - Українська гривня
int displ_coin_value_time = 2000; // час відображення на дисплеї номіналу монети, що пролетіла (мілісекунди)
int stb_time = 30000;    // час бездіяльності (мілісекунди), через який система піде в сон
```

Рисунок 3.4 – Налаштування монетниці для української гривні

Після цього завантажуємо оновлений скетч в плату Arduino.

Під час першого першого запуску необхідно провести калібрування монетоприймача та купюроприймача для точно визначення номіналів монет та банкнот відповідно.

Для цього затискаємо ліву кнопку «Калібрування» та перезавантажуємо Arduino. На дисплеї з’явиться повідомлення «Сервіс» потім «Калібровка» (рис.3.5 - 3.6).

Рисунок 3.5 – Повідомлення «Сервіс»

Рисунок 3.6 – Повідомлення «Калібровка»

Потім відбувається замір пустого сигналу з монетоприймача (рис.3.7).

Рисунок 3.7 – Замір фону

Наступним кроком необхідно вставляти монетку певного номіналу стільки раз, скільки написано в правому куті дисплея (рис.3.8).

Рисунок 3.8 – Калібрування монети номіналом 1 грн

Після того як монету кинуто, в правому верхньому куті дисплея виводиться інформація про сигнал монети, а нижче кількість монет цього номіналу, яку необхідно ще вставити (рис.3.9).

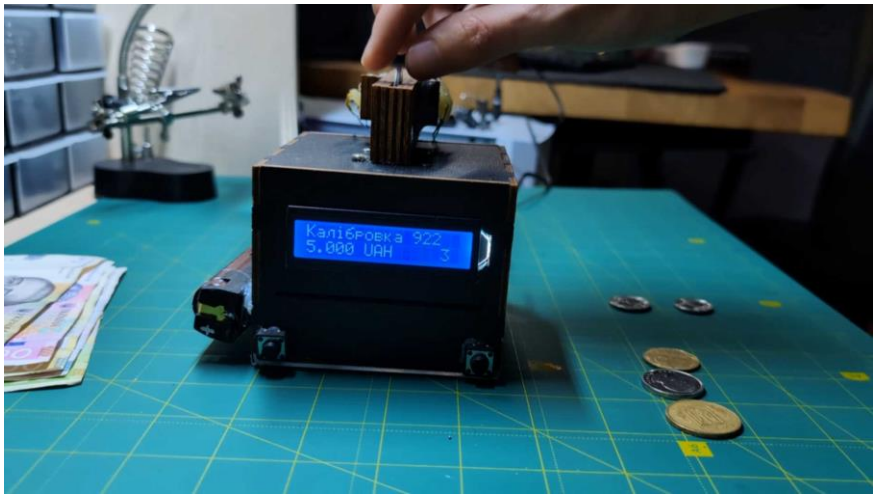


Рисунок 3.9 – Калібрування монети номіналом 5 грн

Коли всі монети прокалібровані в монетоприймачі, на дисплей виводиться назва монетниці «Smart копілка», суму монет та заряд акумулятора.

В монетниці після калібрування повинно бути 97,50 грн (рис.3.10):

$$S = 1\text{грн} \cdot 5 + 10\text{грн} \cdot 5 + 50\text{коп} \cdot 5 + 5\text{грн} \cdot 5 + 2\text{грн} \cdot 5 + 1\text{грн} \cdot 5 = 97{,}50\ \text{грн}$$

Рисунок 3.10 – Головний екран

Далі переходимо до калібрування купюроприймача. Для цього вставляємо в нього банкноту та затискаємо ліву кнопку «Калібрування». На дисплей виводиться значення частот за кожен колір R,G,B (рис.3.11).

Рисунок 3.11 – Калібрування купюроприймача

Так як автоматичне калібрування, як з монетоприймачем, ще не реалізоване, то необхідно ввести ці значення в скетч програми в розділ «Еталонні значення частот для кожного номіналу банкноти» (рис.3.12).

```
// ЕТАЛОННІ ЗНАЧЕННЯ ЧАСТОТ ДЛЯ КОЖНОГО НОМІНАЛУ БАНКНОТИ
banknote_signal_R[0] = 28; banknote_signal_G[0] = 21; banknote_signal_B[0] = 28; // значення частот за кожен колір R,G,B для 10 гривневої банкноти
banknote_signal_R[1] = 49; banknote_signal_G[1] = 50; banknote_signal_B[1] = 40; // значення частот за кожен колір R,G,B для 20 гривневої банкноти
banknote_signal_R[2] = 49; banknote_signal_G[2] = 58; banknote_signal_B[2] = 41; // значення частот за кожен колір R,G,B для 50 гривневої банкноти
banknote_signal_R[3] = 45; banknote_signal_G[3] = 52; banknote_signal_B[3] = 45; // значення частот за кожен колір R,G,B для 100 гривневої банкноти
banknote_signal_R[4] = 43; banknote_signal_G[4] = 52; banknote_signal_B[4] = 35; // значення частот за кожен колір R,G,B для 200 гривневої банкноти
banknote_signal_R[5] = 42; banknote_signal_G[5] = 49; banknote_signal_B[5] = 41; // значення частот за кожен колір R,G,B для 500 гривневої банкноти
banknote_signal_R[6] = 52; banknote_signal_G[6] = 50; banknote_signal_B[6] = 35; // значення частот за кожен колір R,G,B  для 1000 гривневої банкноти
```

Рисунок 3.12 – Еталонні значення частот для кожного номіналу банкноти

Тепер перевіримо правильність визначення номіналу монет.

Вставимо в монетоприймач 1 гривню. Як бачимо з рисунку 3.13 монениця розпізнала цю монету як 1 грн та добавило її до загальної суми. Також перевірено розпізнавання і з іншими монетами.

Рисунок 3.13 – Розпізнавання номіналу монет

Далі перевіримо правильність визначення номіналу купюр.

Вставимо в купюроприймач банкноту 500 гривень. Як бачимо з рисунку 3.14 монетниця розпізнала її як 500 грн, але не добавила її до загальної суми, для того щоб протягом 3 секунд її можна було забрати, якщо не хочемо щоб вона потрапила в монетницю.

Рисунок 3.14 – Розпізнавання номіналу купюр

Якщо купюру не було забрано, то вмикається мікродвигун та затягує її в середину монетниці та працює ще 1,5 секунди, щоб точно її протягнути. Після цього номінал банкноти зараховується до загальної суми (рис.3.15).

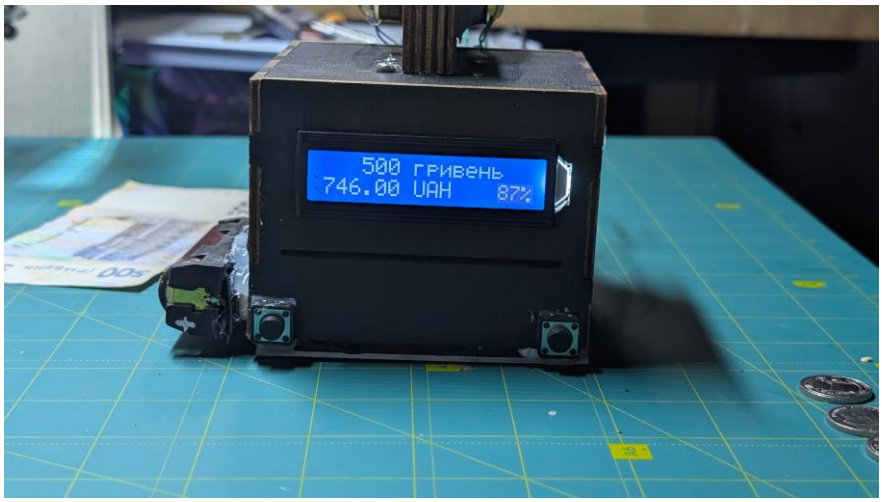


Рисунок 3.15 – Зарахування 500 грн

Якщо ж номінал купюри не був розпізнаний, то на дисплей виводиться повідомлення «ERROR» (рис.3.16).

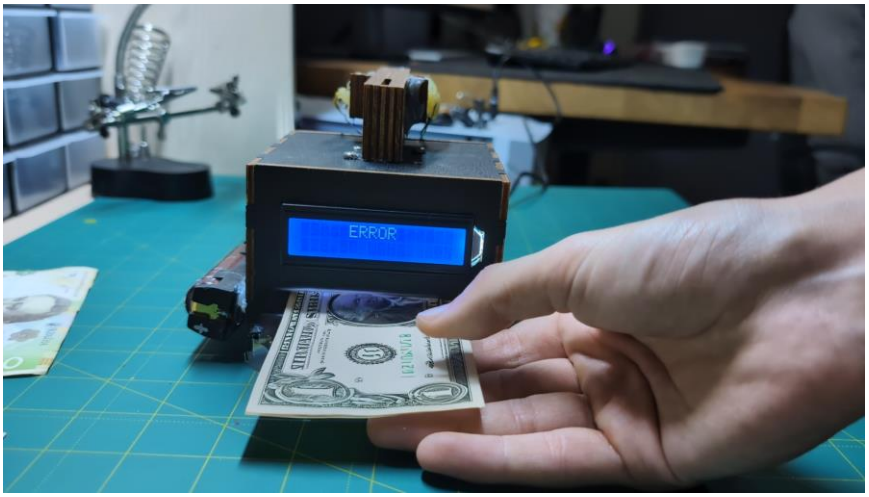


Рисунок 3.16 – Повідомлення “ERROR”

Після 30 секунд бездіяльності мононеиця йде в сон. Для того щоб вона прокинулась і перейшла на головний екран необхідно натиснути праву кнопку «Прокинутись» (рис.3.17).

Рисунок 3.17 – Система після пробудження

Також можна переглянути статистику кількості монет та купюр кожного номіналу та іншу службову інформацію.

Для цього потрібно натиснути кнопку “Прокинутись” та тримати її натиснутою ще 10 секунд, після цього спочатку на дисплей виводиться статистика кількості банкнот (рис.3.18), потім монет (рис.3.19).

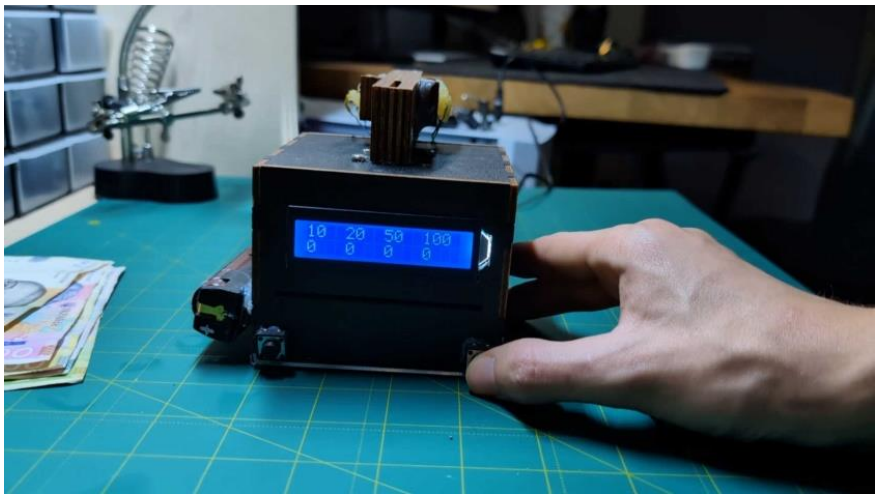


Рисунок 3.18 – Статистика кількості купюр від 10 грн до 100 грн

Рисунок 3.19 – Статистика кількості монет від 2 грн до 1 грн

Якщо після виведення статистики кнопка «Прокинутись» залишається натиснутою, на дисплей почергово виводиться додаткова інформація. Спочатку відображається версія прошивки та загальний час роботи пристрою (рис.3.20).

Рисунок 3.20 – Версія прошивки та час роботи пристрою

Далі показуються напруга живлення Arduino на піні VCC та напруга на батареї (рис.3.21).

Рисунок 3.21 – Напруга на батареї та піні VCC

Після цього виводиться рівень заряду батареї у відсотках та рівень фонового сигналу (рис.3.22).

Рисунок 3.22 – Відсоток заряду батареї та фоновий сигнал

Завершує послідовність виведення інформації відображення меж діапазонів розпізнавання для кожного номіналу монет (рис.3.23).

Рисунок 3.23 – Межі діапазонів розпізнавання

Після того як виникає потреба вийняти кошти з монетниці, необхідно скинути кількість грошей, зафіксовану в системі. Для цього потрібно перезавантажити монетницю, затиснути кнопку «Калібрування» та утримувати її протягом 10 секунд. Після цього на дисплеї з’являється повідомлення «Пам’ять очищена», а сума обнуляється (рис.3.24–3.25).

Рисунок 3.24 – Повідомлення “Пам'ять очищена”

Рисунок 3.25 – Головний екран

Після перевірки працездатності багатофункціональна автономна монетниця готова до подальшого використання та зберігання коштів.

Висновки до розділу

У третій частині було розроблено програмне забезпечення, яке забезпечує роботу багатофункціональної автономної монетниці та реалізовує всі необхідні

функції, зокрема, збереження даних в енергонезалежній пам’яті, що дає змогу відновлювати роботу після вимкнення живлення.

Також реалізовано можливість налаштовувати пристрій та калібрувати монетоприймач та купюроприймач. Це дозволяє користувачам адаптувати пристрій до різних валют та забезпечувати точність прийому коштів.

В результаті проведеної перевірки працездатності багатофункціональної автономної монетниці з LCD-дисплеєм на базі мікроконтролера ATmega328, встановлено, що вона працює належним чином і виконує поставлені на неї завдання.

Розроблена багатофункціональна автономна монетниця визначає номінали монет та банкнот, сумує їх кількість, відображає на LCD-дисплей дані про назву валюти, суму накопичених коштів, заряд акумулятора, а також зберігає вищевказану інформацію в енергонезалежну пам’ять для подальшого відновлення роботи після режиму сну.

Розроблена багатофункціональна автономна монетниця також дозволяє користувачеві отримувати статистику по кількості монет та банкнот в монетниці для кожного номіналу.

ВИСНОВКИ

У процесі виконання бакалаврської роботи було розроблено багатофункціональну автономну монетницю з LCD-дисплеєм на базі мікроконтролера ATmega328, яка забезпечує інфомування користувача про кількість коштів в скарничці. Багатофункціональна монетниця забезпечує зчитування даних з датчиків, відображає розпізнаний номінал монети або банкноти на LCD-дисплеї та зберігає ці дані в енергонезалежну пам'ять.

Розроблена багатофункціональна монетниця може бути використана користувачем для накопичення коштів та відслідковування прогресу накопичення. Монетниця об’єднала в собі основні функції лічильника монет та банкнот.

Необхідно відмітити, що вартість розробленої багатофункціональної монетниці становить 1372,50 гривень, що є в 11 разів дешевше ніж один найдешевший лічильник банкнот.

В результаті проведеної перевірки працездатності багатофункціональної автономної монетниці з LCD-дисплеєм на базі мікроконтролера ATmega328, встановлено, що вона працює належним чином і виконує поставлені на неї завдання.

ПЕРЕЛІК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Банківське обладнання. *ТЕХНОЛОДЖИК*. URL: https://technologic.ua/uk/cat/bankovskoye-oborudovaniye/?srsltid=AfmBOoob3P6HnulBI5fbYKAqk94PFrTgm91165tYEg0qvs7vnFwep7gj (дата звернення: 10.04.2025).

2. Лічильник монет Native C8 . *SystemGroup | Автоматизація бізнесу*. URL: https://systemgroup.com.ua/uk/equipment/bankivske-obladnannya/rahuvalnyky-banknot-ta-monet/lichylnyk-monet-native-c8 (дата звернення: 12.05.2025)

3. Машинка для рахунку монет Cassida C550. *CASSIDA UKRAINE – інтернет-магазин*. URL: https://cassida.com.ua/ua/schetchik-monet-cassida-c550/ (дата звернення: 12.04.2025).

4. The Pelican 300 Series of Coin Counters. *CTcoin.com*. URL: https://www.ctcoin.com/products/table-top-coin-counters/pelican/ (дата звернення: 12.04.2025).

5. Лічильник купюр PRO 15. *Super Money Counter*. URL: https://super-money-counters.com.ua/ua/pro-15/ (дата звернення: 12.04.2025).

6. Лічильник банкнот BCASH 9500T. *ТокШоп*. URL: https://www.tokshop.net/shop/1896/desc/bcash-9500t?srsltid=AfmBOoqsUbZk2GnYIR1o0b3LemSyTaM4E61edH5BUtgbarb_A9S1i2D_ (дата звернення: 12.04.2025).

7. Професійна машинка для рахування грошей Cassida Xpecto. *CASSIDA UKRAINE*. URL: https://cassida.com.ua/ua/schetchik-banknot-cassida-xpecto/ (дата звернення: 12.04.2025).

8. Про монети. *Національний банк України*. URL: https://bank.gov.ua/ua/uah/obig-coin (дата звернення: 01.05.2025).

9. Про банкноти. *Національний банк України*. URL: https://bank.gov.ua/ua/uah/obig-banknote (дата звернення: 01.05.2025).

10. Плата Arduino Uno R3 Atmega328 з кабелем. *BeeGreen*. URL: https://arduino.ua/prod2177-arduino-nano-v3-0-avr-atmega328-p-20au (дата звернення: 01.05.2025).

11. Arduino Nano V3 ATmega328P-AU нерозпаяна. *Arduino*. URL: https://arduino.ua/prod2177-arduino-nano-v3-0-avr-atmega328-p-20au (дата звернення: 01.05.2025).

12. Плата розробки ESP32-WROOM-32UE WIFI Bluetooth 2в1 Type C. *BeeGreen*. URL: https://beegreen.com.ua/plata-razrabotki-esp32-wroom-32ue-wifi-bluetooth-2v1-type-c-20249 (дата звернення: 01.05.2025).

13. Дисплей 1602 5V зелений екран з інтерфейсною шиною IIC I2C. *BeeGreen*. URL: https://beegreen.com.ua/displej-1602-5v-zelenij-ekran-z-interfejsnoyu-shinoyu-iic-i2c-17648 (дата звернення: 01.05.2025).

14. Дисплей OLED модуль SSD1306 I2C 0.96 128x64 синій. *BeeGreen*. URL: https://beegreen.com.ua/displej-1602-5v-zelenij-ekran-z-interfejsnoyu-shinoyu-iic-i2c-17648 (дата звернення: 01.05.2025).

15. Дисплей SPI TFT ILI9341 240*320 2.8. *BeeGreen*. URL: https://beegreen.com.ua/displej-spi-tft-ili9341-240-320-28-14616 (дата звернення: 01.05.2025).

16. Інфрачервоний світлодіод LED F3 3mm 940nm. *BeeGreen*. URL: https://beegreen.com.ua/%D1%96nfrachervonij-sv%D1%96tlod%D1%96od-led-f3-3mm-940nm-15333 (дата звернення: 01.05.2025).

17. Фотодіод інфрачервоний приймач світлодіодний датчик F5 940nm. *BeeGreen*. URL: https://beegreen.com.ua/led-svitlodiod-f5-940nm-infrachervonij-infrared-11885 (дата звернення: 01.05.2025).

18. Мікродвигун з редуктором 6V 30 RPM. *Arduino.ua*. URL: https://arduino.ua/prod3382-mikromotor-s-redyktorom-6v-30-rpm (дата звернення: 01.05.2025).

19. Транзистор SUD50N024 (50N024) оригінал, DPAK (22V, 80A ,6mR). *RadioPulse*. URL: https://radiopulse.com.ua/tranzistor-sud50n024-d-pak-22v-80a-6mr?srsltid=AfmBOoqqpV6yk-

o5yGZQvBp3vabwxyV0jTl9knLWULGbcGeDXYDFKldH (дата звернення: 01.05.2025).

20. Датчик кольору GY-31 TCS230 TCS3200. *МікроАмпер*. URL: https://uamper.com/index.php?route=product/product&path=159&product_id=5560&gad_source=1&gad_campaignid=20839106384&gbraid=0AAAADk5dvo50bogviI3Wz7rkT-totPbZ&gclid=Cj0KCQjw0qTCBhCmARIsAAj8C4aG8VbukE6KOwOU6-de_vQPH393rxvptx-9RTC0KvZaLM4zGhZP5hsaAjpQEALw_wcB (дата звернення: 01.05.2025).

21. RGB датчик кольору на ISL29125 від SparkFun. *Arduino.ua*. URL: https://arduino.ua/prod1807-rgb-datchik-cveta-na-isl29125-ot-sparkfun?srsltid=AfmBOooLg26sQ78Df33Scv-5xNKs-mMni5lXQSVlWre0ZK5OfuZRU0tB (дата звернення: 01.05.2025).

22. Датчик розпізнавання кольору TCS34725 підтримка RGB I2C STM32. *BeeGreen*. URL: https://beegreen.com.ua/datchik-rozpiznavannya-koloru-tcs34725-pidtrimka-rgb-i2c-stm32-17860 (дата звернення: 01.05.2025).

23. Модуль ДБЖ 5 V/1 A на літієвій батареї 18650. *Перша гуртівня ЕЛЕКТРИКИ*. URL: https://1gurtivnya-elektryky.prom.ua/ua/p2231045162-modul-ibp-litievoj.html (дата звернення: 01.05.2025).

24. Мікровимикач з малим ходом V-156-1C25. *BeeGreen*. URL: https://beegreen.com.ua/mikrovimikach-z-malim-hodom-v-156-1c25-18940 (дата звернення: 01.05.2025).

25. Тактова кнопка без фіксації з розміром 12*12*4.3мм SMD 4Pin. *BeeGreen*. URL: https://beegreen.com.ua/taktovaja-knopka-bez-fiksacii-12-12-43mm-smd-4p-12542 (дата звернення: 01.05.2025).

26. Getting Started with Arduino IDE 2. *Arduino Docs*. URL: https://docs.arduino.cc/software/ide-v2/tutorials/getting-started-ide-v2/ (дата звернення: 05.05.2025).

27. Currency-counting machine. *Wikipedia*. URL: https://en.wikipedia.org/wiki/Currency-counting_machine (дата звернення: 17.05.2025).

28. MoneyBox_counter: розпізнавач монет із лічильником суми та іншою статистикою. *GitHub*.
URL: https://github.com/AlexGyver/MoneyBox_counter (дата звернення: 14.05.2025).

29. Банкноти і монети. *Національний банк України*. URL: https://bank.gov.ua/ua/uah (дата звернення: 13.05.2025).

30. Guide for TCS230/TCS3200 Color Sensor with Arduino. *Random Nerd Tutorials*. URL: https://randomnerdtutorials.com/arduino-color-sensor-tcs230-tcs3200/ (дата звернення: 21.05.2025).

31. Coin Acceptors Are Higher-Tech Than You Think. *Hackaday*. URL: https://hackaday.com/2022/04/10/coin-acceptors-are-higher-tech-than-you-think/.
(дата звернення: 14.05.2025).

32. Photodiode: Working and how to use in circuits. *Gadgetrinix.* URL: https://www.gadgetronicx.com/working-photodiode-applications/ (дата звернення: 25.10.2024).

33. Weldin N. R., Margolis M., Jepson B. Arduino Cookbook: Recipes to Begin, Expand, and Enhance Your Projects. O'Reilly Media, Incorporated, 2020.

34. Hughes J. M. Arduino: a Technical Reference: A Handbook for Technicians, Engineers, and Makers. O'Reilly Media, Incorporated, 2016.

35. I²C. *Wikipedia.* URL: https://en.wikipedia.org/wiki/I%C2%B2C (дата звернення: 02.06.2025).

36. Gawande G. Arduino Assisted Vending Machine with RFID Technology. *Academia.edu - Find Research Papers, Topics, Researchers*. URL: https://www.academia.edu/114418724/Arduino_Assisted_Vending_Machine_with_RFID_Technology (дата звернення: 14.06.2025).

37. Про затвердження Правил визначення платіжних ознак та обміну банкнот, розмінних та обігових монет національної валюти України :

Постанова Нац. банку України від 03.12.2018 № 134 : станом на 2 лип. 2024 р. URL: https://zakon.rada.gov.ua/laws/show/v0134500-18#Text (дата звернення: 14.06.2025).

38. Automatic paper vending and direction control / A. V. Chavan та ін. PCE Electronics Journal. 2018. URL: https://www.pce.ac.in/wp-content/uploads/2019/05/ETRX-student-journal-2018-19.pdf (дата звернення: 01.06.2025).

39. Microcontroller Based Coin Counter with Segregator and Packing System. *ResearchGate*.
URL: https://www.researchgate.net/publication/377847357_Microcontroller_Based_Coin_Counter_with_Segregator_and_Packing_System (дата звернення: 03.06.2025).

40. Currency Recognition Using Image Processing. *Saiwa | Privacy Preserving AI and ML Services*. URL: https://saiwa.ai/blog/currency-recognition-using-image-processing/ (дата звернення: 06.06.2025).

Додатки

Програмне забезпечення для багатофункціональної автономної монетниці з LCD-дисплеєм на базі мікроконтролера ATmega328

```
//-------НАЛАШТУВАННЯ---------
#define coin_amount 6       // число номіналів монет, які потрібно розпізнати
#define banknote_amount 7     // число номіналів банкнот, які потрібно розпізнати
#define coin_amount_calibration 5   //   число монет одного номіналу для калібрування
float coin_value[coin_amount] = {1.0, 10.0, 0.50, 5.0, 2.0, 1.0 };  // вартість, номінал монет у порядку зменшення діаметра
float banknote_value[banknote_amount] = {10.0, 20.0, 50.0, 100.0, 200.0, 500.0, 1000.0};  // номінали банкнот
String currency = "UAH"; // валюта UAH - Українська гривня
int displ_coin_value_time = 2000; // час відображення на дисплеї номіналу монети, що пролетіла (мілісекунди)
int stb_time = 30000;     // час бездіяльності (мілісекунди), через який система піде в сон
char* print_coin_value[coin_amount] = {"RU", "RU", "RU", "RU", "RU", "RU"}; // для виводу номіналу монет кирилецею
char* print_banknote_value[coin_amount] = {"RU", "RU", "RU", "RU", "RU", "RU"}; // для виводу номіналу банкнот кирилецею
char * stime; // char * текстовий масив для зберігання часу від початку запуску програми у форматі  (h, m, s)

//-------НАЛАШТУВАННЯ ДЛЯ ВОЛЬТМЕТРА--------
// r3 і r4 резистори дільника напруги
const float r3 = 215.0;  // 220 Ом,
const float r4 =  2050000.0;  // 2.0~8.0 МОм
float VCC = 0.0; // напруга живлення мікроконтролера Vcc (безпосередньо на піні Vcc)
float battery_voltage = 0.0; // напруга на батареї
```

```
    float max_battery_voltage = 4.20; // виставляємо максимальне значення напруги на батареї при повній зарядці (4,10-4,25 в)
    float min_battery_voltage = 2.60; // задаємо мінімальну допустиму робочу напругу на батареї (2,60 в)
    float battery_charge; // залишок заряду батареї у відсотках (0~100%)
    const float vcc_const = 1.097; //     1.0 -- 1.2 константа(const) калібрування напруги

    //-----ОГОЛОШЕНННЯ ЗМІННИХ ТА ЇХНІЙ ТИП ДАНИХ
    int coin_signal; // змінна - максимальне значення сигналу монети, що пролетіла, використовується для всіх номіналів, не зберігаємо
    int sens_signal, last_sens_signal; // значення рівня сигналу монети від фотодатчика в динаміці
    int coin_signal_min[coin_amount];  // тут зберігається нижнє значення межі діапазону сигналу для кожного розміру (номіналу) монет
    int coin_signal_max[coin_amount];  // тут зберігається верхнє значення межі діапазону сигналу для кожного розміру (номіналу) монет
    int  coin_quantity[coin_amount];    // зберігаємо кількість монет кожного номіналу
    int banknote_quantity[banknote_amount];  // кількість банкнот кожного номіналу
    int banknote_signal_R[banknote_amount];  // тут зберігається значення частоти червоного кольору для кожної банкноти
    int banknote_signal_G[banknote_amount];  // тут зберігається значення частоти зеленого кольору для кожної банкноти
    int banknote_signal_B[banknote_amount];  // тут зберігається значення частоти синього кольору для кожної банкноти
    int  empty_signal_B = 200; // зберігаємо фонове значення частоти синього кольору купюроприймача
```

```
byte empty_signal;   // зберігаємо фонове значення сигналу детектора монет (пустий_сигнал)
byte N = 10;   // створюємо змінну «N», порядковий номер номіналу монети або банкноти. При N = 10 - купюра не визначена
unsigned long standby_timer; // таймер очікування (перехід в сон)
unsigned long reset_timer; // таймер очищення пам'яті, він так само використовується для примусового виходу з калібрування
float summ_money = 0;          // сума монет і банкнот у скарбничці

// Опис функцій
void good_night();// Функція сну
float readVcc();//  Функція читання внутрішньої опорної напруги
void wake_up(); // Функція - обробник переривання

// Зберігання частоти зчитування з Out датчика кольору. Створюємо змінну ... frequency для кожного кольору
int redFrequency = 0; // значення частоти для червоного кольору
int greenFrequency = 0;// значення частоти для зеленого кольору
int blueFrequency = 0;// значення частоти для синього кольору

// =====Створюємо структуру для зберігання трьох частот за кожен колір
struct MyStruct {
  int redFrequency; // значення частоти для червоного кольору
  int greenFrequency;  //  значення частоти для зеленого кольору
  int blueFrequency; // значення частоти для синього кольору
};
// ============
```

```
    boolean sleep_flag = true;   // прапорець sleep_flag = true (прапор сну)
    boolean coin_flag = false;//  Прапорець для фіксації прольоту монети

    //------ПІДКЛЮЧЕННЯ БІБЛІОТЕК---------
    #include "LowPower.h" // бібліотека для керування режимами енергозбереження
    #include "EEPROMex.h" // бібліотека для роботи з внутрішньою пам'яттю
    #include "LCD_1602_RUS.h"// бібліотека для LCD дисплея
    #include <Wire.h>      // бібліотека для роботи з I2C інтерфейсом
    LCD_1602_RUS lcd(0x27, 16, 2); // створюємо дисплей
    // якщо дисплей не працює, замінити 0x3f на 0x27

    //-------ПІНИ КНОПОК І ПІНИ ЖИВЛЕННЯ--------

    #define button 2                // кнопка «прокинутися для монетоприйомника»
    #define banknote_button 4 // кнопка «прокинутися для купюроприймача»
    #define calibr_button 3   // прихована кнопка калібрування скидання
    #define disp_power 12    // живлення дисплея
    #define LEDpin 11        // живлення ІЧ діода

    #define IRpin A3        // живлення фотодіода
    #define IRsens A0        // сигнал фотодіода
    #define VoltmetrBatt A1  // вольтметр батареї

    // Контакти підключення датчика кольору TCS3200 до Arduino та призначення інших пінів детектора банкнот
```

```
    #define S2 6          // S2 і S3 вибір фільтра (червоний, зелений, синій)
    #define S3 7          // S2 і S3 вибір фільтра (червоний, зелений, синій)
    #define sensorOut 8        // вказуємо пін входу частоти від датчика кольору
    #define sensor_color_power 9    // живлення датчика кольору
    #define motor_power 5         // управління живленням мотора, (високе на Mosfet)
    //-------ПІНИ---------

    void setup() {
      Serial.begin(38400);  // відкриваємо порт для зв'язку з ПК для налагодження і задаємо швидкість роботи монітора порту
      delay(500); // зупиняємо виконання програми на пів секунди
      void(* resetFunc) (void) = 0;//оголошуємо функцію reset з адресою 0 (для програмного ресету)
      analogReference(DEFAULT);  // опорною напругою вважати напругу живлення мікроконтролера Vcc=5 вольт, (режим за замовчуванням, можна не задавати, на піні AREF=Vcc=5 в)


      pinMode(banknote_button, INPUT_PULLUP);// кнопка в купюроприймачі
      pinMode(button, INPUT_PULLUP);// кнопка монетоприймача
      pinMode(calibr_button, INPUT_PULLUP);// кнопка калібрування
      pinMode(disp_power, OUTPUT); // живлення дисплея
      pinMode(LEDpin, OUTPUT); // живлення ІЧ діода
      pinMode(IRpin, OUTPUT); // живлення фотодіода
      pinMode(S2, OUTPUT); // комбінація S2 і S3 активує червоні, зелені або сині фотодіоди датчика кольору
      pinMode(S3, OUTPUT);
```

```
    pinMode(sensor_color_power, OUTPUT); // живлення датчика кольору
      pinMode(motor_power, OUTPUT); // управління живленням мотора, (високе на Mosfet)
      // Встановлення sensorOut як входу
      pinMode(sensorOut, INPUT); // вхід для вимірювання частоти з датчика кольору

      // подати живлення на дисплей і діоди датчика монетоприймача
      digitalWrite(disp_power, HIGH); // живлення дисплея
      digitalWrite(LEDpin, HIGH);  //  живлення ІЧ діода
      digitalWrite(IRpin, HIGH);  //  живлення фотодіода


      // підключити переривання
      //апаратне переривання №0, використовується цифровий PIN D2, куди під'єднана кнопка (або контакти) прокинутися монетоприймача
      attachInterrupt(0, wake_up, CHANGE);
      //апаратне переривання №1, використовується цифровий PIN D3, куди під'єднана кнопка прокинутися купюроприймача
      attachInterrupt(1, wake_up, CHANGE);

      empty_signal = analogRead(IRsens);  // зчитати порожній (опорний/фоновий) сигнал empty_signal

      // ІНІЦІАЛІЗАЦІЯ ДИСПЛЕЮ
      lcd.init(); // ініціалізація дисплею
      lcd.backlight(); //  Вмикаємо підсвітку LCD дисплея
      if (!digitalRead(calibr_button)) {  // якщо під час запуску натиснуто кнопку КАЛІБРУВАННЯ
        for (byte i = 0; i < coin_amount; i++) {
```

```
          coin_quantity[i] = EEPROM.readInt(i * 2); // зчитуємо кількість монет кожного номіналу, якщо вони були збережені раніше
        }
        // Виведення на дисплей «Сервіс» та інформації про процес калібрування
        lcd.clear();

      lcd.setCursor(5, 0);
        lcd.print(L"Сервіс");
        delay(500);
        reset_timer = millis(); //скинути таймер
        while (1) {
          if (millis() - reset_timer > 10000) {// якщо кнопка все ще утримується і минуло 10 секунд,
            // то очистити кількість монет у пам'яті
            for (byte i = 0; i < coin_amount; i++) {
              coin_quantity[i] = 0; // Пам'ять ОБНУЛЕНА! Кількість монет у скарбничці всіх номіналів - нуль
              EEPROM.writeInt(i * 2, 0); // заповнюємо всі комірки пам'яті з адреси 0 по .... нулями
            }
            // а також очистити кількість банкнот у пам'яті
            for (byte i = 0; i < banknote_amount; i++) {
              banknote_quantity[i] = 0; // Пам'ять ОБНУЛЕНА! Кількість банкнот у скарбничці всіх номіналів - нуль
              EEPROM.writeInt(60 + i * 2, 0); // заповнюємо всі комірки пам'яті з адреси 60 по .... нулями
            }
            // Кількість комірок пам'яті ATmega328 (Arduino UNO, Nano, Pro Mini): 1 кБ(1024 Б), адреси з 0 по 1024
            lcd.clear();
            lcd.setCursor(1, 0);
            lcd.print(L"Память очищена");
            delay(200);
```

```
          }
          if (digitalRead(calibr_button)) {    // якщо відпустили кнопку, перейти до калібрування
            lcd.clear();
            lcd.setCursor(0, 0);
            lcd.print(L"Калібровка");
            delay(500);
            reset_timer = millis(); // скинути таймер

        break;
          }
         }
         /**/
         // ==========   ВИМІРЮВАННЯ ТА ВИВЕДЕННЯ МЕЖ ФОНУ НА ДИСПЛЕЙ ========================================
         empty_signal = analogRead(IRsens);   // зчитати порожній (опорний/фоновий) сигнал empty_signal
         last_sens_signal   =   empty_signal;      //   значення last_sens_signal   =   порожньому   (опорному/фоновому)   сигналу empty_signal
         while (1) {// цикл на 3 секунди для заміру і виведення меж фону на дисплей
          lcd.setCursor(0, 0); lcd.print(L"Замір Фону");
          sens_signal = analogRead(IRsens); // зчитати фотодатчик

          if (sens_signal < empty_signal) {
            empty_signal = sens_signal;   // фіксуємо мінімальне значення фону
          }
          if (sens_signal > last_sens_signal) {
            last_sens_signal   =   sens_signal;   //   фіксуємо максимальне значення фону
          }
```

```
          if (millis() - reset_timer > 3000) break; // виходимо з режиму виміру фону після закінчення часу 3 сек
        } // кінець циклу заміру меж фону
        sens_signal = empty_signal  ; // виводимо межі фону
        lcd.setCursor(11, 0);
        lcd.print(sens_signal);                    lcd.print("-"); lcd.print(last_sens_signal); // виводимо на дисплей min і max значення фону
        Serial.print(empty_signal);             Serial.print("-"); Serial.println(last_sens_signal); // виводимо в порт min і max значення фону
        last_sens_signal = empty_signal; // скидаємо змінну last_sens_signal у мінімальне значення фону

      delay(3000);// час відображення фону, після чого переходимо безпосередньо до калібрування
        lcd.clear();
        lcd.setCursor(0, 0); lcd.print(L"Калібровка");
        // ====== КІНЕЦЬ БЛОКУ ВИМІРЮВАННЯ ТА ВИВЕДЕННЯ МЕЖ ФОНУ НА ДИСПЛЕЙ  ==================================

        while (1) {
          for (byte i = 0; i < coin_amount; i++) { // послідовний перебір номіналів монет, розмір, яких калібрується
            for (byte k = 0; k < coin_amount_calibration; k++) { // повторення калібрування монети того самого номіналу (k разів)
              // виведення на дисплей
              lcd.setCursor(0, 1); lcd.print(coin_value[i]);   // відобразити ціну монети, розмір якої калібрується
              lcd.setCursor(6, 1); lcd.print(currency);        // відобразити валюту
              lcd.setCursor(14,                                1); lcd.print(coin_amount_calibration - k); // відобразити кількість спроб калібрування монети того самого номіналу, що залишилася
```

```
                if (coin_amount_calibration - k == 9 ) {
                  lcd.setCursor(15, 1);
                  lcd.print(" "); // очистити сегмент 16 від нуля
(кількість спроб - двозначне число)
                }
                empty_signal = analogRead(IRsens);    // зчитати
порожній (опорний/фоновий) сигнал empty_signal
                last_sens_signal = empty_signal;    // значення
last_sens_signal = порожньому (опорному/фоновому) сигналу
empty_signal

                while (1) {    // нескінченний цикл сканування
фотодатчика й аналізу отриманого значення
                  // якщо нічого не робити 60 секунд, то система
вийде з циклу калібрування програмним перезавантаженням
                  if (millis() - reset_timer > 60000) resetFunc();
//викликаємо функцію reset
                  sens_signal = analogRead(IRsens);      // зчитати
фотодатчик

          if (sens_signal > last_sens_signal) last_sens_signal =
sens_signal;   // якщо поточне значення більше за попереднє, то
запам'ятовуємо його
                  if (sens_signal - empty_signal > 5) coin_flag =
true;   // якщо значення перевищило фонове на 5, вважати, що
пролітає монета, ставимо прапор в ІСТИНА
                  if    (coin_flag    &&    (abs(sens_signal    -
empty_signal)) < 4) { // якщо потім значення впало, повернулося
майже до фонового, то монета точно пролетіла
                    coin_signal = last_sens_signal; // зафіксували
максимальне значення сигналу монети, що пролетіла
                    if (k < 1) {
                      coin_signal_max[i] = coin_signal;
                      coin_signal_min[i] = coin_signal;
```

```
                    }
                    if     (coin_signal     >     coin_signal_max[i])
coin_signal_max[i] = coin_signal; // фіксуємо максимальне значення
                    if     (coin_signal     <     coin_signal_min[i])
coin_signal_min[i] = coin_signal; // фіксуємо мінімальне значення
                    coin_quantity[i]++;    // додаємо до загальної
кількості монет даного номіналу ще одну
                    coin_flag  =  false;    // переводимо прапорець
фіксації прольоту монети знову в 0
                    reset_timer   =   millis();   //скинути   таймер
програмного reset для примусового виходу з калібрування
                    lcd.setCursor(11,  0);  lcd.print(coin_signal);
lcd.print("    ");// виводимо у верхній правий кут дисплея сигнал
за монету
                    break; // вихід з циклу
                  } // кінець, монета, що пролітає, відсканована,
повернення в безперервний цикл сканування фотодатчика, чекаємо на
проліт наступної монети
                } // дужка циклу while (1) { } безперервний цикл
сканування фотодатчика
              }  //  дужка  циклу  for  (byte  k  =  0;  k  <
coin_amount_calibration; k++), повторення калібрування монети того
самого номіналу (k разів)
            }// дужка циклу for (byte i = 0; i < coin_amount; i++),
перебору номіналів монет, розмір, яких калібрується
            break;// Вихід з циклу калібрування
          } // дужка while (1) { } циклу калібрування

      //---------------Виведення результатів калібрування---------
----------
          // для контролю, виводимо отримані статистичні значення
меж діапазонів розпізнавання монет
          Serial.println("Statistical     boundaries     of     coin
recognition ranges");
```

```
        for (byte i = 0; i < coin_amount; i++) {
          Serial.println("coin_signal_max~min[" + String(i) + "]: " + String(coin_signal_max[i]) + "~" + String(coin_signal_min[i]));
        }
        coin_signal_max[0] = 1023; // верхня межа монети з максимальним діаметром обмежена цифрою 1023
        coin_signal_min[coin_amount - 1] = empty_signal + 15 ; // нижня межа монети з мінімальним діаметром на 15 одиниць вища за фонове значення
        for (byte i = 0; i < coin_amount; i++) {
          if (i > 0) {
            coin_signal_max[i] = (coin_signal_min[i - 1] - 1); //Верхня межа діапазону номіналу монети
          }
          if (i < coin_amount - 1) { // поки змінна last_sens_signal ВІЛЬНА, використовуємо її для обчислень меж діапазонів!
            //-------------------------------------------------------------------------------------
            last_sens_signal = (coin_signal_min[i] - coin_signal_max[i + 1]) / 2; // величина збільшення межі діапазону, округлена до цілого значення
            coin_signal_min[i] = coin_signal_min[i] - last_sens_signal; // нижня межа діапазону номіналу монети
          }
        }

        // Записуємо як масив даних розраховані, розширені межі діапазонів розпізнавання монет у пам'ять і кількість монет
        for (byte i = 0; i < coin_amount; i++) {
```

```
          EEPROM.updateInt(i * 2, coin_quantity[i]); // оновлює байт даних у комірці за кількість монет, якщо її старий вміст відрізняється від нового

        EEPROM.writeInt(20 + i * 2, coin_signal_min[i]); // послідовно записали мінімальне значення за кожну монету в ПАМ'ЯТЬ у клітинку з 20 по ...
          EEPROM.writeInt(40 + i * 2, coin_signal_max[i]); // послідовно записали максимальне значення за кожну монету в ПАМ'ЯТЬ у комірку з 40 по ...
        }

      } // дужка кінець циклу калібрування, кнопку «Калібрування» раніше було натиснуто
      // під час старту системи (або під час запуску COM порту) зчитуємо з пам'яті кількість монет і банкнот кожного номіналу в скарбничці
      // а так само зчитуємо з пам'яті верхнє і нижнє значення діапазону сигналу для кожного номіналу монет і кольору банкнот
      for (byte i = 0; i < coin_amount; i++) {
        coin_quantity[i] = EEPROM.readInt(i * 2);// кількість монет кожного номіналу
        coin_signal_min[i] = EEPROM.readInt(20 + i * 2); // зчитує значення нижньої межі діапазону
        coin_signal_max[i] = EEPROM.readInt(40 + i * 2); // зчитує значення верхньої межі діапазону
        summ_money += coin_quantity[i] * coin_value[i]; // рахуємо загальну суму монет, як добуток номіналів монет на їхню кількість
      }
      // під час старту системи зчитати з пам'яті значення кольору банкнот для подальшої роботи, а також їхню кількість у скарбничці
      for (byte i = 0; i < banknote_amount; i++) {
```

```
        banknote_quantity[i] = EEPROM.readInt(60 + i * 2);//
кількість банкнот кожного номіналу
        //banknote_signal_R[i] = EEPROM.readInt(80+i * 2); //
частота інтенсивності червоного кольору банкноти
        //banknote_signal_G[i] = EEPROM.readInt(100+i * 2); //
частота інтенсивності зеленого кольору банкноти
        //banknote_signal_B[i] = EEPROM.readInt(120+i * 2); //
частота інтенсивності синього кольору банкноти
        summ_money += banknote_quantity[i] * banknote_value[i];
// до суми вже підрахованих раніше монет,
        // додаємо загальну суму банкнот, як добуток номіналів
банкнот на їхню кількість
      }

      // ЕТАЛОННІ ЗНАЧЕННЯ ЧАСТОТ ДЛЯ КОЖНОГО НОМІНАЛУ БАНКНОТИ
      banknote_signal_R[0] = 38; banknote_signal_G[0] = 54;
banknote_signal_B[0] = 43; // значення частот за кожен колір R,G,B
для 10 гривневої банкноты
      banknote_signal_R[1] = 44; banknote_signal_G[1] = 45;
banknote_signal_B[1] = 40; // значення частот за кожен колір R,G,B
для 20 гривневої банкноты
      banknote_signal_R[2] = 42; banknote_signal_G[2] = 46;
banknote_signal_B[2] = 34; // значення частот за кожен колір R,G,B
для 50 гривневої банкноты
      banknote_signal_R[3] = 37; banknote_signal_G[3] = 42;
banknote_signal_B[3] = 35; // значення частот за кожен колір R,G,B
для 100 гривневої банкноты
      banknote_signal_R[4] = 39; banknote_signal_G[4] = 47;
banknote_signal_B[4] = 33; // значення частот за кожен колір R,G,B
для 200 гривневої банкноты
      banknote_signal_R[5] = 39; banknote_signal_G[5] = 48;
banknote_signal_B[5] = 39; // значення частот за кожен колір R,G,B
для 500 гривневої банкноты
```

```
      banknote_signal_R[6]  =  50;  banknote_signal_G[6]  =  53;
banknote_signal_B[5] = 39; // значення частот за кожен колір R,G,B
для 1000 гривневої банкноты
      // Кінець ручного завдання розпізнавання банкнот

      standby_timer  =  millis()  -  (displ_coin_value_time  +  4)  ;
// не обнуляємо таймер, а скидаємо таймер, до значення
      // «таймера тривалості часу виведення номіналу монет», щоб
не було потім повторного виведення інформації на дисплей

    } // Кінець виконання функції void setup

    //==========          Основний          блок          програми
==================================================

    void loop() {
      // якщо if sleep_flag = true, то виконуємо те, що в дужках
{...} нижче
      if  (sleep_flag)  {  //  прапор  сну,  все,  що  в  дужках,
виконується ТІЛЬКИ після прокидання або старту (RESET)
        delay(500);// пауза пів секунди

       // Звернення до функції TimeToString часу активної роботи
МК і до функції readVcc() читання внутрішньої опорної напруги.
        stime  =  TimeToString(millis()  /  1000);  //звернення  до
функції TimeToString з параметрами millis()/1000, сумарний час з
моменту запуску програми
        VCC = readVcc(); // звернення до функції readVcc()читання
внутрішньої  опорної  напруги.  Змінна  VCC  =  значенню  напруги
живлення Vcc
        // Звернення до функції BattCharg() вимірювання напруги
на  акумуляторній  батареї  та  розрахунку  залишку  заряду
акумуляторної батареї
```

```
        battery_charge = BattCharg(); //звернення до функції BattCharg()
        // Звертаємося до функції виведення на дисплей інформації: назви скарбнички, накопиченої суми грошей, символ валюти, залишок заряду батареї
        lcd.init();//ініцилізація дисплея тільки після сну і старту, займає час, тому в середині самої функції виведення DisplayOutput() не застосовуємо.
        DisplayOutput(); // виконується функція виведення на дисплей стандартної інформації
        // ------------------------------------------------------------------
        empty_signal = analogRead(IRsens); // зчитати датчик (зчитати порожній (опорний/фоновий) сигнал empty_signal)
        sleep_flag = false; // прапорець сну
      }
      last_sens_signal = empty_signal;  // максимальне значення сигналу останньої монети, що пролетіла, скидаємо до рівня фону empty_signal
      N = 10;  // змінна «N» у вихідне значення 10, неіснуючого порядкового номера номіналу монети або банкноти, (невизначений номінал)
      // «N» стає дорівнює 10 щоразу після: запуску, reset, прокидання і виходу break з переходом до початку void loop()

      // Далі працюємо в нескінченному циклі безперервного циклічного сканування датчиків монет і купюр =========
      // While працюватиме в циклі безперервного сканування доти, доки час таймера відходу в сон не вийшов і потім «break»

      //==ГОЛОВНИЙ ЦИКЛ while (true) БЕЗПЕРЕРЕРЕРВНОГО ОПИТУВАННЯ КУПЮРО/МОНЕТО-ПРИЙОМНИКІВ, РОЗПІЗНАННЯ БАНКНОТ ТА МОНЕТ, ПІДРАХУНКУ СУМИ====
```

```
        while (true) {

          //===========БЛОК        СКАНУВАННЯ        КУПЮРОПРИЙМАЧА,
РОЗПІЗНАВАННЯ І ПІДРАХУНКУ БАНКНОТ===========
          if  (!digitalRead(banknote_button))   {  // якщо вставлена
купюра (замкнута кнопка купюроприймача)
            //Serial.println(N);// для контролю значення «N»

            // ========== БЛОК ВИЗНАЧЕННЯ КОЛЬОРУ КУПЮРИ, У ВИГЛЯДІ
ЧАСТОТИ ЗА КОЖЕН КОЛІР R,G,B ================

            //Звертаємося   до   функції   «Banknote_RGB_Frequency»
вимірювання інтенсивності частот за кожен колір R,G,B
            MyStruct F_RGB; // оголошуємо структуру
            F_RGB = Banknote_RGB_Frequency();
            //  =====  КІНЕЦЬ  БЛОКУ  ВИЗНАЧЕННЯ  КОЛЬОРУ  КУПЮРИ  У
ВИГЛЯДІ ЧАСТОТИ ЗА КОЖЕН КОЛІР R,G,B =====

            //  зняти  живлення  з  датчика  кольору,  якщо  забрали
купюру
            if  (digitalRead(banknote_button))    {  //  якщо  була
вийнята банкнота (розімкнулася кнопка)
              digitalWrite(sensor_color_power,  0);  //  LOW,  зняти
живлення з датчика кольору
              standby_timer = millis();  // скинути таймер
              break;//  0-й  Break  примусового  виходу  з  головного
циклу циклу while (true) і переходу на void loop()
            }


            //  ==============  БЛОК  ВИЗНАЧЕННЯ  НОМІНАЛУ  КУПЮРИ
===============
```

```
          while (!digitalRead(banknote_button)) {// якщо, як і раніше, вставлена купюра (замкнута кнопка купюроприймача)
            // Виходячи з отриманих раніше частот за кожен колір R,G,B, визначаємо колір банкноти та її номінал
            for (byte i = 0; i < banknote_amount; i++) {
              // обчислюємо абсолютне (модуль) значення різниці отриманих частот із нашими значеннями з пам'яті
              if (abs(redFrequency - banknote_signal_R[i]) < 7
                  && abs(greenFrequency - banknote_signal_G[i]) < 7
                  && abs(blueFrequency - banknote_signal_B[i]) < 7)
              { // якщо виконалися всі три умови за трьома частотами, то запам'ятовуємо номінал купюри N = i, точніше номер купюри
                N = i;
                // вставлена банкнота розпізнана, але поки що не протягнута і не порахована
                //    ДАЛІ,    при    зверненні    до    функції DisplayOutput(), виводимо на дисплей номінал розпізнаної банкноти.
                break;//Break використовується для примусового виходу з циклу for, щоб не перебирати інші комбінації
              }
            }
            break;//Break використовується для примусового виходу з циклу while(!digitalRead(banknote_button) (Робимо його ОДНОРАЗОВИМ!)
          } // дужка while (!digitalRead(banknote_button)). Виконається цикл тільки один раз, оскільки в кінці стоїть «break

          // якщо купюра не розпізнана, то N == 10, виводимо «ПОМИЛКА»="ERROR»
          // протягом часу stb_time буде прийнято кілька спроб визначити номінал купюри (при 15 сек це 3 рази),
```

```
          // і якщо банкнота, залишаючись у купюроприймачі, не буде впізнана, то система піде в сон. Купюра протягнута не буде.
          if (N == 10) {
            lcd.clear();

     lcd.setCursor(5,  0);  lcd.print("ERROR");//  виведення  на дисплей
            // якщо час таймера вийшов, спимо (stb_time час бездіяльності, через який система піде в сон )
            if (millis() - standby_timer > stb_time) {
              good_night(); // звернення до функції сну
            }

          }
          // якщо купюра не впізнана (N == 10), то пропускаємо else {...}, потрапляємо на 3-й Break
          // для примусового виходу з ГОЛОВНОГО циклу while (true) і переходу на void loop()
          // в іншому випадку, якщо «N» стала відмінна від 10, то купюра була впізнана! Виконуємо те, що в else {...}
          else  { // якщо N стала відмінна від 10, значить, купюра, була розпізнана!
            DisplayOutput();// звертаємося до функції виведення на дисплей інформація.(умови «N»!= 10(не дорівнює) і banknote_flag = false)
            // виводиться номінал розпізнаної банкноти (але поки вона не простягнута і не підрахована) та інша інформація

            // ================ КІНЕЦЬ БЛОКУ ВИЗНАЧЕННЯ НОМІНАЛУ КУПЮРИ ================

            // ========= БЛОК КОНТРОЛЮ ПОТРАПЛЯННЯ/НЕ ПОТРАПЛЯННЯ В СКАРБНИЧКУ ТА ОБЛІКУ/НЕ ОБЛІКУ КУПЮРИ ==========
```

```
            delay(3000);// робимо паузу перед увімкненням мотора, даємо час забрати банкноту, якщо не хочемо відправити її в скарбничку
            // зняти живлення з датчика кольору, якщо протягом цих 3 секунд забрали купюру
            if (digitalRead(banknote_button))   { // якщо була вийнята банкнота (розімкнулася кнопка)
              digitalWrite(sensor_color_power, 0); // LOW, зняти живлення з датчика кольору
              standby_timer = millis();  // скинути таймер
              break;// 1-й Break примусового виходу з головного циклу while (true) і переходу на void loop()

      // все, що нижче, виконуватися вже не буде, оскільки забрали купюру
            }
            // якщо не забрали банкноту, виконуємо все, що нижче
            digitalWrite(sensor_color_power,  HIGH);  //  подати живлення на датчик кольору
            digitalWrite(motor_power, HIGH); // увімкнути мотор для протягання купюри в скарбничку

            //Якщо вставлена банкнота розпізнана і її не забрали з купюроприймача, потрапляємо в цикл while (1)
            while (1) {//Працюємо безперервно в циклі контролю стану кнопки і потім читання частоти синього кольору, вийдемо з циклу і вимкнемо мотор
              if (digitalRead(banknote_button)) { //Якщо кнопка розімкнулася, перевіряємо, куди пройшла купюра. Всередину скарбнички чи ні, її вилучили.
                if (pulseIn(sensorOut, LOW) < empty_signal_B - 30) {//якщо кнопка розімкнулася і частота з датчика кольору менша за фонове значення (empty_signal_B)
```

```
                    //синьої складової кольору чорної підкладки купюроприймача, значить купюра захоплена, перебуває під датчиком і майже простягнута в скарбничку
                    // Тепер купюра точно в скарбничці, рахуємо її і виводимо суму в порт і на дисплей
                    summ_money += banknote_value[N];   // до суми грошей у скарбничці додаємо ціну банкноти ([N]:0 - це 10 гривень,....3 - це 50 гривень...)
                    banknote_quantity[N]++;    // для розпізнаного номера купюри додаємо кількість
                    DisplayOutput();//  звертаємося до функції виведення на дисплей інформація.(умови «N»!= 10(не дорівнює))
                    delay(1500);//  даємо мотору із запасом ще попрацювати 1,5 секунди. (Реально, після розмикання кнопки, купюра протягується 0,3 секунди)
                    digitalWrite(motor_power,  LOW);  //  вимикаємо мотор протягання купюри
                    digitalWrite(sensor_color_power, LOW); // зняти живлення з датчика кольору
                    //банкноту протягнуто і підраховано
                    //номінал розпізнаної і протягнутої банкноти відображається  на  час:12с=3[пауза]+(5,5+1,5)[час протягання]+~2с[displ_coin_value_time]
                    //якщо цей час минув, то номінал банкноти перестає відображатися і виводиться «Розумна скарбничка» та інша стандартна інформація.

     break;//виходимо примусово з циклу while (1), оскільки купюра протягнута в скарбничку і врахована в загальній сумі грошей.
                    // і блок, який відповідає за забрану назад купюру, який нижче, вже не виконується
                  } //скобка умови if (pulseIn(sensorOut, LOW) < empty_signal_B - 30)
```

```
                //===== блок програми, який визначає, що купюру в цьому циклі while (1) забрали і вона не буде врахована ============
                //мотор працює, кнопка розімкнута, під датчиком немає банкноти, чорний колір, значить, купюру забрали. Не рахуємо її.
                digitalWrite(motor_power, LOW); // вимикаємо мотор протягання купюри
                digitalWrite(sensor_color_power, LOW); // зняти живлення з датчика кольору
                delay(500);//пауза, як захист від брязкоту під час вилучення купюри, щоб не спрацювало помилково if (!digitalRead(banknote_button))
                break;//виходимо примусово з циклу while (1), оскільки купюру забрали, і вона не врахована! Купюроприймач порожній.
              }

              //якщо час таймера сну + 10 сек - вийшло, спимо. (stb_time час бездіяльності)
              if (millis() - standby_timer > stb_time + 10000) { //Тут це потрібно, щоб мотор не працював вічно, якщо ролики не захопили купюру.
                good_night(); // звернення до функції сну. Після прокидання, програма продовжить працювати з цього місця.
                N = 10;
                break; // після прокидання, break використовується для примусового виходу з циклу while(1)
              }
            } //дужка циклу while (1)
            standby_timer = millis();  // скинути таймер
          } // скупка else {купюра була розпізнана, варіанти: (1) і підрахована; (2) але, її забрали назад і вона не врахована}
```

```
          //    варіант    (3)    -    купюра    не    розпізнана,
«ПОМИЛКА»=«ERROR», пропустили else { ... }, і потрапляємо на 3-й
Break

      break;// 3-й Break використовується для примусового виходу з
ГОЛОВНОГО циклу while (true) і переходу на void loop()
          }    //    дужка    якщо    банкнота    вставлена    if
(!digitalRead(banknote_button))  {
          //===   ЗАКІНЧЕННЯ   БЛОКУ   СКАНУВАННЯ   КУПЮРОПРИЙМАЧА,
РОЗПІЗНАВАННЯ І ПІДРАХУНКУ БАНКНОТ ====

          //====== БЛОК ОПИТУВАННЯ ДАТЧИКА МОНЕТ, ФІКСАЦІЇ ПРОЛЬОТУ
І ПІДРАХУНКУ МОНЕТ==========
          // алгоритм такий самий, як під час калібрування.
          sens_signal = analogRead(IRsens);  // считать датчик
          if  (sens_signal  >  last_sens_signal)  last_sens_signal  =
sens_signal;  // якщо поточне значення більше за попереднє
          if (sens_signal - empty_signal > 5) coin_flag = true;  //
якщо значення перевищило фонове на 5, вважати, що пролітає монета,
ставимо прапор в ІСТИНА
          if (coin_flag && (abs(sens_signal - empty_signal)) < 4) {
// якщо значення потім упало, повернулося майже до фонового, то
монета точно пролетіла
            // знайшли максимум для монетки, що пролетіла, записали
в last_sens_signal
            //   далі   в   програмі   він   скинеться   до   рівня
порожнього/фонового сигналу empty_signal
            // далі починаємо порівнювати сигнал із діапазонами
монет, що зберігаються в пам'яті

            for (byte i = 0; i < coin_amount; i++) {
              if   (   last_sens_signal   >=   coin_signal_min[i]   &&
last_sens_signal <= coin_signal_max[i]) {  // !!! і ось тут якщо
сигнал потрапляє в діапазон, то вважаємо монетку розпізнаною
```

```
                summ_money += coin_value[i];// до суми тупо додаємо ціну монетки (так, сума рахується двома різними способами. При старті системи сумою всіх монет, а тут додавання
                coin_quantity[i]++;    // для розпізнаного номера монетки додаємо кількість
                N = i;// змінну N використовуємо для запам'ятовування порядкового номера номіналу монети
                //номіналу розпізнаної монети, щоб потім використовувати у функції DisplayOutput() для виведення на екран номіналу монети, що пролетіла
                // вивести на дисплей інформацію: НОМІНАЛ МОНЕТИ, що пролетіла, накопичену суми грошей, символ валюти, залишок заряду батареї

      DisplayOutput();// Звертаємося до функції DisplayOutput() виведення на дисплей
                break;// Break використовується для примусового виходу з циклу, for (byte i = 0; i < coin_amount; i++). Монету впізнано, не треба перебирати інші межі.
              }    // Монета потрапила в діапазон, розпізнана, порахована і виведена на дисплей. Break виходимо з циклу for (byte i = 0; i < coin_amount; i++)
            } // кінець циклу for (byte i = 0; i < coin_amount; i++) перебору і порівняння сигналу монетою, що пролетіла, зі збереженими в пам'яті
            coin_flag = false; // переводимо прапорець фіксації прольоту монети знову в 0 для фіксації нового прольоту монети
            standby_timer = millis();  // скинути таймер, millis() кількість мілісекунд з моменту початку виконання програми
            break; // 4-й Break використовується для примусового виходу з ГОЛОВНОГО циклу while (true) і переходу на void loop()
          }
          if (millis() - standby_timer >= displ_coin_value_time && millis() - standby_timer < displ_coin_value_time + 3) {
```

```
          // Звертаємося до функції виведення на дисплей інформації: назви скарбнички, накопиченої суми грошей, символ валюти, залишок заряду батареї
            DisplayOutput();
          }

          // якщо нічого не робили, час таймера вийшов, спимо (stb_time час бездіяльності, через який система піде в сон )
          if (millis() - standby_timer > stb_time) {
            good_night();  // звернення до функції сну. Після прокидання програма продовжить працювати з цього місця
            break; // 5-й Break використовується для примусового виходу з ГОЛОВНОГО циклу while (true) і переходу на void loop()
          }

          // Виведення статистики за кількістю монет кожного номіналу і виведення службової інформації

        while (!digitalRead(button)) { // натиснута кнопка «ПРОСНУТИСЯ» (або монетка замикає контакти)
            if (millis() - standby_timer > 10000) { // якщо контакти «ПРОСНУТИСЯ» замкнуті > 10 секунд, то виведення статистики

              //=== Якщо одночасно натиснуті кнопки «ПРОСНУТИСЯ» і «КАЛІБРОВКА», то калібрується батарея ============
              if (!digitalRead(calibr_button)) {  // якщо так само натиснута кнопка «КАЛІБРОВКА», то калібруємо батарею, 100% рівень заряду
                lcd.clear();
                lcd.setCursor(0, 0); lcd.print(L"Калібрування");
                lcd.setCursor(0, 1); lcd.print(L"батареі");
                // напруга батареї під час повної зарядки мінус 1%, приймається, як верхня межа, що відповідає 100% рівню заряду
```

```
                max_battery_voltage     =     battery_voltage     -
(battery_voltage - min_battery_voltage) / 100;
                delay(3000);
                lcd.clear();
                lcd.setCursor(0, 0); lcd.print(L"Завершена");
                lcd.setCursor(0,     1);     lcd.print("MaxBattV=");
lcd.print(String(max_battery_voltage,   3)   +   "   V");//   напруга
батареї верхня межа
                standby_timer = standby_timer - stb_time; // щоб
після наступної паузи delay (5000) відразу з калібрування йшов у
сон
                delay(5000);
                break;
              }

//================================================================
================================

              lcd.clear();
              //   Виведення номіналів банкнот зверху і виведення
статистики кількості банкнот знизу!
              /**/

       // Виведення номіналів банкнот та їхньої кількості, якщо їх
більше 4-х у два етапи!
              for (byte i = 0; i < 4; i++) { // відобразити на
дисплеї статистику для перших 4-х банкнот
                lcd.setCursor(i          *          4,          0);
lcd.print(banknote_value[i], 0); // зверху номінал банкноти цілі
числаж
                lcd.setCursor(i          *          4,          1);
lcd.print(banknote_quantity[i]); // знизу їхня кількість
              }
              delay(5000);
```

```
            if (banknote_amount > 4) { // якщо число номіналів банкнот більше ніж 4, виведення другим етапом інших
              lcd.clear();
              for (byte i = 4; i < banknote_amount; i++) { // відобразити на дисплеї статистику для інших монет
                lcd.setCursor(i * 5 - 20, 0); lcd.print(banknote_value[i], 0); // зверху номінал банкноти цілі числа
                lcd.setCursor(i * 5 - 20, 1); lcd.print(banknote_quantity[i]); // знизу їхня кількість
              }
            }
            delay(5000);
            lcd.clear();
            // Виведення номіналів монет і їхньої кількості, якщо їх більше 4-х у два етапи!
            for (byte i = 0; i < 4; i++) { // відобразити на дисплеї статистику для перших 4-х монет
              lcd.setCursor(i * 4, 0); lcd.print(coin_value[i], 1); // зверху номінал монети (округлений до десятих)
              lcd.setCursor(i * 4, 1); lcd.print(coin_quantity[i]); // знизу їхня кількість
            }
            delay(5000);
            if (coin_amount > 4) { // якщо число номіналів монет більше ніж 4, виведення другим етапом решти
              lcd.clear();
              for (byte i = 4; i < coin_amount; i++) { // відобразити на дисплеї статистику для інших монет

        lcd.setCursor(i * 5 - 20, 0); lcd.print(coin_value[i], 2); // зверху номінал монети (округлений до десятих)
                lcd.setCursor(i * 5 - 20, 1); lcd.print(coin_quantity[i]); // знизу їхня кількість
```

```
            }
          }
          //  ------------------------------------------------
-----------------------------------

          delay(4000);
          standby_timer = millis() ;  //скинути таймер

          //        ВИВЕДЕННЯ        СЛУЖБОВОЇ        ІНФОРМАЦІЇ
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~
          // Якщо кнопка «прокинутися», як і раніше, натиснута
під час виведення статистики монет, то виводимо на дисплей
службову інформацію:
          //  напруг живлення мікроконтролера, напруги на
батареї та сумарного часу роботи МК тощо.
          while  (!digitalRead(button))  {  //  якщо  кнопка
«прокинутися» як і раніше натиснута
            //-----------------------------------------------
-----------------------------------------------------------
---------
            // Звернення до функції TimeToString()- сумарний
час активної роботи МК. І до функції readVcc()- читання
внутрішньої опорної напруги.
            stime = TimeToString(millis() / 1000); //звернення
до функції TimeToString з параметрами millis()/1000, сумарний час
з моменту запуску програми
            VCC  =  readVcc();  //  Звернення  до  функції
readVcc()читання внутрішньої опорної напруги. Змінна VCC =
значенню напруги живлення Vcc

            // Звернення до функції BattCharg() вимірювання
напруги на акумуляторній батареї та розрахунку залишку заряду
акумуляторної батареї
```

```
                battery_charge   =   BattCharg();   //звернення   до
функції BattCharg()

                //-----------------------------------------------
-----------------------------------------------------------------
------------

        lcd.clear();
                lcd.setCursor(0,   0);   lcd.print("Version   V3.1");
//версія прошивки
                lcd.setCursor(0,       1);       lcd.print("time=");
lcd.print(stime); // сумарний час роботи МК

                delay(5000);
                lcd.setCursor(0,        0);        lcd.print("Vcc=");
lcd.print(VCC, 3); lcd.print(" volt "); //вивід напруги живлення
Ардуіно на піні Vcc
                lcd.setCursor(0,      1);      lcd.print("Vbatt=");
lcd.print(battery_voltage, 3); lcd.print(" volt "); // виведення
напруги на батареї Vbatt
                delay(5000);
                lcd.setCursor(0,    0);    lcd.print("BatCharge=");
lcd.print(battery_charge);  lcd.print("%");  //  відсоток  рівня
заряду батареї
                lcd.setCursor(0,   1);   lcd.print("empty_signal=");
lcd.print(String(empty_signal));  lcd.print("   ");  //  виведення
рівня фонового сигналу
                delay(5000);
                //  виведення  результату  калібрування  монет  на
дисплей---------------------------
                for  (byte  i  =  0;  i  <  coin_amount;  i  +=  2)  {  //
відобразяться на дисплеї межі діапазонів розпізнавання для кожного
номіналу монет
                  lcd.clear();
```

```
                lcd.setCursor(0, 0);
                lcd.print(coin_value[i]);    lcd.print(":");    //
номінал монети
                lcd.print(coin_signal_max[i]);     lcd.print("~");
lcd.print(coin_signal_min[i]); // та її діапазон
                lcd.setCursor(0, 1);
                if (i < coin_amount - 1) {
                  lcd.print(coin_value[i  +  1]);  lcd.print(":");
// номінал монети
                  lcd.print(coin_signal_max[i        +        1]);
lcd.print("~");  lcd.print(coin_signal_min[i  +  1]);  //  та  її
діапазон
                }
                delay(5000); // з паузою в 5 секунд, послідовно
відображаються результати калібрування за кожну монету
              }  //---------------Кінець виведення результатів
калібрування монет, що зберігаються в пам'яті---------------

                standby_timer = millis() ;  //скинути таймер
                break;
            } // дужка якщо кнопка «прокинутися» як і раніше
натиснута
            //    КІНЕЦЬ    ВИВЕДЕННЯ    СЛУЖБОВОЇ    ІНФОРМАЦІЇ
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~
          } // дужка if (millis() - standby_timer > 5000){ коли
якщо контакти «прокинутися» замкнуті > 5 секунд, то виведення
статистики }
        } // дужка while (!digitalRead(button)) { коли натиснуто
кнопку «прокинутися» }
      } // дужка ГОЛОВНОГО циклу while (true) {безперервного
циклічного сканування датчиків монет і купюр}
    }  //дужка циклу void loop() { }
```

```
// ФУНКЦІЇ

// функція сну
void good_night() {
  // перед тим як піти спати, записуємо в EEPROM нові отримані дані про кількість монет за адресами починаючи з 0-го
  for (byte i = 0; i < coin_amount; i++) {
    EEPROM.updateInt(i * 2, coin_quantity[i]); // EEPROM.update обновляет байт данных
    // та же запись EEPROM.write, но лучше, заменяет значение в ячейке, если её старое содержимое отличается от нового
  }
  for (byte i = 0; i < banknote_amount; i++) {
    EEPROM.updateInt(60 + i * 2, banknote_quantity[i]); // EEPROM.update оновлює байт даних
    // той самий запис EEPROM.write, але краще, замінює значення в комірці, якщо її старий вміст відрізняється від нового
  }
  sleep_flag = true;
  // вимкнути живлення з дисплея і всіх датчиків
  digitalWrite(disp_power, LOW);
  digitalWrite(LEDpin, LOW);

 digitalWrite(IRpin, LOW);
  digitalWrite(motor_power, LOW);
  digitalWrite(sensor_color_power, LOW);
  delay(100);
  LowPower.powerDown(SLEEP_FOREVER, ADC_OFF, BOD_OFF);
}

// Прокидаємося за перериванням (ця функція - обробник переривання)
```

```
    // Спрацьовує під час замикання кнопок купюроприймача і монетоприймача (замикання контактів) «уві сні», а також в активному режимі роботи скарбнички

    void wake_up() {
      // повертаємо живлення на дисплей і датчик
      digitalWrite(disp_power, HIGH);
      digitalWrite(LEDpin, HIGH);
      digitalWrite(IRpin, HIGH);
      standby_timer = millis() - (displ_coin_value_time + 4) ; // не обнуляємо таймер, а скидаємо таймер, до значення
      // «таймера тривалості часу виведення номіналу монет», щоб не було повторного виведення інформації на дисплей після прокидання

    }

    // Функція читання внутрішньої опорної напруги
    float readVcc() {
      byte i;
      float result = 0.0;
      float tmp = 0.0;

      for (i = 0; i < 5; i++) {
        // Read 1.1V reference against AVcc

      // set the reference to Vcc and the measurement to the internal 1.1V reference - analogReference(internal)
    #if defined(__AVR_ATmega32U4__) || defined(__AVR_ATmega1280__) || defined(__AVR_ATmega2560__)
        ADMUX = _BV(REFS0) | _BV(MUX4) | _BV(MUX3) | _BV(MUX2) | _BV(MUX1);
    #elif defined (__AVR_ATtiny24__) || defined(__AVR_ATtiny44__) || defined(__AVR_ATtiny84__)
```

```
        ADMUX = _BV(MUX5) | _BV(MUX0);
    #elif defined (__AVR_ATtiny25__) || defined(__AVR_ATtiny45__)
|| defined(__AVR_ATtiny85__)
        ADMUX = _BV(MUX3) | _BV(MUX2);
    #else
        // works on an Arduino 168 or 328
        ADMUX = _BV(REFS0) | _BV(MUX3) | _BV(MUX2) | _BV(MUX1);
    #endif

        delay(3); // Wait for Vref to settle
        ADCSRA |= _BV(ADSC); // Start conversion
        while (bit_is_set(ADCSRA, ADSC)); // measuring

        uint8_t low  = ADCL; // must read ADCL first - it then
locks ADCH
        uint8_t high = ADCH; // unlocks both

        tmp = (high << 8) | low;
        tmp = (vcc_const * 1023.0) / tmp;
        result = result + tmp;
        delay(5);
      }
      result = result / 5;
      return result;
    }

    // Функція вимірювання напруги на акумуляторній батареї та
розрахунку залишку заряду акумуляторної батареї
    float BattCharg() {
      // зчитуємо точне значення напруги з піна A1 5 разів, де
знаходиться наш вольтметр з дільником напруги

      battery_voltage = 0.0;  // напруги на акумуляторній батареї
      for (byte i = 0; i < 5; i++) {
```

```
        battery_voltage = battery_voltage + analogRead(VoltmetrBatt); // сума 5 вимірів
        delay(10);
      }
      // Розрахунок напруги батареї у вольтах як середньоарифметичне 5 вимірювань з урахуванням опорного VCC і резисторів R3 і R4
      battery_voltage = ((battery_voltage / 5) * VCC / 1024) / (r4 / (r3 + r4)); // підсумкова формула розрахунку напруги батареї вольт
      // відображення залишку заряду у відсотках від повної ємності! Інтерпольовано вручну за графіком розряду літієвого акумулятора
      battery_charge = battery_voltage * 100;
      if (battery_charge > 410)
        battery_charge = map(battery_charge, max_battery_voltage * 100, 410, 100, 95);
      else if ((battery_charge <= 410) && (battery_charge > 405))
        battery_charge = map(battery_charge, 410, 405, 95, 76);
      else if ((battery_charge <= 405) && (battery_charge > 360))
        battery_charge = map(battery_charge, 405, 360, 76, 29);
      else if ((battery_charge <= 360) && (battery_charge > 340))
        battery_charge = map(battery_charge, 360, 340, 29, 16);
      else if ((battery_charge <= 340) && (battery_charge > 310))
        battery_charge = map(battery_charge, 340, 310, 16, 5);
      else if (battery_charge <= 310)
        battery_charge = map(battery_charge, 310, min_battery_voltage * 100, 5, 0);
```

```
      battery_charge = constrain(battery_charge, 0, 100); //заряд батареї визначений областю допустимих значень 0~100% constrain
      // Кінець інтерполяції за графіком
      return battery_charge;// «return» припиняє обчислення у функції BattCharg() і повертає значення battery_charge у місце виклику функції
      //Кінець розрахунку залишку заряду акумуляторної батареї. Залишок у % підрахований і повернутий у місце звернення до функції BattCharg()
    }

    // Функція розрахунку сумарного часу активної роботи МК
    // t is time in seconds = millis()/1000; Час t у секундах від початку початку запуску програми
    char * TimeToString(unsigned long t)
    {
      static char str[12];
      long h = t / 3600;
      t = t % 3600; //  % повертає залишок від ділення одного цілого (int) операнда на інший
      int m = t / 60;
      int s = t % 60;
      sprintf(str, "%04ld:%02d:%02d", h, m, s); // виведення сумарного часу у форматі (h, m, s)
      return str;// повертає stime, припиняє обчислення у функції та повертає значення (str) з перерваної функції в викликаючу TimeToString(millis()/1000)=str
    }
    // Функція виведення на дисплей: назви скарбнички або номіналу розпізнаної монети/банкноти,
    // накопиченої суми грошей, символу валюти, залишку заряду батареї
    void DisplayOutput()
    {
```

```
        lcd.clear();
        if (N == 10) { // стан прапорця coin_flag = false
          lcd.setCursor(1, 0); lcd.print(L"Smart копілка"); //
        }

        // якщо «N» не дорівнює 10 і прапорець фіксації прольоту монети в скарбничку - не істина, а false
        else if (N != 10 && !coin_flag) { // якщо N стала відмінна від 10 і coin_flag = false, то було впізнано купюру!
          lcd.setCursor(3,                                    0); lcd.print(print_banknote_value[N]);// вивести номінал розпізнаної банкноти

          lcd.setCursor(3, 0);
          if (N == 0 && print_banknote_value[N] == "RU") lcd.print(L"10 гривень");
          if (N == 1 && print_banknote_value[N] == "RU") lcd.print(L"20 гривень");
          if (N == 2 && print_banknote_value[N] == "RU") lcd.print(L"50 гривень");
          if (N == 3 && print_banknote_value[N] == "RU") lcd.print(L"100 гривень");
          if (N == 4 && print_banknote_value[N] == "RU") lcd.print(L"200 гривень");
          if (N == 5 && print_banknote_value[N] == "RU") lcd.print(L"500 гривень");
          if (N == 6 && print_banknote_value[N] == "RU") lcd.print(L"1000 гривень");
          // кінець
        }
        else {// в іншому випадку, пролетіла монета і розпізнана (coin_flag = true), виводимо її номінал на дисплей.
          // Раніше, у БЛОЦІ ..., ФІКСАЦІЇ ПРОЛЬОТУ І ПІДРАХУНКУ МОНЕТ, змінній «N» присвоєно номер номіналу розпізнаної монети
```

```
        lcd.setCursor(4, 0); lcd.print(print_coin_value[N]);

        lcd.setCursor(4, 0);
        if (N == 0 && print_banknote_value[N] == "RU")
lcd.print(L"1 гривня");
        if (N == 1 && print_banknote_value[N] == "RU")
lcd.print(L"10 гривень");
        if (N == 2 && print_banknote_value[N] == "RU")
lcd.print(L"50 копійок");
        if (N == 3 && print_banknote_value[N] == "RU")
lcd.print(L"5 гривень");
        if (N == 4 && print_banknote_value[N] == "RU")
lcd.print(L"2 гривні");
        if (N == 5 && print_banknote_value[N] == "RU")
lcd.print(L"1 гривня");
      }

      lcd.setCursor(0, 1); lcd.print(summ_money); //сума монет,
якщо потрібно відобразити цілі числа, то lcd.print(summ_money,0)
або ((int)summ_money)
      lcd.print(" " + currency); // виведення символів валюти
      if (round(battery_charge) < 10) {
        lcd.setCursor(14, 1);
      }
      else {
        lcd.setCursor(13, 1);
      }
      if (round(battery_charge) > 99) {
        lcd.setCursor(12, 1);
      }
      lcd.print(battery_charge, 0); lcd.print(L"%"); // відсоток
заряду батареї
    }
```

```
    // Кінець функції виведення DisplayOutput() ----------------
-------------------------------------


    //функція «Banknote_RGB_Frequency» вимірювання інтенсивності
частот за кожен колір R,G,B
    //де MyStruct це ярлик структури, яка описує тип даних

    MyStruct Banknote_RGB_Frequency() {
      digitalWrite(sensor_color_power, HIGH); // подати живлення
на датчик кольору
      //Serial.println("0-ON  Sensor  color");//  виводимо  для
контролю
      delay(3000);// пауза, даємо час подати купюру до упору, до
валиків.

      while (1) { // цикл визначення частоти за кожен колір
        // цикл виконається ОДИН раз, оскільки в кінці його
стоїть оператор «break»
        // зациклиться з безперервним виведенням частот, якщо
буде натиснута кнопка «калібрування»
        redFrequency = 0; greenFrequency = 0; blueFrequency = 0;
// обнуляємо частоту

      for (byte i = 0; i < 5; i++) { //зчитує 5 разів частоту
інтенсивності кожного кольору

          // Встановленням S2 і S3 (низький, низький) активуємо
червоні фотодіоди, щоб отримати дані для червоного кольору
          digitalWrite(S2, LOW);
          digitalWrite(S3, LOW);
          // Читання вихідної частоти для червоного кольору
          redFrequency = redFrequency + pulseIn(sensorOut, LOW);
// сума 5-ти вимірів
```

```
            delay(100);// пауза

            // Встановленням S2 і S3 (високий, високий) активуємо
зелені фотодіоди, щоб отримати дані для зеленого кольору
            digitalWrite(S2, HIGH);
            digitalWrite(S3, HIGH);
            // Читання вихідної частоти для зеленого кольору
            greenFrequency = greenFrequency + pulseIn(sensorOut,
LOW); //
            delay(100);

            // Встановленням S2 і S3 (високий, високий) активуємо
сині фотодіоди, щоб отримати дані для синього кольору
            digitalWrite(S2, LOW);
            digitalWrite(S3, HIGH);
            // Читання вихідної частоти для синього кольору
            blueFrequency = blueFrequency + pulseIn(sensorOut,
LOW); //
            delay(100);

          }  //дужка циклу for

          //  тепер знаходимо середнє арифметичне 5 вимірювань для
кожного фільтра
          redFrequency = redFrequency  / 5 ; // середнє арифметичне
червоного кольору
          greenFrequency = greenFrequency  / 5 ; // середнє
арифметичне зеленого кольору
          blueFrequency = blueFrequency  / 5 ; // середнє
арифметичне синього кольору

          // ДЛЯ НАЛАШТУВАННЯ ВСТАВИТИ БАНКНОТУ І НАТИСНУТИ КНОПКУ
КАЛІБРУВАННЯ. БЕЗПЕРЕРВНО ВИВОДЯТЬСЯ ВИМІРЯНІ ЧАСТОТИ ЗА КОЖЕН
ФІЛЬТР R,G,B
```

```
        if (digitalRead(calibr_button) or
digitalRead(banknote_button)) break ;
        // Для контролю, якщо натиснуто кнопку «калібрування»,
виводимо постійно значення частоти червоного, зеленого і синього
кольору
        //виводимо одним рядком значення частоти червоного,
зеленого і синього кольору на дисплей

        lcd.clear(); lcd.setCursor(0, 0);
        lcd.print("R" + String(redFrequency) + " G" +
String(greenFrequency) + " B" + String(blueFrequency));
        blueFrequency = redFrequency + greenFrequency +
blueFrequency ; // і виводимо контрольну суму
        lcd.setCursor(0, 1); lcd.print("S" +
String(blueFrequency));
        standby_timer = millis(); //скидаємо таймер, щоб, якщо до
цього довго тримали кнопку «калібрування», не зупинився мотор,
        //зі спливом часу бездіяльності standby_timer(таймера
сну), коли він ще протягує купюру в скарбничку
      } //дужка циклу while (1) визначення інтенсивності частоти
за кожен колір R,G,B
      //далі, «return» закінчує і повертає відразу три значення
частот у місце виклику функції, ці три змінні
      //у вигляді структури, з типом даних «int». Структура під
ярликом «MyStruct» була описана раніше на початку скетчу
      return (MyStruct) {
        redFrequency, greenFrequency, blueFrequency
      };
    }
```

БІБЛІОГРАФІЧНА ДОВІДКА

Тема бакалаврської роботи: ***Розробка багатофункціональної автономної монетниці з LCD-дисплеєм на базі мікроконтролера ATmega328***

Обсяг пояснювальної записки 81 аркуш.

5 таблиць;

35 рисунків;

1 додаток.

Дата завершення роботи: *12 червня 2025р*.

Підпис студента-дипломника _____Сеньків Б.М.