Лабораторна робота №3

СОРТУВАННЯ ТЕХНОЛОГІЧНИХ ОБ'ЄКТІВ ЗА КОЛЬОРОМ ЗА ДОПОМОГОЮ РОБОТА *Braccio* ТА ПРОГРАМНОГО СЕРЕДОВИЩА *Arduino IDE*

Мета роботи – розширити практичні навички програмування в середовищі *Arduino IDE* з використанням датчика кольору у комплекті з роботом *Braccio*

3.1. Теоретичні відомості

3.1.1. Основні відомості про світло

Світло – це електромагнітні хвилі видимого спектру. До видимого діапазону належать електромагнітні хвилі в інтервалі частот, що сприймаються людським оком (7,5×1014 - 4×1014 Гц), тобто з довжиною хвилі від 390 до 750 нанометрів (рис. 3.1)

У фізиці термін "світло" має дещо ширше значення і є синонімом до оптичного випромінювання, тобто включає в себе інфрачервону та ультрафіолетову області спектру.

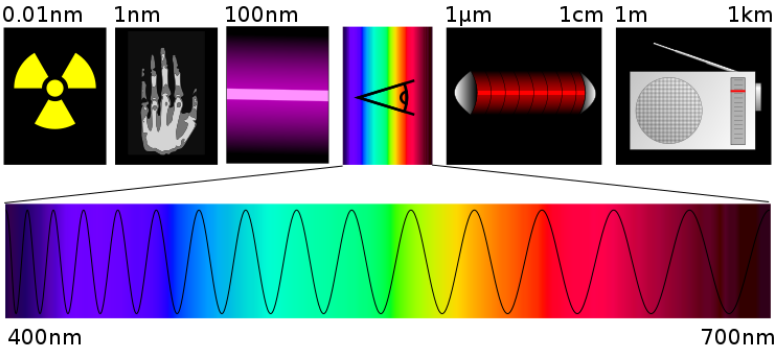

Рис. 3.1. – Видиме світло на електромагнітній шкалі

Як і будь-які інші електромагнітні хвилі, світло характеризується частотою, довжиною хвилі, поляризацією та інтенсивністю. У вакуумі світло розповсюджується зі сталою швидкістю, яка не залежить від системи відліку - швидкістю світла. Швидкість поширення світла в речовині залежить від властивостей речовини і загалом менша від швидкості світла у вакуумі. Довжина хвилі зв'язана з частотою законом дисперсії, який також визначає швидкість поширення світла в середовищі.

Взаємодіючи з речовиною, світло розсіюється, поглинається і відбивається. При переході з одного середовища в інше змінюється швидкість розповсюдження світла, що призводить до його заломлення.

Поряд із заломленням на межі двох середовищ світло частково відбивається. Заломлення та відбиття світла використовується в різноманітних оптичних приладах: призмах, лінзах, дзеркалах, що дозволяють формувати зображення.

3.1.2. Основі відомості про датчик кольору *RGB*

Датчик кольору дозволяє визначати колір поверхні об’єкта, від якого відбивається світло. Біле світло складається з усіх кольорів веселки (весь діапазон довжин хвиль світла). Коли світло падає на поверхню, деякі кольори поглинаються, а деякі відбиваються. Людське око сприймає відбитий від об'єкта колір. Для вимірювання та визначення кольору конкретного об’єкту за допомогою датчика необхідно виміряти інтенсивність різних довжин хвиль світла, відбитого від поверхні. За допомогою світлочутливого датчика (фоторезистору або фотодіоду) можна визначити інтенсивність падаючого на нього світла, тобто інтенсивність конкретних довжин хвиль, які “відповідають” за конкретний колір.

В даній лабораторній роботі використовується датчик кольору *RobotDyn APDS-9960* типу *RGB* (*Red, Green, Blue*). Датчик може визначати відтінки трьох основних кольорів: червоного, зеленого і синього. Основні технічні характеристики *RGB*-датчика показані у табл. 3.1.

Таблиця 3.1. – Характеристики датчика кольору RobotDyn APDS-9960

№	Найменування	Параметр	Величина
1	Інтерфейс передачі даних	тип	I2C
2	Напруга живлення	В	3,3
3	Робоча відстань детектування	мм	5 – 100
4	Споживання енергії	мкА	1,0
5	Кількість фотодіодів	шт	4

На рис. 3.2 показаний зовнішній вигляд датчика кольору моделі *RobotDyn APDS-9960*.

Рис. 3.2. – Зовнішній вигляд датчика кольору RobotDyn APDS-9960

Датчик має 5 виходів: *VCC, GND, INT, SDA, SCL*. Виходи продемонстровані на рис. 3.3.

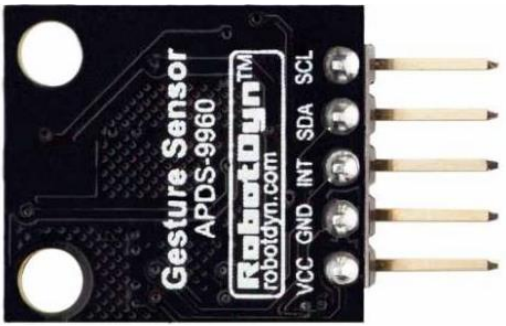

Рис. 3.3. – Виходи датчика кольору RobotDyn APDS-9960

Нижче подана інформація про виходи датчика *RobotDyn APDS-9960:*

VCC – вихід живлення датчика, робоча напруга 3,3 В, (вхід плюса);

GND – земля або мінус підключення живлення;

INT – вихід переривання, вмикається при подачі низького сигналу (логічний рівень *LOW*);

SDA – вихід для роботи з терміналом входу-виходу послідовних даних по *I2C* шині (вхід / вихід серійних даних для *I2C*-шини);

SCL – вихід для тактових сигналів для передачі даних по інтерфейсу *I2C*.

Нище на рис. 3.4 представлена схема підключення датчика до плати *Arduino Uno*:

VCC (червоний провідник) підключений до виходу *3,3 V* на платі *Arduino Uno*;

GND (чорний провідник) до ***GND***;

SDA (жовтий провідник) до ***SDA***;

SCL (синій провідник) до ***SCL***.

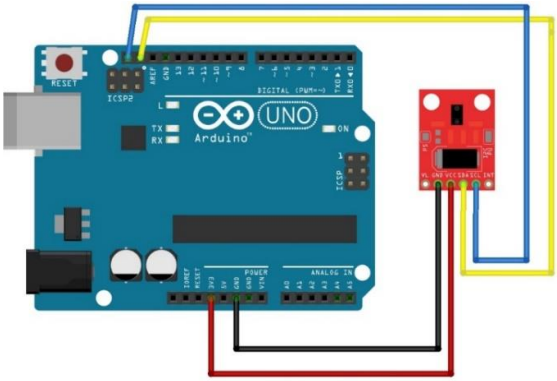

Рис. 3.4. – Схема підключення датчика кольору RobotDyn APDS-9960 до плати Arduino Uno

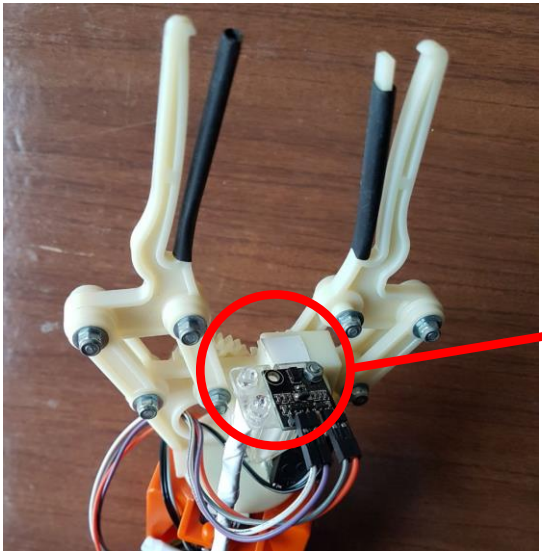

Рис. 3.5. – Розміщення датчика кольору RobotDyn APDS-9960 на роботі Tinker Kit Braccio

На роботі моделі *TinkerKit Braccio* датчик кольору *RobotDyn APDS-9960* закріплений на схваті (рис. 3.5.).

3.2. Програмні складові мови С++

Нижче представлені дані, що використовуються при програмуванні плати *Arduino Uno* і доповнюють інформаційну складову п. 3.5.

3.2.1. Поняття функції мови С++

Функція – це іменована послідовність операцій, яка знаходиться поза інших функції і може бути використана в будь-якій іншій ***функції.***

Функції дозволяють розділити програму на кілька підпрограм, які в сукупності виконують поставлену задачу. Також функції можна багатократно використовувати, що значно скорочує розмір коду програми.

Функція в програмуванні – фрагмент програмного коду, до якого можна звернутися з іншого місця програми. З іменем функції нерозривно пов'язана адреса першої інструкції (оператора), що входить до функції, яка передає керування при зверненні до функції. Після виконання функції управління повертається назад до адреси повернення – точка програми, де ця функція була викликана.

Функція може приймати параметри і повинна повертати деякі значення, можливе значення може бути пустим. Функції, які повертають пусте значення, часто називають процедурами. У деяких мовах програмування оголошень функцій і процедур мають різні синтаксиси, зокрема, можуть використовуватися різні ключові слова.

3.2.2. Визначення функції в програмі

Синтаксис (структура написання елемента коду) при написанні УП з врахуванням особливостей мови С++ наступний:

Тип значення (те значення, яке повертається) ***ім'я_функції*** (тип ім'я_змінної_1, тип им'я_змінної_2);

{
тіло функції;
};

Приклад синтаксису функції представлений на рис. 3.6.

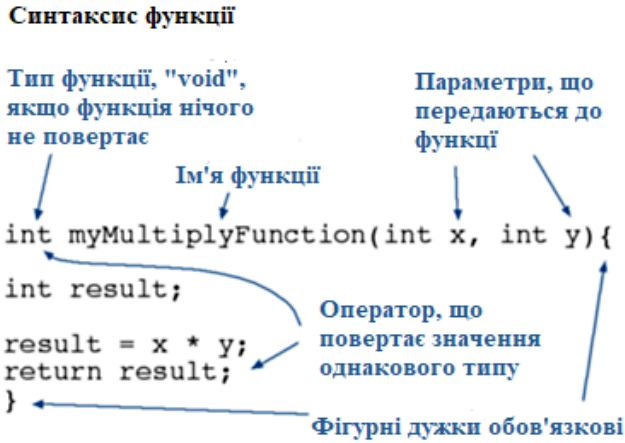

Рис. 3.6. – Синтаксис функції при програмування Arduino Uno

Сама функція може приймати значення певного типу. Цей тип вказується перед ***ім'ям функції***. Значення, яке приймає функція, пишеться після слова *return* і називається значенням, що повертається. Якщо функція не має значення, що

повертається, а просто виконує будь-яку операцію, то перед іменем функції вказується слово *void.*

Функція может мати параметри. Це ті значення, які в ході виконання можуть використовуватися функцією і які передаються до неї з головної функції (у нашому випадку використовується функція *loop ()*). Параметри функції вказуються в круглих дужках після імені функції. При визначенні функції параметри вказуються з їх типами.

3.2.3. Умовні оператори

Умовний оператор – це оператор, конструкція мови програмування, що забезпечує виконання певної команди тільки за умови істинності деякого логічного виразу, або виконання однієї з декількох команд в залежності від значення деякого виразу (рис. 3.7).

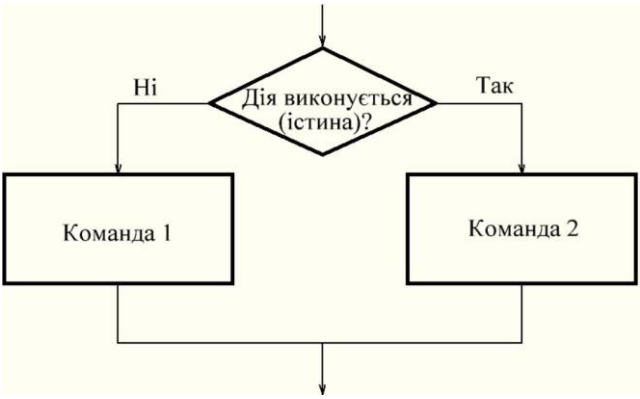

Рис. 3.7. – Приклад умовного оператора

Якщо умова вірна, тобто істинна, то виконується спеціально вказаний для цього випадку фрагмент коду. Якщо ж умова не вірна, тобто помилкова, то виконується або інша

спеціально зазначена частина коду, або не виконується нічого і робота мікроконтролера триває далі за кодом.

3.2.4. Визначення умовної операції

```
if (умова) {
   // Дії виконуються, якщо умова виконана
} else {
   // Дії виконуються, якщо умова не виконана
}
```

Такий ***умовний оператор*** зветься "*if*", що в перекладі з англійської мови означає "*якщо*" – це безпосередня вказівка оператора, потім необхідна умова в круглих дужках і повний код, що виконується при істинності цієї умови. Але у випадках, коли важливо враховувати не тільки істинність, а й хибність (невиконання поставленої умови), після фігурних дужок пишеться слово "*else*" (в перекладі з англійської означає "*інакше*") і ставляться такі ж фігурні дужки, тільки код в них буде виконуватися при хибності заданої умови.

3.2.5. Монітор порту, передача даних з плати *Arduino Uno* до ПК

Комунікація з ПК в програмному середовищі *Arduino IDE* виконується за домогою монітору порта та група функцій ***Serial***.

Serial використовується для зв'язку між платою *Arduino Uno* та комп'ютером або іншими пристроями. Всі плати сімейства *Arduino* мають щонайменше один послідовний порт (також відомий як *UART* або *USART*): серійний. Він підтримує зв'язок на цифрових контактах *0 (RX)* і *1 (TX)*, а також з комп'ютером через *USB*. Таким чином, при використанні цих

функцій використання виходів *0* і *1* для цифрового введення або виведення неможливе.

Можливим є використання вбудованого серійного монітору (або монітору порта) середовища *Arduino IDE* для комунікації з платою *Arduino Uno*. Для цього натискається кнопка серійного монітора на панелі інструментів та обирається така ж швидкість передачі, яка використовується в коді.

Послідовний зв'язок на виходах *TX / RX* використовує рівні логіки *TTL* (5 В або 3,3 В залежно від плати).

3.2.5.1. Бібліотека *Serial* для роботи з *UART Arduino*

Для роботи з апаратними *UART* контролерами в *Arduino* існує вбудована група функцій *Serial*. Вона призначена для управління обміном даними через *UART***.**

Через послідовний інтерфейс дані завжди передаються в двійковому коді.

У групі функцій *Serial* дані можуть передаватися в двох форматах:

- формат бінарного коду;
- формат *ASCII* символів.

Наприклад, монітор послідовного порту в програмі *Arduino* IDE приймає дані як *ASCII* текст. ASCII (*American Standard Code for Information Interchange*) в обчислювальній техніці це система кодів, у якій числа від 0 до 127 включно поставлені у відповідність літерам, цифрам і символам пунктуації [].

3.2.5.2. Основні функції групи *Serial*

Нижче представлені основні функції групи *Serial* з описом їх функціоналу чи призначення:

void begin (long speed) // дозволяє роботу порту *UART* і задає швидкість обміну в бод (біт за сек або **бітрейт**). Для

задання швидкості передачі даних рекомендується використовувати стандартні значення.

Serial.begin (38400); // ініціалізація порту, швидкість 38400 бод (за технічною документацією плат сімейства *Arduino*)

void end (void) // відключає порт *UART*, звільняє висновки RX і TX.

Serial.end (); // закрити порт *UART*

int available (void) // повертає кількість байт, прийнятих послідовним портом і записаних в буфер. Буфер послідовного порту може зберігати до 64 байт. У разі порожнього буфера повертає 0

int n; // допоміжна змінна

n = Serial. available (); // прирівнює змінну *n* в число прийнятих байтів

int read (void) // повертає черговий байт з буфера послідовного порту. Якщо буфер порожній - повертає число - 1 (0xffff)

receiveByte = Serial.read (); // читання байта з буфера

void flush (void) // вертає закінчення передачі даних з буфера послідовного порту

Serial.flush (); // очікування закінчення передачі

print () // виводить дані через послідовний порт *UART* у вигляді *ASCII* символів. Функція має різні форми виклику для різних форматів і типів даних:

print (char d) // якщо аргумент типу *char* виводить в порт код символу

char d = 83; // ініціалізація змінної *d* типу *char*, яка дорівнює 83 (має значення 83)

Serial.print (d); // виводить код для символу *d*, так як *char d* = 83 (див. вище), тому код відповідає значеню 83.

Serial.print ('S'); // виводить один символ (букву) *S*

print (byte d) // дані типу *byte* виводяться двійковим кодом числа

byte d = 83; // ініціалізація змінної *d* типу *byte*

Serial.print (d); // виводить код для символу *d* (*byte d* = 83) у двійковій системі

Serial.print (byte (83)); // виводить двійковий код для значення *byte (83)*

print (* str) // якщо аргумент покажчик ("*") на масив або рядок, то масив або рядок побайтно передається в порт, **покажчик** – це змінна, яка містить адресу іншої змінної в пам'яті. Наприклад, якщо змінна *a* містить адресу змінної *b*, то це означає, що змінна *a* вказує на змінну *b*

char letters [3] = {65, 66, 67};

Serial.print ("Букви"); // виводить рядок "*Букви*"

Serial.print (letters); // виводить рядок з 3 символів з кодами 65, 66, 67

int write () // виводить двійкові дані через послідовний порт *UART*. Повертає кількість переданих байтів. Функція має різні форми виклику для різних форматів і типів даних:

int write (val) // передає байт

Serial.write (83); // передає байт 83

int write (str) // передає рядок як послідовність байтів

int bytesNumber; // число байтів

bytesNumber = Serial.write ("Рядок"); // передає рядок "*Рядок*", повертає довжину рядка

int write (* buf, len) // передає байти з масиву, число байтів – *len* (довжина масиву)

char buf = "Рядок";

Serial.write (buf, 3); // виводить рядок "*Стор*"

3.3. Приклад виконання завдання та програми пошуку об’єкта за кольором

3.3.1. Структура стенду та рекомендації до роботи зі стендом

Для даної лабораторної роботи використовується робот №1 моделі *TinkerKit Braccio* (див. рис. 2.29), на якому закріплений датчик кольору *RobotDyn APDS-9960*. На стенді певним чином позначене координатне поле радіального виду (див. рис. 3.8), яке представлене у вигляді п’яти окремих кіл, які позначені відповідними символами A, B, C, D, E і мають кутову протяжність від 0° до 180° відповідно з робочою зоною робота.

Позиції ТО та точок вивантаження задається індивідуальними варіантами для виконання лабораторної роботи, а визначаються конкретною радіальною дугою (A, B, C, D, E) та відповідним кутом. Розміщення ТО виконується таким чином, щоб точка перетину діагоналей площини ТО направленої вгору, була співвісна на перетині радіальної дуги та конкретного кута відносно робота (див. рис. 3.3).

В наведеному нижче прикладі надано код (текст УП) саме з цими позиціями для варіанту-прикладу №***. Три ТО розміщаються на визначених позиціях так, щоб різнокольорові сторони ТО були направлені вертикально вгору. Позиції на координатній сітці ТО №1 – C 20°, ТО №2 – C 40°, ТО №3 – C 60°, а позиції ТВ №1 та ТВ №2 відповідно C 110° та C 160°, розміщення по точці перетину діагоналей опорної площини ТО. Послідовність кольорів вертикально направлених сторін кубів обираються студентом довільно, але з врахуванням даних табл. 3.2 (див. далі).

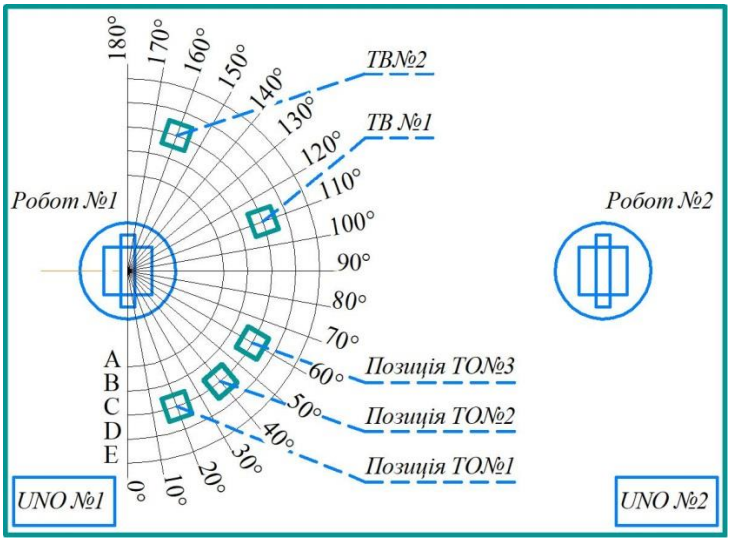

Рис. 3.8. – Структурна схема лабораторного стенду з активним роботом №1 з точками ТО та ТВ

Перед початком роботи необхідно впевнитись, що ТО розташовані правильно і не виходять за визначену розмітку. Положення ТО в робочому просторі робота можуть змінюватись. Такі положення ТО характеризують так зване статичне упорядковане технологічне середовище.

Рекомендації щодо освітлення:

УВАГА!

Робота датчику кольору моделі *RobotDyn APDS-9960* залежить від зовнішнього освітлення. Тому:

- не надавати додаткового зовнішнього освітлення (ліхтарі та інші спрямовані промені світлового потоку);

- використовувати стенд при нормальному денному освітленні або з увімкненим світлом у лабораторіях.

3.3.2. Приклад завдання та приклад його виконання

3.3.2.1. Загальні положення

Неоюхідно: запрограмувати робот *TinkerKit Braccio* на пошук об'єкта за визначеним кольором згідно з варіантом ***, наведеним нижче. При цьому пошук кольорів сторін ТО виконується за всіма трьома позиціями ТО №1, ТО №2 та ТО №3.

Варіант №	Позиція ТО№1	Позиція ТО№2	Позиція ТО№3	ТВ №1 та колір	ТВ №2 та колір
***	С 20°	С 40°	С 60°	С 110° червоний	С 160° зелений

У даному прикладі представлена програма пошуку ТО за кольором з управлінням з монітору порту. У спеціальному рядку команд оператор прописує команду-символ *r*, *g* чи *b*, що відповідає відповідному кольору (*r* – червоний, *g* – зелений, *b* – синій). Робот проходить три заздалегідь визначені точки розташування ТО та визначає їх колір за допомогою датчика. На заданий колір робот проходить всі три точки незалежно чи було знайдено ТО потрібного кольору. При виявленні ТО відповідного кольору робот захоплює ТО схватом і переміщує його у точку вивантаження №1 (ТВ №1) згідно з рис 3.9. При закінченні одного проходу пошуку ТО за кольором робот переміщується у вихідне положення і очікує команду-символ на наступний пошук ТО з відповідним кольором.

3.3.2.2. Виконання завдання

Завдання виконується наступною послідовністю кроків **К**:
К1. запустити додаток (програмне середовище ***Arduino IDE***);

К2. створити новий скетч (файл > новий, або *Ctrl + N*);
К3. підключити бібліотеки за допомогою команди (*#include< **Braccio.h** >, #include< **Servo.h** >* та *#include< **"Adafruit_APDS9960.h"** >*);
посилання на бібліотеку для датчика кольору [https://github.com/adafruit/Adafruit_APDS9960];
К4. ініціалізувати окремі серводвигуни під бібліотеку "*Servo.h*" за допомогою команди *Servo*:
Servo base; // серводвигун М1, ланка L2 (основа);
Servo shoulder; // серводвигун М2, ланка L3 (плече);
Servo elbow; // серводвигун М3, ланка L4 (лікоть);
Servo wrist_rot; // серводвигун М4, ланка L5 (вертикальний зап'ясток);
Servo wrist_ver; // серводвигун М5, ланка L6 (обертальний зап'ясток);
Servo gripper; // серводвигун М6, ланка L8 (схват);
К5. ініціалізувати ***Servo gripper*** вихід №13 плати для індикації стану комунікації плати *Arduino Uno* з ПК за допомогою команди *const int ledPin = 13*;
К6. визначити режим роботи виходу №13 плати за допомогою команди *pinMode(ledPin, OUTPUT)*;
К7. ініціалізувати бібліотеку для ПР за допомогою функції ***void setup()*** та команди ***Braccio.begin();***
К8. виконати перевірку підключення датчика кольору до плати за допогою команди *if(!apds.begin())* у функції ***void setup();***
К9. ініціалізувати бібліотеку датчика кольору за допомогою функції ***void setup()*** та команди ***apds.enableColor(true);***
К10. створити функції ***void*** *start_position()* (стартова позиція робота) та ***void*** *red()* (функція пошуку об'єкта червоного кольору);
К11. створити функцію ***void loop ()*** та організувати в ній роботу з монітором порту. Організувати виклик функцій

void *start_position()*, ***void*** *red()*та ***void*** *green()* через символьні команди в моніторі порту за допомогою функцій ***Serial.available*** та ***Serial.read*** ***;***

К12. підключити плату *Arduino Uno* до ПК та обрати відповідний *COM*-порт;

К13. завантажити програму в *Arduino Uno*;

К14. перевірити роботу коду на стенді.

3.3.2.3. Структура програми пошуку ТО за кольором

У нижче наведеній схемі (див. рис. 3.10) показані основні елементи програмного коду пошуку ТО за кольором. Кожний блок є невід'ємною складовою програми і виконує свою функцію. Блоки між собою взаємопов'язані складними непослідовними зв'язками, але їх послідовність в блок структурі є важливою вимогою мови програмування, тобто при порушенні послідовності, наданої в схемі, програма не буде скомпільована в середовищі програмування *Arduino IDE*.

Нижче описано кожен блок окремо.

Блок ***#include*** **–** це особливий елемент в програмному, де закріплюються бібліотеки та задаються змінні для подальшої роботи в окремих функціях. Завдяки бібліотекам можливо використовувати спеціальні пристрої, наприклад, датчики, драйвери, GSM модулі тощо, або використовувати плати сімейства *Arduino* для особливих цілей. В даній лабораторній роботі в блоці *#include* використовуються бібліотеки для драйвера *Braccio Shield* та для датчика кольору. І спеціальна змінна **int incomingByte,** яка потрібна для організації комунікації комп'ютера з платою.

У блоці ***void setup*** встановлюються режими роботи цифрових виходів плати, які можуть працювати як входи, або як виходи, тобто сприймати сигнали або видавати сигнали. Також

в даному елементі програмного коду виконується перевірка підключення датчика до плати та запуск бібліотеки для нього.

Блоки ***void red, void green*** та ***void blue*** ідентичні за своїм змістом, різниця лише в тому що перший створений для пошуку ТО червоного кольору, другий для пошуку ТО зеленого кольору, а – третій для пошуку ТО синього кольору. Ці блоки складаються з пропису кутів серводвигунів для кожної окремої ланки для забезпечення переміщення схвату між ТО та умовами для виконання процесу сортування ТО за кольором. Наприклад, за умови **if (r>g && r>b)** то схват захоплює ТО та переміщує його в точку вивантаження №1.

Ідентична умова використовується для виявлення ТО синього та зеленого кольору, а саме **if (b>r && b>r)** та **if (g>r && g>b).** Знак **&&** означає виконання двох умов одночасно.

Блок ***void loop*** є основним елементом програмного коду. При цьому викликаються інші елементи та забезпечується комунікація ПК з платою. Ця частина коду працює в циклі, тобто постійно виконується перевірка виконання умов, а у випадку їх відпрацювання починається нова перевірка. Завдяки цьому оператор може подати команду на пошук ТО, після чого може виконати таку операцію безліч разів.

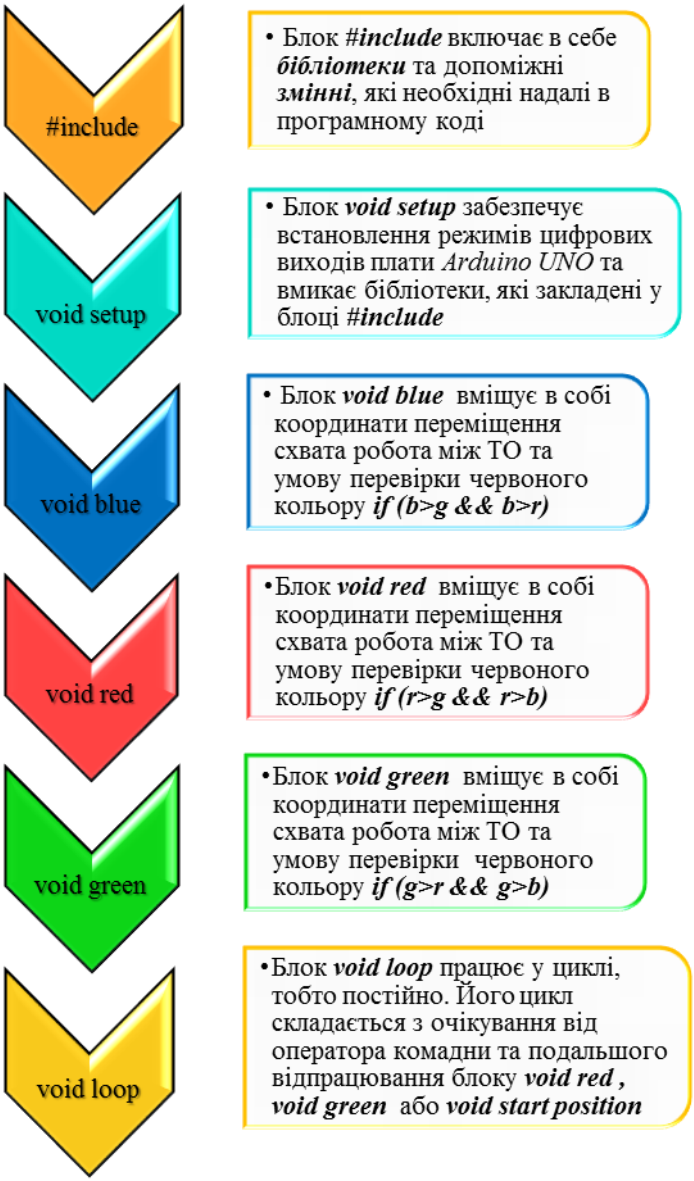

Рис. 3.9. – Структурна схема послідовності виконання складових елементів коду УП функціонування робота

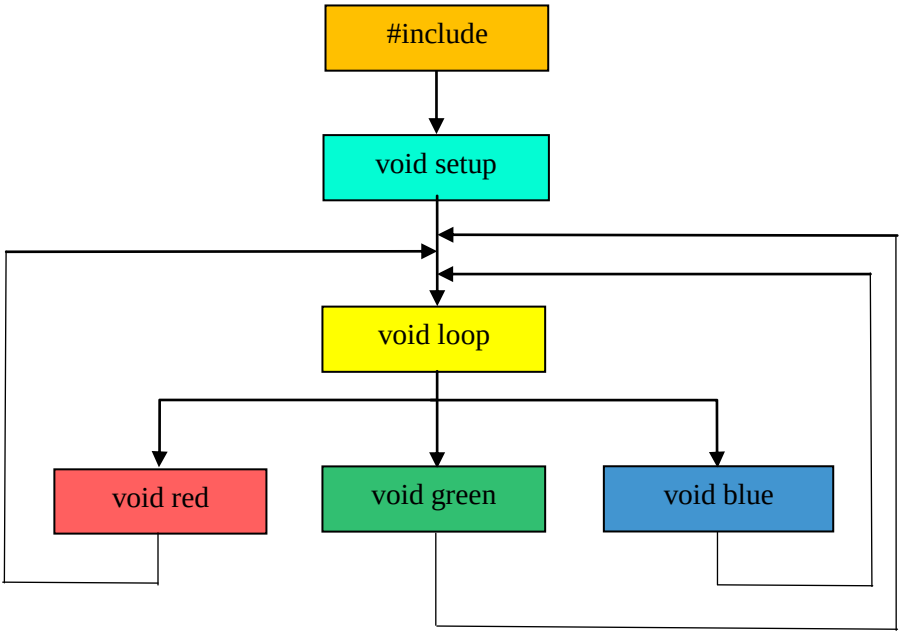

Рис. 3.10. – Схема-алгоритм складових коду УП функціонування робота Braccio

3.3.2.4. Текст УП (програмний код) за прикладом варіанту ***, п. 3.3.2

Нижче представлено програмний код наведеного в п. 33 нижче прикладу. При цьому кольоровий фон відповідає кольорову фону блоків на схемі за рис. 3.9.

```
// Блок #include
#include "Adafruit_APDS9960.h" // підключення бібліотеки для датчика кольору, доступ бібліотеки за посиланням [https://github.com/adafruit/Adafruit_APDS9960];
Adafruit_APDS9960 apds; // ініціалізація функцій бібліотеки датчика кольору
#include <Braccio.h> //підключення бібліотеки Braccio.h
#include <Servo.h> //підключення бібліотеки Servo.h
Servo base; // серводвигун M1 ланки L2 (база)
Servo shoulder; // серводвигун  M2 ланки L3 (плече)
Servo elbow; // серводвигун M3 ланки L4 (лікоть)
Servo wrist_rot; // серводвигун M4 ланки L5 (вертикальний зап'ясток)
Servo wrist_ver; // серводвигун M5 ланки L6 (обертальний зап'ясток)
Servo gripper; // серводвигун M6 ланки L8 (схват)
const int ledPin = 13; // визначення LED виходу (13) Arduino UNO для індикації стану його комуніцкації з ПК
int incomingByte; // змінна комунікації з ПК
```

```
// Блок void setup

void setup() { // встановлення режиму виходів та вмикання бібліотеки Braccio
  pinMode(ledPin, OUTPUT); // встановлення LED виходу на посилання сигналу
  Braccio.begin(); // вмикання бібліотеки Braccio
```

```
  Serial.begin(115200); // встановлення швидкості передачі даних між платою Arduino Uno і ПК, що визначено умовами роботи датчика кольору RGB

if(!apds.begin()){ // перевірка підключення датчика кольору
    Serial.println("Problem: failed to initialize device! Please check your wiring.");
  }
  else Serial.println("Device initialized!"); // текст-індикатор, з'являється на моніторі порту при успішному під'єднані датчика до плати
  apds.enableColor(true); // вмикання датчика кольору
}
```

```
// Блок void start

void start_position(){ // стартова позиція ПР, в даному випадку як приклад
  Braccio.ServoMovement(25,          135,  125, 145, 180, 125,  10);
// підхід схвата робота до позиції ТО №1
  delay(1000); // затримка часу в 1 секунду
  Braccio.ServoMovement(25,          90,  100, 175, 172, 90,  10);
 // підхід схвата робота до позиції ТО №2
  delay(1000); // затримка часу в 1 секунду
  Braccio.ServoMovement(25,          60,  105, 165, 170, 55,  10);
// Переміщення схвату з позиції ТО №2 до позиції ТО №3
  Braccio.ServoMovement(25,          45,  125, 145, 180, 32,  10);
// підхід робота до позиції ТО №3
  delay(1000); // затримка часу в 1 секунду
 }
```

```
// Блок void red

void red(){ // функція пошуку червоного куба
```

```
  uint16_t r, g, b, c; // ініціалізація допоміжних змінних r, g, b, c (де r – червоний, g – зелений, b – синій,  c – прозорість )
  Braccio.ServoMovement(25,          142,  111, 170, 180, 130,  10);  // підхід робота до позиції ТО №1
   delay(1000); // затримка часу в 1 секунду
  Braccio.ServoMovement(25,          132,  111, 170, 180, 130,  10);  // підхід робота до позиції ТО №1
  delay(1000); // затримка часу в 1 секунду
   apds.getColorData(&r, &g, &b, &c); // перевірка допоміжних змінних r, g, b, c     digitalWrite(ledPin, HIGH); // вмикання світлодіоду
   delay(1000); // затримка часу в 1 секунду
   // показання датчика по логічним рівнями
   Serial.print("red: "); // вивід у порт повідомлення
   Serial.print(r); // вивід у порт повідомлення
   Serial.print(" green: "); // вивід у порт повідомлення
   Serial.print(g); // вивід у порт повідомлення
   Serial.print(" blue: "); // вивід у порт повідомлення
   Serial.print(b); // вивід у порт повідомлення
   Serial.print(" clear: "); // вивід у порт повідомлення
   Serial.println(c); // вивід у порт повідомлення
   Serial.println();// вивід у порт повідомлення
   if (r>g && r>b) // перевірка кольору, рівень інтенсивності випромінювання червоного кольору вищий за зелений (g) та синій (b)? Якщо вище, то виконується переміщення закріпленого в схваті ТО в ТВ №1
   {
    digitalWrite(ledPin, LOW); // вимикання світлодіоду
    Braccio.ServoMovement(25,          132,  111, 170, 180, 130,  70);
    delay(500); // затримка часу в 0,5 с
    Braccio.ServoMovement(25,          135,  80, 145, 180, 180,  70);
    delay(500); // затримка часу в 0,5 с
    Braccio.ServoMovement(15,          0,  80, 145, 180, 180,  70);
    delay(500); // затримка часу в 0,5 с
    Braccio.ServoMovement(25,          0,  90, 180, 180, 180,  70);
```

```
    // переміщення ТО №1 до зони вивантаження схват закритий
    delay(500); // затримка часу в 0,5 с
    Braccio.ServoMovement(25,          0,  90, 180, 180, 180,  40);
    // ТО №1 до зони вивантаження схват відкритий
    delay(500); // затримка часу в 0,5 с
    Braccio.ServoMovement(25,          0,  90, 130, 180, 180,  40);
    delay(500); // затримка часу в 0,5 с
    Braccio.ServoMovement(25,          0,  90, 90, 180, 180,  20);
    delay(500); // затримка часу в 0,5 с
    Braccio.ServoMovement(15,          90,  90, 90, 180, 180,  20);
    // до ТО №2
    delay(500); // затримка часу в 0,5 с
    Braccio.ServoMovement(25,          90,  100, 175, 172, 180,  10);
    delay(500); // затримка часу в 0,5 с
    Braccio.ServoMovement(15,          90,  100, 175, 172, 85,  10);
    delay(500); // затримка часу в 0,5 с
   }
   else // операція при невиконанні умови
   {
     digitalWrite(ledPin, LOW); // вимикання світлодіоду
   }
    Braccio.ServoMovement(25,          110,  102, 180, 180, 90,  10);
    delay(500); // затримка часу в 0,5 с
    Braccio.ServoMovement(25,          93,  106, 180, 180, 90,  10);
  // ТО №2
 delay(500); // затримка часу в 0,5 с
  apds.getColorData(&r, &g, &b, &c); // перевірка змінних
    digitalWrite(ledPin, HIGH); // вмикакння світлодіода
    delay(1000); // перевірка змінних r, g, b, c
    Serial.print("red: "); // вивід у порт повідомлення
    Serial.print(r); // вивід у порт повідомлення
    Serial.print(" green: "); // вивід у порт повідомлення
    Serial.print(g); // вивід у порт повідомлення
    Serial.print(" blue: "); // вивід у порт повідомлення
```

```
    Serial.print(b); // вивід у порт повідомлення
    Serial.print(" clear: "); // вивід у порт повідомлення
    Serial.println(c); // вивід у порт повідомлення
    Serial.println(); // вивід у порт повідомлення
  if (r>g && r>b) // перевірка кольору. Якщо колір ТО визначений як заданий, тоді робот переміщує ТО в ТВ №1
  {
   digitalWrite(ledPin, LOW); //вимикання світлодіоду
   Braccio.ServoMovement(25,          93,  106, 180, 180, 90,  70);
   delay(500); // затримка часу в 0,5 с
   Braccio.ServoMovement(25,          93,  80, 175, 172, 90,  70);
   delay(500); // затримка часу в 0,5 с
   Braccio.ServoMovement(25,          93,  80, 145, 172, 120,  70);
   delay(500); // затримка часу в 0,5 с
   Braccio.ServoMovement(15,          0,  80, 145, 170, 180,  70);
   delay(500); // затримка часу в 0,5 с
   Braccio.ServoMovement(25,          0,  90, 180, 180, 180,  70);
   delay(500); // затримка часу в 0,5 с
   Braccio.ServoMovement(25,          0,  90, 180, 180, 180,  40);
   delay(500); // затримка часу в 0,5 с
   Braccio.ServoMovement(25,          0,  90, 130, 180, 180,  40);
   delay(500); // затримка часу в 0,5 с
   Braccio.ServoMovement(25,          0,  90, 90, 180, 130,  20);
   delay(500); // затримка часу в 0,5 с
   Braccio.ServoMovement(15,          60,  90, 90, 180, 90,  20);
   delay(500); // затримка часу в 0,5 с
   Braccio.ServoMovement(15,          60,  90, 90, 180, 55,  20);
   delay(500); // затримка часу в 0,5 с
   Braccio.ServoMovement(25,          60,  105, 165, 170, 55,  10);
   delay(500); // затримка часу в 0,5 с
  }
  else // операція при не виконанні умови
  {
    digitalWrite(ledPin, LOW); // вимикання світлодіоду
  }
```

```
  Braccio.ServoMovement(25,          75,  105, 180, 180, 70,  10);
  Braccio.ServoMovement(25,          60,  105, 173, 180, 55,  10);  //
додаткова точка переміщення для запобігання зіткнення з ТО
  Braccio.ServoMovement(25,          50,  112, 169, 180, 46,  10);
  // позиціонування в точці розташування ТО №3
  delay(1000); // затримка часу в 1с
    apds.getColorData(&r, &g, &b, &c); // перевірка змінних
      digitalWrite(ledPin, HIGH);
      delay(1000);
      Serial.print("red: "); // вивід у порт повідомлення
      Serial.print(r); // вивід у порт повідомлення
      Serial.print(" green: ");// вивід у порт повідомлення
      Serial.print(g); // вивід у порт повідомлення
      Serial.print(" blue: "); // вивід у порт повідомлення
      Serial.print(b); // вивід у порт повідомлення
      Serial.print(" clear: "); // вивід у порт повідомлення
      Serial.println(c); // вивід у порт повідомлення
      Serial.println(); // вивід у порт повідомлення
   if (r>g && r>b) // Якщо колір ТО визначений як заданий,
тоді робот переміщує ТО в ТВ №1
    {
     digitalWrite(ledPin, LOW);
     Braccio.ServoMovement(25,          50,  112, 169, 180, 46,  70);
     delay(500); // затримка часу в 0,5 с
     Braccio.ServoMovement(25,          45,  90, 145, 180, 92,  70);
     delay(500); // затримка часу в 0,5 с
     Braccio.ServoMovement(15,          0,  90, 145, 180, 180,  70);
     delay(500); // затримка часу в 0,5 с
     Braccio.ServoMovement(25,          0,  90, 180, 180, 180,  70);
     delay(500); // затримка часу в 0,5 с
     Braccio.ServoMovement(25,          0,  90, 180, 180, 180,  40);
     delay(500); // затримка часу в 0,5 с
     Braccio.ServoMovement(25,          0,  90, 145, 180, 180,  20);
     delay(500); // затримка часу в 0,5 с
    }
```

```
    else // операція при невиконанні умови
    {
      digitalWrite(ledPin, LOW); //вимикання світлодіода
    }
 }



// Блок void green

void green(){ // функція пошуку червоного куба
  uint16_t r, g, b, c; // ініціалізація допоміжних змінних r, g, b, c (де r – червоний, g – зелений, b – синій,  c – прозорість )
  Braccio.ServoMovement(25,          142,  111, 170, 180, 130,  10);  // підхід робота до позиції ТО №1
   delay(1000); // затримка часу в 1 секунду
  Braccio.ServoMovement(25,          132,  111, 170, 180, 130,  10);  // підхід робота до позиції ТО №1
  delay(1000); // затримка часу в 1 секунду
    apds.getColorData(&r, &g, &b, &c); // перевірка допоміжних змінних r, g, b, c     digitalWrite(ledPin, HIGH); // вмикання світлодіоду
    delay(1000); // затримка часу в 1 секунду
    // показання датчика по логічним рівнями
    Serial.print("red: "); // вивід у порт повідомлення
    Serial.print(r); // вивід у порт повідомлення
    Serial.print(" green: "); // вивід у порт повідомлення
    Serial.print(g); // вивід у порт повідомлення
    Serial.print(" blue: "); // вивід у порт повідомлення
    Serial.print(b); // вивід у порт повідомлення
    Serial.print(" clear: "); // вивід у порт повідомлення
    Serial.println(c); // вивід у порт повідомлення
    Serial.println();// вивід у порт повідомлення
    if (g>r && g>b) // перевірка кольору, рівень інтенсивності випромінювання червоного кольору вище за зелений (g) та
```

```
синій (b)? Якщо вище, то виконується переміщення
закріпленого в схваті ТО в ТВ №1
  {
   digitalWrite(ledPin, LOW); // вимикання світлодіоду
   Braccio.ServoMovement(25,          132,  111, 170, 180, 130,  70);
   delay(500); // затримка часу в 0,5 с
   Braccio.ServoMovement(25,          135,  80, 145, 180, 180,  70);
   delay(500); // затримка часу в 0,5 с
   Braccio.ServoMovement(15,          0,  80, 145, 180, 180,  70);
   delay(500); // затримка часу в 0,5 с
   Braccio.ServoMovement(25,          0,  90, 180, 180, 180,  70);
   // ТО №1 до зони вивантаження схват закритий
   delay(500); // затримка часу в 0,5 с
   Braccio.ServoMovement(25,          0,  90, 180, 180, 180,  40);
   // ТО №1 до зони вивантаження схват відкритий
   delay(500); // затримка часу в 0,5 с
   Braccio.ServoMovement(25,          0,  90, 130, 180, 180,  40);
   delay(500); // затримка часу в 0,5 с
   Braccio.ServoMovement(25,          0,  90, 90, 180, 180,  20);
   delay(500); // затримка часу в 0,5 с
   Braccio.ServoMovement(15,          90,  90, 90, 180, 180,  20);
   // переміщення до ТО №2
   delay(500); // затримка часу в 0,5 с
   Braccio.ServoMovement(25,          90,  100, 175, 172, 180,  10);
   delay(500); // затримка часу в 0,5 с
   Braccio.ServoMovement(15,          90,  100, 175, 172, 85,  10);
   delay(500); // затримка часу в 0,5 с
  }
  else // операція при невиконанні умови
  {
    digitalWrite(ledPin, LOW); // вимикання світлодіоду
  }
   Braccio.ServoMovement(25,          110,  102, 180, 180, 90,  10);
   delay(500); // затримка часу в 0,5 с
   Braccio.ServoMovement(25,          93,  106, 180, 180, 90,  10);
```

```
  // ТО №2
  delay(500); // затримка часу в 0,5 с
    apds.getColorData(&r, &g, &b, &c); // перевірка змінних
      digitalWrite(ledPin, HIGH); // вмикакння світлодіода
      delay(1000); // перевірка змінних r, g, b, c
      Serial.print("red: "); // вивід у порт повідомлення
      Serial.print(r); // вивід у порт повідомлення
      Serial.print(" green: "); // вивід у порт повідомлення
      Serial.print(g); // вивід у порт повідомлення
      Serial.print(" blue: "); // вивід у порт повідомлення
      Serial.print(b); // вивід у порт повідомлення
      Serial.print(" clear: "); // вивід у порт повідомлення
      Serial.println(c); // вивід у порт повідомлення
      Serial.println(); // вивід у порт повідомлення
    if (g>r && g>b) // перевірка кольору. Якщо колір ТО
визначений як заданий, тоді робот переміщує ТО в ТВ №1
    {
      digitalWrite(ledPin, LOW); //вимикання світлодіоду
      Braccio.ServoMovement(25,          93,  106, 180, 180, 90,  70);
      delay(500); // затримка часу в 0,5 с
      Braccio.ServoMovement(25,          93,  80, 175, 172, 90,  70);
      delay(500); // затримка часу в 0,5 с
      Braccio.ServoMovement(25,          93,  80, 145, 172, 120,  70);
      delay(500); // затримка часу в 0,5 с
      Braccio.ServoMovement(15,          0,  80, 145, 170, 180,  70);
      delay(500); // затримка часу в 0,5 с
      Braccio.ServoMovement(25,          0,  90, 180, 180, 180,  70);
      delay(500); // затримка часу в 0,5 с
      Braccio.ServoMovement(25,          0,  90, 180, 180, 180,  40);
      delay(500); // затримка часу в 0,5 с
      Braccio.ServoMovement(25,          0,  90, 130, 180, 180,  40);
      delay(500); // затримка часу в 0,5 с
      Braccio.ServoMovement(25,          0,  90, 90, 180, 130,  20);
      delay(500); // затримка часу в 0,5 с
      Braccio.ServoMovement(15,          60,  90, 90, 180, 90,  20);
```

```
    delay(500); // затримка часу в 0,5 с
    Braccio.ServoMovement(15,          60,  90, 90, 180, 55,  20);
    delay(500); // затримка часу в 0,5 с
    Braccio.ServoMovement(25,          60,  105, 165, 170, 55,  10);
    delay(500); // затримка часу в 0,5 с
   }
   else // операція при не виконанні умови
   {
    digitalWrite(ledPin, LOW); // вимикання світлодіоду
   }
  Braccio.ServoMovement(25,          75,  105, 180, 180, 70,  10);
  Braccio.ServoMovement(25,          60,  105, 173, 180, 55,  10);  //
додаткова точка переміщення для запобігання зіткнення з ТО
  Braccio.ServoMovement(25,          50,  112, 169, 180, 46,  10);
  // позиціонування в точці розташування ТО №3
  delay(1000); // затримка часу в 1с
   apds.getColorData(&r, &g, &b, &c); // перевірка змінних
    digitalWrite(ledPin, HIGH);
    delay(1000);
    Serial.print("red: "); // вивід у порт повідомлення
    Serial.print(r); // вивід у порт повідомлення
    Serial.print(" green: ");// вивід у порт повідомлення
    Serial.print(g); // вивід у порт повідомлення
    Serial.print(" blue: "); // вивід у порт повідомлення
    Serial.print(b); // вивід у порт повідомлення
    Serial.print(" clear: "); // вивід у порт повідомлення
    Serial.println(c); // вивід у порт повідомлення
    Serial.println(); // вивід у порт повідомлення
  if (g>r && g>b) // Якщо колір ТО визначений як заданий,
тоді робот переміщує ТО в ТВ №1
   {
    digitalWrite(ledPin, LOW);
    Braccio.ServoMovement(25,          50,  112, 169, 180, 46,  70);
    delay(500); // затримка часу в 0,5 с
    Braccio.ServoMovement(25,          45,  90, 145, 180, 92,  70);
```

```
    delay(500); // затримка часу в 0,5 с
    Braccio.ServoMovement(15,          0,  90, 145, 180, 180,  70);
    delay(500); // затримка часу в 0,5 с
    Braccio.ServoMovement(25,          0,  90, 180, 180, 180,  70);
    delay(500); // затримка часу в 0,5 с
    Braccio.ServoMovement(25,          0,  90, 180, 180, 180,  40);
    delay(500); // затримка часу в 0,5 с
    Braccio.ServoMovement(25,          0,  90, 145, 180, 180,  20);
    delay(500); // затримка часу в 0,5 с
   }
   else // операція при не виконанні умови
   {
     digitalWrite(ledPin, LOW); //вимикання світлодіода
   }
}
```

```
// Блок void loop

void loop() {
  // змінні для збереження даних кольору
  uint16_t r, g, b, c;

while(!apds.colorDataReady()){   // затримка для ініціалізації
бібліотеки датчику кольору
    delay(5); // затримка часу в 5 мс
  }
  delay(1000);
 Braccio.ServoMovement(25,          90,  90, 90, 90, 90,  10);

 if (Serial.available() > 0)  // передача інформації в монітор
порту
     { // read the oldest byte in the serial buffer:
           incomingByte = Serial.read();
```

```
          if (incomingByte == 'R') {   // символ запуску функції
          пошуку червоного куба
          red(); // запуск функції пошуку ТО червоного кольору
          }

          if (incomingByte == 'G') {   // символ запуску функції
пошуку зеленого куба
          green(); // запуск функції пошуку ТО зеленого кольору
          }
          if (incomingByte == 'S') {   // символ запуску функції
стартової позиції
          start_position(); // запуск вказаної функції
          }
    }
}
```

3.4. Завдання та порядок виконання роботи

1. Ознайомитись та засвоїти теоретичні відомості.
2. Перевірити підключення датчика кольору *RobotDyn APDS-9960* до плати *Arduino UNO*.
3. Запустити додаток *Arduino IDE* та підключити плату *Arduino UNO* до комп'ютера за допомогою *USB*-кабелю.
4. Завантажити програму-приклад на плату *Arduino UNO* та відкрити монітор порту для комунікації з ПК (виставити *бітрейт* 115200**)**.
5. Розташувати ТО на стенді у визначених місцях (позначено на основі стенду позначками).
6. Встановити всі ланки робота у безпечне положення та подати живлення на *Braccio Shield* за допомогою блока живлення через відповідний роз'єм.
7. Дочекатись ініціалізації датчика кольору: на моніторі порту буде виведено повідомлення "Device initialized!".

8. Подати команду на запуск функції пошуку ТО прописом відповідним символом та вводу клавішою *Enter* на ПК, спостерігати за ходом виконання програми.

9. Після відпрацювання програми-прикладу визначити структуру (черговість виконання та загальний алгоритм) програми пошуку ТО за кольором відповідно до заданого варіанта, який вказаний в табл. 2.2.;

10. Створити програму (код, УП) пошуку ТО за кольором на основі коду, який наданий у прикладі п 3.3.2.2, та записати в програмне середовище *Arduino IDE*.

11. Продемонструвати відпрацювання роботом позицій, заданих за варіантом.

12. Записати відео роботи програми пошуку ТО за кольором, завантажити відео у хмарне сховище (довільний вибір) та згенерувати *QR*-код посилання на відео.

13.Оформити звіт за вимогами п. 3.6.

3.5. Варіанти індивідуальних завдань

Таблиця 3.2. – Варіанти індивідуальних завдань

Варіант №	Позиції ТО			Колір ТО та ТВ	
	ТО №1	ТО №2	ТО №3	ТВ №1	ТВ №2
1	С 0°	С 20°	С 40°	Зелений С 110°	Синій С 160°
2	С 0°	С 20°	С 40°	Червоний С 110°	Синій С 160°
3	С 0°	С 20°	С 40°	Синій С 110°	Зелений С 160°
4	С 0°	С 20°	С 40°	Зелений С 110°	Червоний С 160°
5	С 0°	С 20°	С 40°	Червоний С 110°	Зелений С 160°
6	С 0°	С 20°	С 40°	Синій С 110°	Червоний С 160°
7	С 20°	С 40°	С 60°	Зелений С 130°	Синій С 180°

Закінчення табл. 3.2

8	С 20°	С 40°	С 60°	**Червоний** С 130°	**Синій** С 180°
9	С 20°	С 40°	С 60°	**Синій** С 130°	**Зелений** С 180°
10	С 20°	С 40°	С 60°	**Зелений** С 130°	**Червоний** С 180°
11	С 20°	С 40°	С 60°	**Червоний** С 130°	**Зелений** С 180°
12	С 20°	С 40°	С 60°	**Синій** С 130°	**Червоний** С 180°
13	С 40°	С 60°	С 80°	**Зелений** С 100°	**Синій** С 150°
14	С 40°	С 60°	С 80°	**Червоний** С 100°	**Синій** С 150°
15	С 40°	С 60°	С 80°	**Синій** С 100°	**Зелений** С 150°
16	С 40°	С 60°	С 80°	**Зелений** С 100°	**Червоний** С 150°
17	С 40°	С 60°	С 80°	**Червоний** С 100°	**Зелений** С 150°

3.6. Зміст звіту

1. Назва та мета роботи.
2. Короткі теоретичні відомості про поняття “колір”.
3. Основні відомості щодо датчика кольору *RobotDyn APDS-9960.*
4. Схема, склад лабораторного стенду та опис його роботи.
5. Основні відомості про функцію та умовного оператора.
6. Основні відомості про структуру та зміст програми.
7. Індивідуальне завдання за варіантом лабораторної роботи.

8. Програма (УП, код) за варіантом індивідуальних завдань виконання лабораторної роботи з повним та деталізованим описом тексту програми.

9. Представити *QR*–посилання відеодемострації працездатності коду (УП) за варіантом індивідуальних завдань.

10. Висновки по роботі.

3.7. Контрольні питання

1. Надати короткий опис датчика *RobotDyn APDS-9960*.
2. Принцип роботи датчика кольору.
3. Поняття функції в мові С++.
4. Поняття умовного оператора.
5. Монітор порту та функція *Serial*.
6. Опис структури програми для робота *Braccio* з датчиком кольору *RGB*.