Міністерство освіти та науки України
Державний університет «Житомирська політехніка»

Кирилович В.А.
Кравчук А.Р.
Дімітров Л.В.

РОБОТЕХНІКА ТА МЕХАТРОНІКА.
Лабораторний практикум.

Навчальний посібник
для виконання лабораторних робіт з навчального курсу
«Робототехніка та мехатроніка»
для студентів всіх форм навчання
спеціальності
151 «Автоматизація та комп'ютерно-інтегровані технології»

Житомир 2021

УДК 621.9
Р58

Рекомендовано до видання за рішенням Вченої Ради
Державного університету «Житомирська політехніка»

протокол №3 від 25.06.2021р.

Рецензенти:

Пасічниик В.А. доктор технічних наук, професор, проректор з наукової роботи Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського»

Подчашинський Л.Г. доктор технічних наук, професор, завідувач кафедри метрології та інформаційно-вимірювальної техніки Державного університету «Житомирська політехніка»

Грабар І.Г. доктор технічних наук, професор, завідувач кафедри процесів, машин і обладнання Поліського національного університету

Р58 Кирилович В.А., Кравчук А.Р., Дімітров Л.В., за редакцією В.А. Кириловича. Робототехніка та мехатроніка. Лабораторний практикум: навчальний посібник для виконання лабораторних робіт з курсу ‘‘Роботехніка та мехатроніка’’ для студентів всіх форм навчання спеціальності 151 «Автоматизація та комп’ютерно-інтегровані технології»’. – Житомир: Електронне видання, 2021. – 110 с.

ISBN XXXXXXXX

Даний навчальний посібник підготовлений міжнародним колективом авторів. В ньому викладено методичні вказівки щодо виконання комплексу із чотирьох лабораторних робіт. В їх основу покладено використання лабораторної бази як щодо відомих засобів жосткої та гнучкої автоматизації вібраційних бункерних завантажувальних пристроїв – ВБЗП (лабораторна робота 1), так і використання мініатюрних роботів Tinker Kit Brassio (лабораторні роботи 2–4). При дослідження параметрів роботи ВБЗП використовується мехатронний підхід, а при програмуванні роботи вказаних роботів в різних режимах та умовах використовуються елементи сучасної інформаційно-програмно-апаратної бази.

Використання розроблених матеріалів є необхідною складовою підготовки бакалаврів, написання та захисту дипломної роботи бакалавра за спеціальністю 151.

УДК 621.9

ISBN XXXXXXXX

ЗМІСТ

ПЕРЕЛІК ПРИЙНЯТИХ СКОРОЧЕНЬ……. 4

Лабораторна робота 1.
ДОСЛІДЖЕННЯ ПРОДУКТИВНОСТІ ВІБРАЦІЙНИХ БУНКЕРНИХ ЗАВАНТАЖУВАЛЬНИХ ПРИСТРОЇВ (ВБЗП)……. 5

Лабораторна робота 2.
ОСНОВИ ПРОГРАМУВАННЯ РОБОТА Braccio В СЕРЕДОВИЩІ Arduino IDE..... 25

Лабораторна робота 3.
СОРТУВАННЯ ТЕХНОЛОГІЧНИХ ОБ'ЄКТІВ ЗА КОЛЬОРОМ ЗА ДОПОМОГОЮ РОБОТА Braccio ТА ПРОГРАМНОГО СЕРЕДОВИЩА Arduino IDE……... 62

Лабораторна робота 4.
ОСНОВИ ПРОГРАМУВАННЯ КОМПЛЕКСУ З ДВОХ РОБОТІВ Braccio В ПРОГРАМНОМУ СЕРЕДОВИЩІ Arduino IDE……... 97

ПЕРЕЛІК ПРИЙНЯТИХ СКОРОЧЕНЬ

ВБЗП – вібраційний бункерний завантажувальний пристрій

МТС – мехатронна система

О – об’єкт

ПО – пристрій орієнтування

ПР – промисловий робот

ЦВ – центр ваги

ЛАБОРАТОРНА РОБОТА №1

ДОСЛІДЖЕННЯ ПРОДУКТИВНОСТІ ВІБРАЦІЙНИХ БУНКЕРНИХ ЗАВАНТАЖУВАЛЬНИХ ПРИСТРОЇВ (ВБЗП)

Мета роботи – вивчити конструкцію ВБЗП та дослідити його продуктивність.

Прилади та пристрої – ВБЗП, заготовки (деталі), ємність для накопичення заготовок, блок живлення, обмотки електромагніту.

1.1. Теоретичні відомості

В автоматизованому виробництві називають вібраційними машини, які здійснюють між операції транспортування і автоматичне завантаження робочого обладнання виробами. До таких машин відносяться вібраційні бункерні завантажувальні пристрої (ВБЗП), які здійснюють розподіл і орієнтування штучних виробів, що надходять навалом. Застосовуються ВБЗП з несиметричним циклом коливань лотка і з симетричним (гармонійним) циклом. Рух заготовок по лотку може здійснюватися без відриву від поверхні (безвідривний режим) і в відривному режимі, при якому заготовка, частина кожного циклу рухається, не торкаючись поверхні транспортного лотка.

Основними перевагами ВБЗП є:

- відсутність тертя (довговічність), мале споживання енергії (економічність);
- придатність для автоматичного завантаження широкого асортименту деталей, в тому числі деталей з маломіцних і крихких матеріалів;
- придатність до швидкого і простого переналагодження на різні типорозміри деталей (універсальність);
- можливість швидкого, зручного регулювання продуктивності (легка керованість);
- безвідмовність в роботі.

Для створення вимушених коливань застосовують вібратори різних типів: механічні, пневматичні, гідравлічні, електромагнітні.

Робота *механічних вібраторів* заснований на принципі періодичного штовхання направляючого лотка, підвішеного на

торсіонних пружинах. Збурююча сила може створюватися ексцентриком, насадженим на вал електродвигуна.

Пневматичні поршневі вібратори складаються з циліндра поршня, що здійснює зворотньо-поступальний рух під дією стисненого повітря. Гідравлічні вібратори аналогічні пневматичним. Одні й другі під час роботи створюють сильний шум.

Електромагнітний вібратор приводиться в дію змінним або пульсуючим струмом. Як вимушене коливання в ньому діє сила електромагнітного тяжіння якоря електромагніту (ЕМ).

1.2. ВБЗП, їх види, призначення та основи розрахунків

В основному в промисловості застосовують ВБЗП, в яких об'єкти (О) переміщаються по лотку і, проходячи повз різного роду конструктивних доповнень (вирізи, виступи і т. д.), отримують певну орієнтацію. Установки такого типу застосовують для подачі О на робочі позиції металорізальних верстатів, пресів, складальних агрегатів, для подачі О до робочого місця при складанні, наповнення касет, сортування та пакування готових виробів.

ВБЗП з електромагнітним приводом умовно можна розділити на дві групи: бункерні ВБЗП зі спіральним лотком і бункерні ВБЗП з прямолінійним лотком.

Обидві групи мають різне конструктивне виконання приводу і бункера. Бункерні ВБЗП зі спіральним лотком за конструктивним виконанням електромагнітного приводу можна розділити на синхронні і роздільні.

За конструктивним виконанням бункера ВБЗП зі спіральним лотком розрізняють: з циліндричним; конічним (прямий чи зворотний конус); з комбінованим бункером (циліндричним і конічним).

У приладобудуванні найбільш поширені ВБЗП – вібробункери (рис. 1.1, *а*). У чашу 2 (бункер) з конічним дном навалом подають заготовки. По гвинтовому лотку 4, закріпленому на чаші 2, заготовки рухаються знизу вгору. Вібрації передаються лотку вібратором 1,

розташованим у корпусі. Вібрації лотка роз'єднують заготовки, що зчепилися.

Схему дії вібробункера пояснює рис. 1.1, *б*. На лотку 4 з кутом підйому β розміщується заготовка 6. Під дією ЕМ і пружних ресор 5, розташованих до вертикалі під кутом α, лоток із заготовкою одержує складні коливання (вертикальні та колові).

При втягуванні якоря ЕМ ресори згинаються, займаючи вертикальне положення (α=0). При цьому лоток 2 з деталлю 6 одержить імпульс руху зі швидкістю V, направлений під кутом α–β. При поверненні у вихідне положення якоря ЕМ рух лотка припиняється і під дією ресор лоток починає швидкий зворотний рух. Заготовки при цьому продовжують по інерції рух вперед і за рахунок вертикального імпульсу відриваються від лотка. Заготовка опускається на лоток далі свого початкового положення.

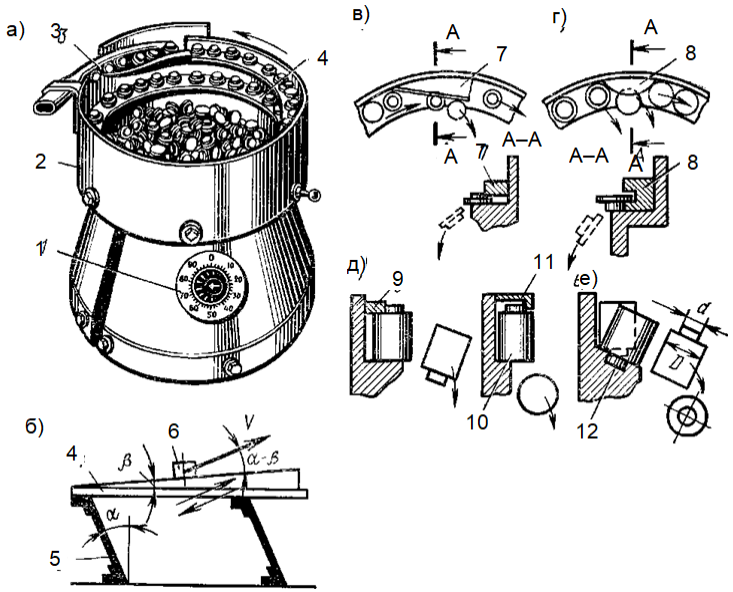


Рис. 1.1. Приклади пристроїв, робота яких базується на вібрації:

а – ВБЗП; б – віброкоток; в, г – приклади пасивної орієнтації О в ВБЗП

Для подачі заготовок, орієнтованих визначеним чином, лоток і чаша мають різні орієнтуючі пристрої 3 (рис. 1.1, *а*), розташовані ближче до виходу заготовок. Орієнтуючі пристрої конструюють з таким розрахунком, щоб заготовки, що розміщуються на лотку в необхідному положенні, переміщувались далі, а ті, що перебувають в інших положеннях, скидалися на дно до загальної маси заготовок. Наприклад, для подачі деталей типу гвинтів головкою вниз можна застосувати орієнтуючі пристрої (див. рис. 1.1 в). Косий відсікач 7 перебуває від лотка на віддалі, трохи більшій від товщини головки, щоб заготовки, які рухаються головкою вниз, проходили під відсікачем і утримувалися на лотку. Заготовки, що рухаються головкою вгору, косим відсікачем 7 скидаються на дно чаші.

Якщо потрібно одержувати з бункера О з головкою, розташованою вгору, то використовують відсікач 8, зображений на рис. 1.1, *г*. Відсікач розміщується проти вузької сторони лотка, і деталі, які рухаються головкою вниз, скидаються з лотка, тому що їхній ЦВ (центр ваги) розташовується поза лотком. У паз відсікача входить головка, повернена вгору, і тому такі заготовки подаються в робочу зону верстата чи в магазин.

Розглянемо орієнтуючі пристрої для коротких валиків з цапфою на кінці. Якщо потрібно подавати валики вертикально цапфою догори (рис. 1.1, *д*), то орієнтуючий пристрій буде складатися з двох відсікачів. Деталі, які лежать на лотку, пройшовши під відсікачем 9, потраплять під другий відсікач, що складається з неширокого лотка 10 і утримувача 11. Деталі, які розташовані цапфою вгору, утримуються на лотку козирком 11 і подадуться в робочу зону верстата. Якщо цапфа повинна бути внизу (рис. 1.1, е), то у відсікача 12 лоток виконують похилим з пазом, трохи більшим за висотою і шириною, ніж висота і діаметр цапфи.

Проектування ВБЗП з електромагнітним приводом зводиться, в основному, до визначення геометричних параметрів чаші, вибору частоти коливань бункера і розрахунку максимальної швидкості руху заготовок у магазині. Діаметр D циліндричної чаші та її висоту H визначають, виходячи з умов забезпечення безперебійної роботи наступного за ВБЗП технологічного агрегату і

розташування орієнтуючого пристрою вище рівня максимального завантаження бункера. Зазвичай приймається діаметр чаші $D=(10\div12)\cdot l_д$, але не меньше діаметра прийнятого нормалізованого вібратора ($l_д$ – довжина подаючих О, що рухаються вздовж лотка).

Оскільки у більшості випадків чашу виготовляють з алюмінію, кут прийому гвинтової доріжки магазина $\beta=3\ldots5^0$. Кут прийому доріжки і її середній діаметр визначають крок гвинтової напрямної $t=\pi D_{ср} tg\beta$. Щоб уникнути заклинювання заготовок між витками, крок гвинтової доріжки повинен бути в 1,5-2 рази більше ширини b, приблизно рівної ширині транспортуючих заготовок.

Тоді середній діаметр гвинтової напрямної дорівнює

$$D_{ср} = (1{,}5 \div 2)b/(\pi tg\beta)$$

При виборі частоти коливань необхідно пам'ятати, що максимальна швидкість транспортування при частоті 50 Гц є більшорю, ніж при 100 Гц.

При діаметрі чаші більше 200 мм використовують частоту 50 Гц.

Необхідна найвища швидкість руху О по лотку визначається за наступним виразом:

$$V = L/(T_ц C_п),$$

де L – крок між деталями (при русі впритул призначають зазвичай рівним довжині заготовки), мм;

$T_ц$ – час технологічного циклу обробки чи вимірювання деталі, с;

$C_п$ – коефіцієнт щільності потоку.

В даний час розроблені конструкції вібробункерів з електродвигунами і балансирами. Такі вібробункери простіші в налаштовуванні переміщення заготовок по лотку.

У різних галузях промисловості найбільшого поширення набули ВБЗП зі спіральним лотком, коливання захватно-орієнтуючим доріжок (лотків) в яких передаються від ресор 3, нахилених під деяким кутом до горизонту і приводяться в рух за допомогою ЕМ 8 (рис. 1.2).

БВЗП включають в себе основні конструктивні елементи: основа 1; електромагнітні приводи 2, 4, 5, 8-10; бункера 6, 7 і пружні елементи 3.

Принцип роботи ВБЗП полягає в тому, що складний коливальний рух доріжки (лотка) передається розташованому на ній

О у вертикальному та горизонтальному напрямках, в результаті чого О переміщається по доріжці з деякою швидкістю. Крім того, в процесі руху по доріжці О орієнтуються в задане положення.

На рис. 1.4. представлена конструкція ВБЗП з роздільним приводом коливань бункера 1 і основи 6 з торсіонною пружною підвіскою 5, горизонтального електромагнітного приводу 4 і з пружними елементами у вигляді пластин прямокутного перерізу 2, вертикального електромагнітного приводу коливань 3.

Відмітною особливістю конструкції ВБЗП з роздільним електромагнітним приводом (рис. 1.4) є застосування в якості пружних елементів вертикального приводу чотирьох стрижнів 1 круглого або прямокутного перерізу. Довжину пружних елементів можна змінювати перестановкою опор 2; в горизонтальному приводі застосовані кручені пружні елементи 3 з перемінним числом витків. Бункер ВЗП має нерухоме дно 4, замкнутий на підставу 5. Для порушення коливань у вертикальному напрямку застосована

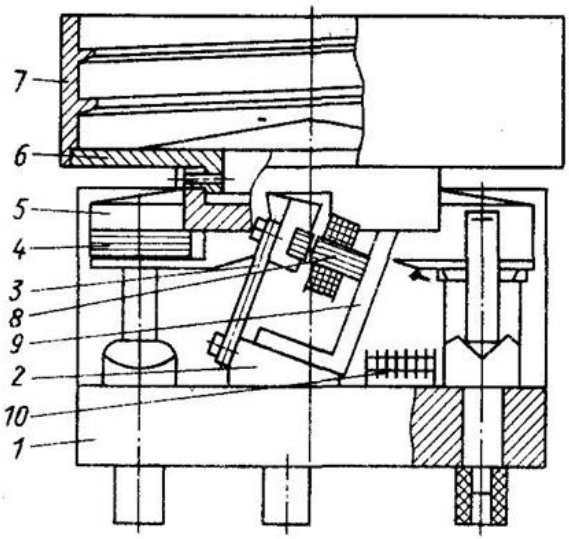


Рис. 1.2. ВЗБП з трьома вібраторами і плоскими пружинами:

основа 1; електромагнітні приводи 2, 4, 5, 8-10; бункера 6, 7 і пружні елементи 3

Конструктивно чаша складається з обичайки, однієї або декількох, спіральної вібродорожкі і конічного дна. Обичайка і дно утворюють бункер. На дні чаші встановлюються пристрої підготовки до захоплення. Вібродорожка служить основним носієм пристроїв і механізмів захоплення орієнтування та видачі.

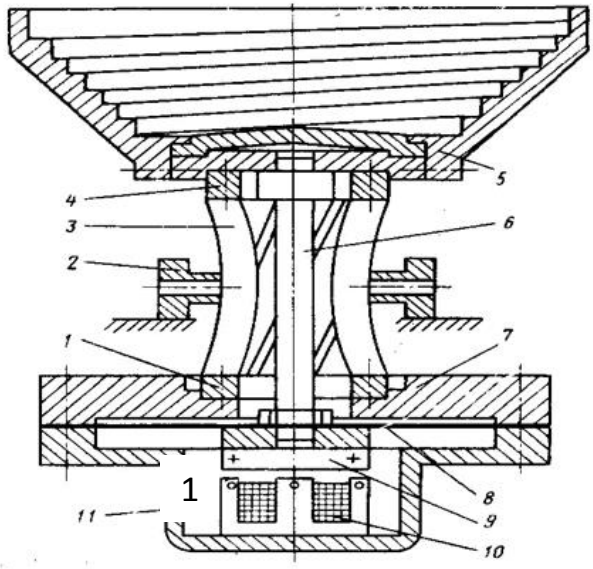


Рис. 1.3. ВБЗП з мембранно-гіперболоїдною пружною системою:

1,4 – фланці; 2 – цапфи; 3 – пластини; 5 – бункер; 6 – торсіон; 7 – основа; 8 – мембрана; 9,10 – електромагнітний привод; 11 – кожух.

Конструктивно чаші поділяють за кількома ознаками**:**

1. За вихідній формі обичайки бункера - конічні, циліндричні, плоскі кругові і комбіновані (рис. 1.5). Конічні мають (рис. 1.5, а) виконання з ламаною, прямолінійною, криволінійної і ступінчастою утворюють. Циліндричні (рис. 1.5, б) використовують як найбільш технологічні у виготовленні для автозавантаження будь-яких О. Плоскі кругові (мал. 1.5, в) застосовують для накопичення ПО , для орієнтування найпростіших деталей, наприклад дисків, переорієнтування, поділу або підсумовування потоків деталей. комбіновані (рис. 1.5, г) виконують, як правило, багатоцільові дії. У них можна виробляти багатономенклатурним завантаження; використовувати одну порожнину чаші як бункер, іншу - як місце установки пристрою, що орієнтує, третю - як накопичувач або суматор і т. д.

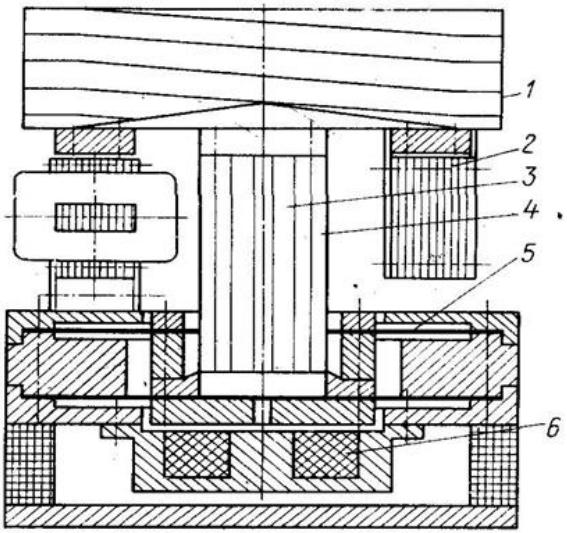


Рис. 1.4. ВБЗП з мембранно-торсіонною пружною системою:

1 – стрижні; 2 – опори; 3 – пружні елементи; 4 – дно; 5 – підстава; 6 – основа.

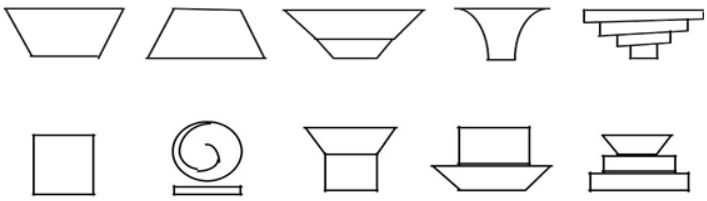


Рис. 1.5. Види чаш ВБЗП

2. За взаємозв'язку обичайки, вібродоріжки, дна - чаші з жорсткою зв'язком елементів, з нерухомою обичайкою, з нерухомим дном, нерухомим дном і обичайкою, з нерухомою доріжкою, з нерухомою доріжкою і обичайкою.

3. За кількістю рухомих потоків ПО - однопоточні і багатопотокові.

4. По розташуванню спіралі вібродорожкі щодо обичайки: усередині, зовні, в дні і їх всілякі поєднання, причому з правого і лівого різьбленням.

5. За характером поля вібрації - з симетрією поля уздовж вібродоріжки з рівномірно змінюваною асиметрією поля в горизонтальній площині; з рівномірно змінюваною асиметрією поля у вертикальній площині; з рівномірно змінюваною асиметрією поля в горизонтальній і вертикальній площині; з нерівномірною асиметрією в горизонтальній площині; з нерівномірною асиметрією у вертикальній площині.

6. На взаємним розташуванням на віброприводі одночашкові, багатоярусні, коаксіальні, аксіальні, комбіновані. Чаші між собою можуть бути з'єднані: загальним потоком, такими, що не є з’єднаними; з'єднані частково.

Розрахунок конструктивних розмірів чаші включає визначення діаметра *D*, висоти *H*, кроку лотка *t*, об'єму *Vд* завантажувальної партії.

Для циліндричної чаші*:*

$$D = D_{\text{в}} + 2\sigma_{\text{с}} = 3\sqrt{\frac{V_{\Delta} Q_z T}{\pi H_p}} + 2\sigma_{\text{с}}$$

де *Dв> (5...8) li* - внутрішній діаметр чаші, мм;

σс-товщина стінки бункера, мм;

Vд – об’єм завантаження виробами, мм3;

T - період часу між заповненнями чаші, с;

*H*р - висота заповнення чаші виробами, мм:

Hр = 2,5·h;

t - крок підйому спірального лотка, мм;

t = Dcp tg (кут 0,5 ... 30, якщо лоток без негативного кута нахилу, то його перевіряють на умову одношарового руху виробів:

$$\text{при li } t \le kd + \delta_{\Delta}, k = \frac{\sqrt{\frac{l_N^2}{d^2}+1}}{\frac{\sqrt{f^2+1}}{d}} = 1 \dots 1{,}5$$

де *Dcp* - середній діаметр руху виробів по лотку, мм;

d - діаметр (висота) вироби, лежачого на лотку, мм;

л - товщина лотка, мм.

Для конічної чаші:

$$D=D_{\text{в}}\left(\frac{H}{H_{\text{р}}}\left(\sqrt{\frac{24V_{\Delta}Q_{z}T}{\pi H_{p}D_{\text{в}}}}-0.75-1.5\right)+1\right)+2\sigma_{\text{с}},$$

де *Dв = (5 ... 8) li*;

H = Hp + (1 ... 1,5) t - повна висота чаші.

Отримане значення діаметра чаші *D* округлюють до найближчого більшого стандартного значення 60, 100, 120, 160, 200, 250, 320, ... 1000 мм;

$$H_{\text{р}} = 2{,}5*h = 2{,}5*(1{,}5d + t_{\text{л}});\ H= H_{\text{р}} + h.$$

1.3. Обраний для досліджень ВБЗП

Для досліджень обрано ВБЗП, що показаний на рис. 1.6.

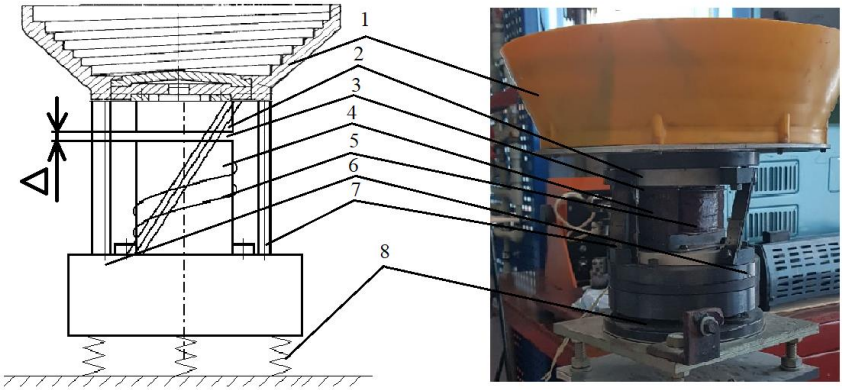


Рис. 1.6. Складові ВБЗП:

1 – чаша; 2 – якір; 3 – зазор; 4 – осердя; 5 – обмотка ЕМ; 6 – масивна основа; 7 – пружини вібраційної системи; 8 – пружні опори

Склад ВБЗП:

1. чаша - матеріал повинен забезпечувати мінімальний коефіцієнт тертя з об'єктом орієнтації. Повинен бути парамагнетик. Чаша має конічне дно з вершиною конуса в центрі;
2. зазор між якорем та осердям електромагніту;
3. якір ЕМ, кріпиться до дна чаши;
4. осердя ЕМ виготовляється з електротехнічної сталі;
5. обмотка ЕМ;
6. масивна основа;
7. віброопори демфуючі елементи, призначені для гасіння коливань і запобігають передачі вібрації на нерухому основу;
8. пружинні, елементи, що визначають жорсткість пружинної системи. Пружні елементи конструктивно виконуються: з плоских пластин, з їх набором з циліндричним стрижнем. Матеріал сталь пружинна.

З рис.1.6 можна зробити висновок що:

- (1 + 2+ О) – активна маса (маса, що вібрує в процесі роботи ВБЗП);
- 4+5+6+7+ жорсткість пружин – реактивна маса (відносно нерухома частина).

При подачі живлення на обмотку ЕМ 5, зміна коливань маси визначається приведеними вище формулами. При цьому активна маса виконує складні коливання при збільшені сигналу живлення, що призводить до зменшення зазору Δ 2, чаша виконує складні вертикально-крутильні переміщення в сторону зменшення жорсткості пружини. При зменшенні подачі сигналу чаша виконує зворотні коливання збільшуючи зазор Δ 2 та за рахунок збільшення жорсткості пружин. При цьому відбувається перерозподіл об'єктів у чаші по перефирії чаші, де О попадають у відповідні гвинтові канавки і переміщуються по них.

1.3. Лабораторна установка і схема управління установкою

Лабораторна установка з ВБЗП складається з чаші 1, на внутрішній циліндричній поверхні, на котрій виконано гвинтовий

лоток. Дно чащі живлення закріплено на 3х похилих плоских пружинах (6), закріплених зажимами в верхнім та нижнім башмаках (5). Пружини розташовані таким чином, що проекція їх на горизонтальну плоскість перпендикулярна до радіусу в точках кріплення їх до дна чащі. Привод БЗУ здійснюється від вертикального вібратора (7), установленого в центрі плити (4). Якір (8) вібратора складається з 2х пакетів електротехнічної сталі, які за допомогою планок кріпляться до основи якоря. Для ізоляції дна бункера від проникнення магнітних силових ліній, які можуть намагнічувати заготовки між основою якоря і дном існує алюмінієва прокладка. Через котушку магніта пропускається змінний струм. Вертикальні коливання якоря вібратора виникають за рахунок вигину похилих пружині і перетворюються в коливання чаші живильника по спіралі. Такі коливальні рухи чаші живильника змушує заготовки, які лежать на поверхності конуса (9), сповзають о гвинтового лотку і піднімаються верх по ньому.

Для віброізоляції БЗУ встановлен на 3-ох гвинтових циліндричних порівняно невеликої жорсткості пружинах. Усунення рухливості живильника на пружинах досягається установкою на основі (2) і осі (1) в розрізної втулкою, якак входить в отвір плити (4) с невеликим зазором. Ця ось, забезпечуючи амортизованою системою дві степені свободи-переміщенняпо вертикалі і обертання навколо вертикальної осі, яка ообмежує можливість переміщення в других направленнях.

Від БЗУ заготовки по лотку (11) потрапляє до тари. Усі елементи управління лабораторної установки поміщені до блока управління. На рис. 1.7. представлена конструкція ВБЗП.

На рис 1.2 представлена схема живлення ВБЗП, на якому діаметр чаші 250мм ($D_{чаш} \geq 250$мм).

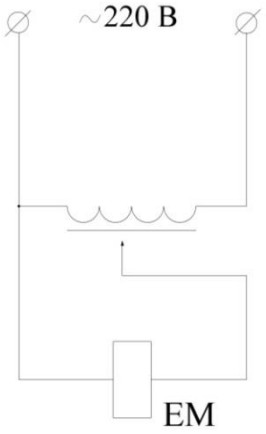


Рис. 1.7. Схема ВБЗП

При частоті струму $f_ж$ = 50Гц, частота коливань чаші з О ВБЗП $f_{ВБЗП}$ = 100 Гц. Це пояснюється реакцією активної маси (чаша ВБЗП, якір електромагніта ЕМ та деталей в чаші) на позитивну і негативну напівсинусоїди на півхвиль частоти живлення ЕМ за рис. 1.7.

Графік амплітуди коливань ВБЗП представлено на рис.1.8.

З графіку рис. 1.8 видно, що електромагніт вмикається два рази за період при додатньому та від'ємному значеннях напівсинусоїди, що і спричиняє зростання частоти у 2 рази. За такою схемою збудження, як правило, працюють ВБЗП з $D_{чаш} \leq 250$мм.

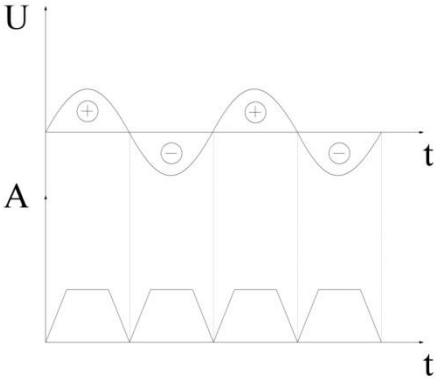


Рис. 1.8. Графік амплітуди

На рис. 1.9. представлена схема ВБЗП із діодом VD1, з яким встановлюють чаши зі діаметром понад 250 мм ($D_{чаш} > 250$мм).

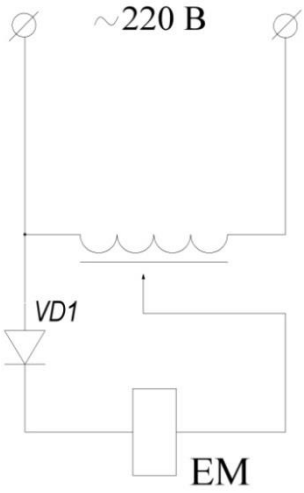


Рис. 1.9. Схема досліджуваного ВБЗП із діодом VD1

Оскільки у колі наявній діод, то частота живлення $f_ж$ = 50Гц відповідає частоті ВБЗП $f_{ВБЗП}$ = 50 Гц, тобто $f_ж = f_{ВБЗП}$ = 50 Гц.

Графік роботи схеми рис. 1.9. представлений на рис. 1.10.

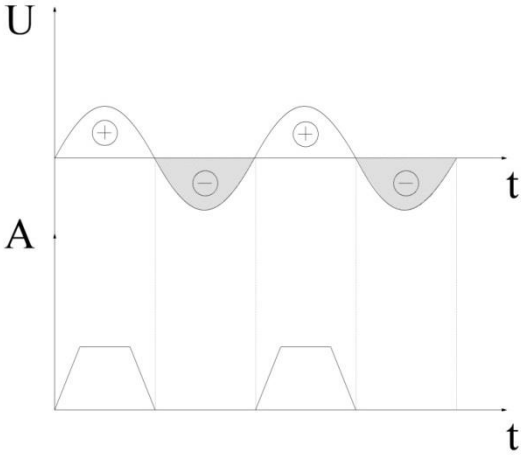


Рис. 1.10. Графік напруги і амплітуди роботи схеми із діодом у колі

Оскільки основною особливістю діода є здатність не пропускати зворотній струм, а саме струм з однією із напівхвиль синусоїди живлення лише при дії іншої на півхвилі, що видно на графіку на рис. 1.10.

1.4. Опис схеми електрично-принципової системи управління БЗУ

Електрична схема управління вібробункером (рис. 1.12.) призначена для живлення котушки електромагніта пульсуючим струмом з частотою 50 Гц.

Схема забезпечує плавне регулювання амплітуди коливань чаші від нуля до максимума. Керування включає у собі окремо взяті схеми:

I. Схема підключається до ЕМ пакетним перемикачем, увімкнутим в перше положення «I» і вмикає в схему змінний опір *R1*. За допомогою цього підключення відбувається регулювання коливань чаші вібробункера шляхом зниження або підвищення напруги живлення.

II. Схема вмикається таким же перемикачем, ввімкнуте у друге «2» положення и складається з змінного опору *R1*, та випрямного діода *VD1*. За допомогою цього діода на ЕМ подаються позитивні на півхвилі мережевої напруги.

III. Схема вмикається аналогічно першим двом, третє положення «3» пакетного перемикача і складається з тиристорного перемикача регулятора. Позитивні на півхвилі мережевої напруги проходять по випрямного ланцюжка *R3, R2, VD4*. Через захисний діод *VD3* вони надходять на керуючий електрод тиристора *VD2*, який відкривається і ЕМ починає працювати. При переміщенні движка змінного резистора *R2* вверх по схемі *VD2* починає відкриватися в кожен напівперіод, якщо раніше – коливання будуть збільшено амплітудою.

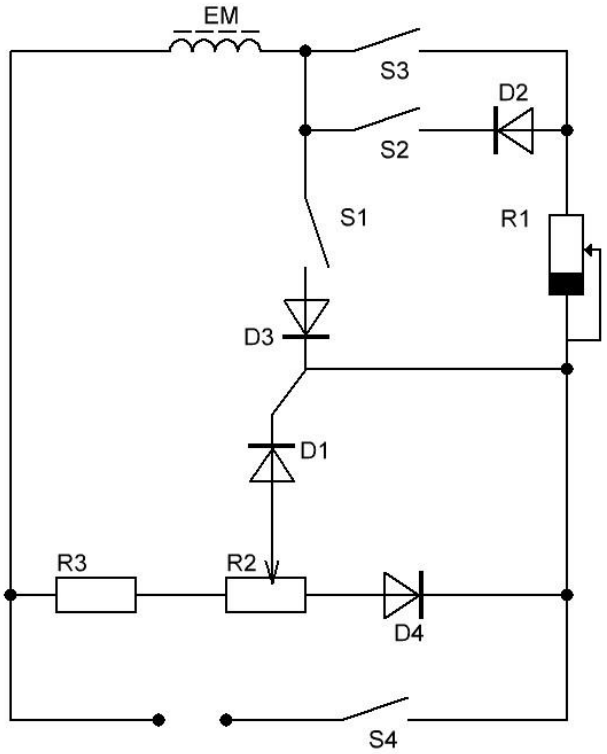


Рис. 1.11. Схема електрична управління вібробункером

Таким чином при подачі живлення на обмотку ЕМ зміна коливань маси визначається наведеними вище формулами. При цьому активна маса виконує складні коливання при збільшені сигналу живлення, що призводить до зменшення зазору Δ, чаша виконує складні вертикально-крутильні переміщення відповідним амплітудами $A_в$ та $A_г$ сторону зменшення жорсткості пружини.

При зменшенні подачі сигналу чаша виконує зворотні коливання, збільшуючи зазор Δ та за рахунок збільшення жорсткості пружин. При цьому відбувається перерозподіл об'єктів у чаші по перефірії чаші, де об'єкти попадають у відповідні гвинтові канавки і переміщуються по них.

Видами регулювання продуктивності ВБЗП і наступні:

1. зміна первинна величини зазору 2;
2. зміна жорсткості пружин 8;
3. зміна величини намагнічуваної сили F = I*W, де I величина струму, що подається на обмотки ЕМ, А; W – кількість витків в обмотці ЕМ4
4. зміна форми сигналу живлення впродовж одного циклу, за рахунок внесення додаткових елементів: діодів, тиристорів тощо;
5. зміна частоти сигналу живлення.

На рис. 1.12. представлена узагальнена структурна схема мехатронної системи (МТС) на базі ВБЗП. Вона в ідеалі відтворює вхідні та досліджувальні, тобто вихідні параметри МТС, а також певну множину досліджувальних параметрів при дослідженні продуктивності ВБЗП.

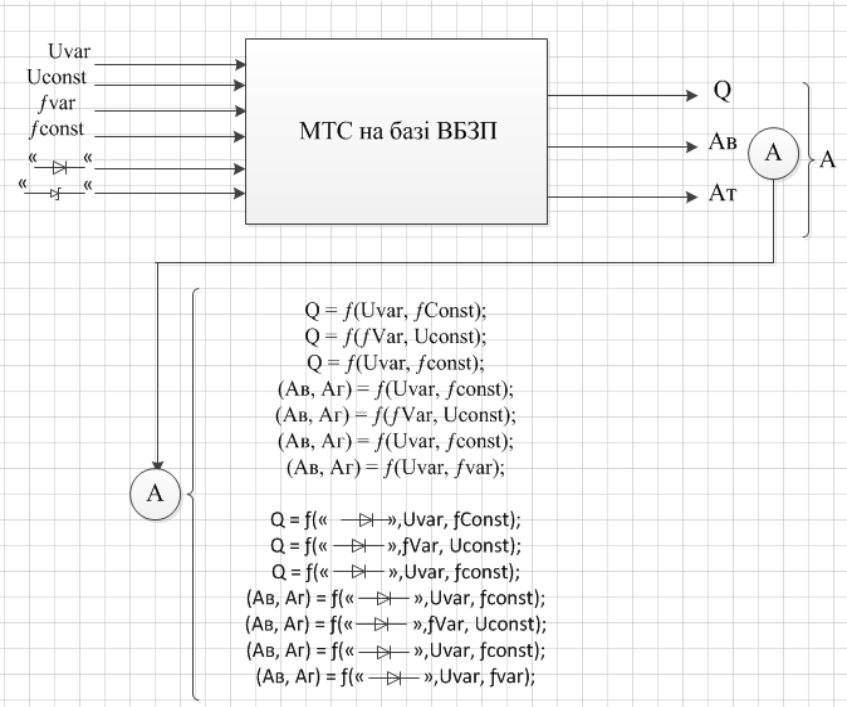


Рис.1.12. узагальнена структурна схема мехатронної системи (МТС) на базі ВБЗП

1.5. Опис досліджуваного об'єкту

Досліджуваний об'єкт є тілом конусоподібної форми, що має більший діаметр Ø12 мм, менший Ø5 мм, а висотою 15 мм. Має деякі пустоти всередині. Із особливостей об'єкту є його ребристість по його зовнішній твірній.

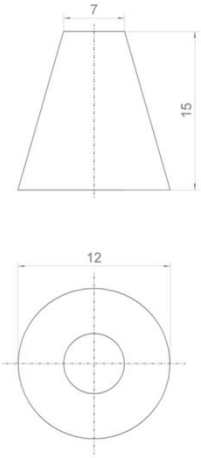


Рис. 1.13. Досліджуваний об'єкт

1.6. Порядок виконання роботи

1. Увімкнути в мережу живлення обмотки ЕМ.
2. Змінюючи величину напруги живлення обмотки ЕМ U в межах (100..250 В) з кроком 10 В визначити продуктивність роботи ВБЗП для кожного із значень напруги за вказаний викладачем період часу. Дані занести до табл. 1.1.

Таблиця 1.1

№ з/п	Кількість вимірів	Напруга, В	Продуктивність, шт.
	1		
	n		

3. Побудувати графік залежності продуктивності Q від величини напруги U що подається на обмотку ЕМ, за даними табл. 1.1.

4. Виконати індивідуальне завдання.
5. Скласти звіт по роботі.

1.7. Зміст звіту

1. Назва та мета роботи.
2. Короткі теоретичні відомості щодо призначення та конструкції ВБЗП.
3. Опис проведення дослідів із заповнення табл. 1.1.
4. Графік Q = f(U).
5. Назва та варіант індивідуального завдання
6. Висновки по роботі.

Лабораторна робота №2

ОСНОВИ ПРОГРАМУВАННЯ РОБОТА *Braccio* В СЕРЕДОВИЩІ *Arduino IDE*

Мета роботи – отримати практичні навички роботи у програмному середовищі *Arduino IDE* щодо відпрацювання основних рухів ланками маніпуляційної системи робота *Braccio.*

2.1. Теоретичні відомості

2.1.1. Основі відомості про робот моделі *TinkerKit Braccio*

TinkerKit Braccio – це робот ангулярної системи, маніпуляційна система (МС) якого має 5 ступенів рухомості, що реалізуються 6 рухомими ланками від L2 до L6, які приводяться в рух відповідними двигунами від М1 до М6 (див. загальний вигляд робота на рис. 2.1):

- L1 – Deatl (основа);
- L2 – Base (основа, приводиться в рух двигуном М1 відносно ланки L1);
- L3 – Shoulder (плече, приводиться в рух двигуном М2);
- L4 – Elbow (лікоть, приводиться в рух двигуном М3);
- L5 – Vertical wrist (вертикальний зап’ясток, приводиться в рух двигуном М4);
- L6 – Rotary wrist (обертальний зап’ясток, приводиться в рух двигуном М5);
- L7 – конструктивна константа;
- L2 – Gripper (схват, затискний пристрій, приводиться в рух двигуном М6);

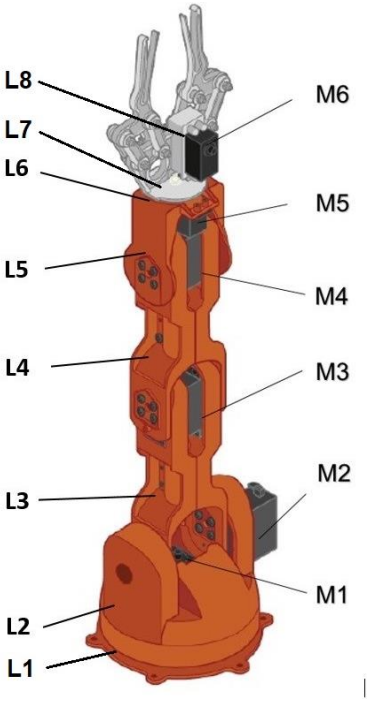


Рис. 2.1. – Загальний вигляд робота TinkerKit Braccio

Конфігурацію робота можна змінювати, зменшуючи кількість рухомих ланок його МС. На робот встановлений класичний двопальцевий схват (gripper), який при бажанні можна замінити на легкий інструмент або інший прилад, наприклад, камеру, ліхтар тощо.

Серводвигуни (рис. 2.2) забезпечують рух всіх ланок МС та роботу схвата. Потужність двигунів різна і залежить від ланки, яку двигун приводить в рух. Наприклад, на ланці горизонтального кутового переміщення станини встановлений двигун М1 потужніший, ніж на схваті (двигун М6).

Дана модель робота ангулярної системи, так як ланка L2 забезпечує обертання інших ланок МС навколо вертикальної осі, а інші навколо горизонтальних осей щодо положення, зображеного на рис. 1.2. Ланки L1, L3, L4 і L5 мають кут

повороту ±90°, а ланка L2 ±75°. Серводвигун схвата відпрацьовує кут в інтервалі від 0° до 70°.

Кінематична структура робота *TinkerKit Braccio* та його розміри показані на рис. 2.2 та 2.3.

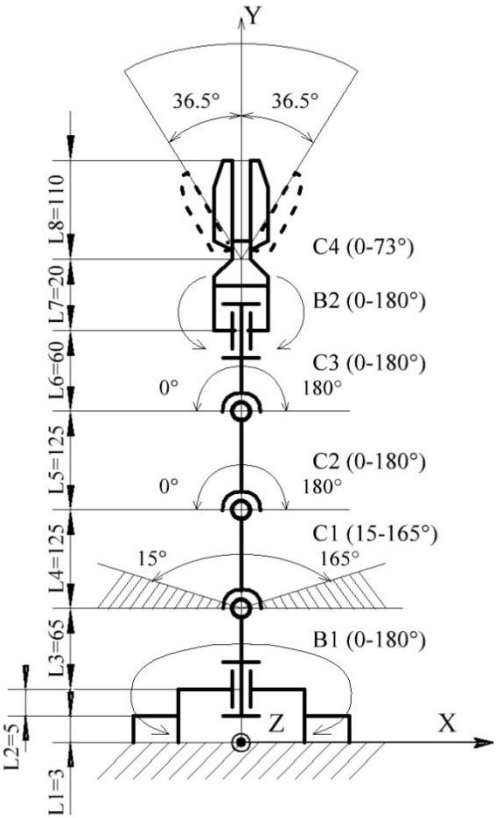


Рис. 2.2. – Кінематична структура робота TinkerKit Braccio

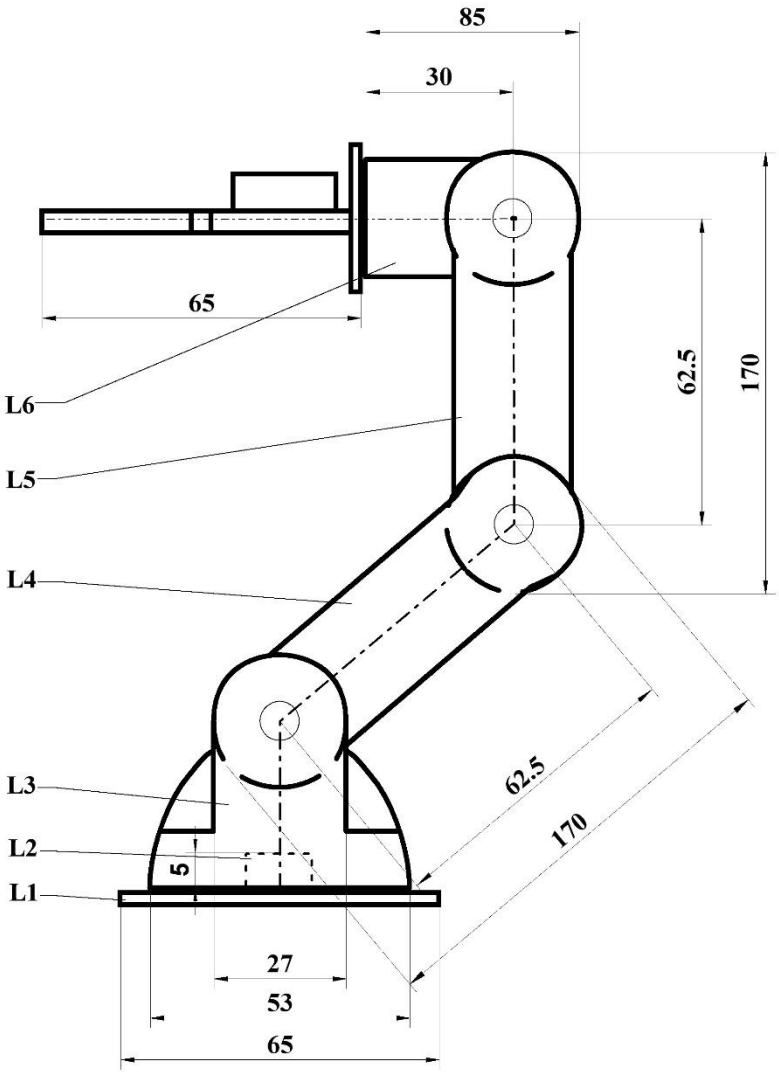


Рис. 2.3. – Основні розміри робота TinkerKit Braccio

Серводвигуни М1 – М6 розміщені на корпусах відповідних ланок та закріплені за допомогою гвинтів у спеціальних пазах.

На рис. 2.4 показано розміщення серводвигунів М1 та М2.

Рис. 2.4. – Розташування серводвигунів М1 та М2

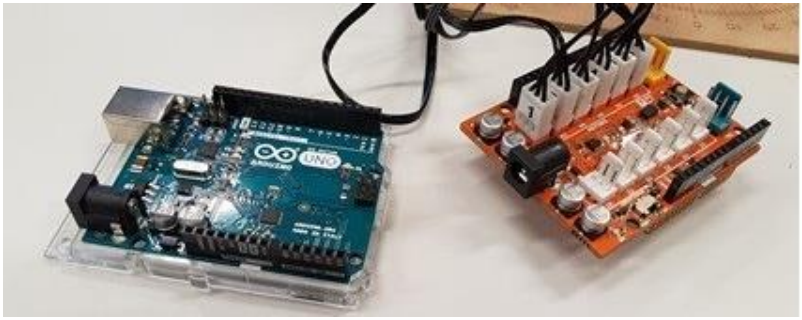
Серводвигун (від лат. *Servus* – слуга, помічник, раб) – механічний привід з автоматичною корекцією стану кутового положення ротора (вала) через внутрішній негативний зворотний зв'язок відповідно до параметрів, заданих із зовні) складається із редукторної системи шестерень зі стопорами ходу (для відпрацювання кута від 0° до 180°), внутрішньої плати керування, двигуна та корпусу.

На рис. 2.5 показано внутрішню будову серводвигунів, які використовуються для переміщення ланок робота *TinkerKit Braccio*.

Робот функціонує на базі мікроконтролера *Arduino Uno* та модуля *Braccio Shield* (рис. 2.6). Для функціонування робота необхідно встановити спеціальні бібліотеки на ПЗ *Arduino IDE*. Програма стандартно компілюється на комп’ютері, а потім

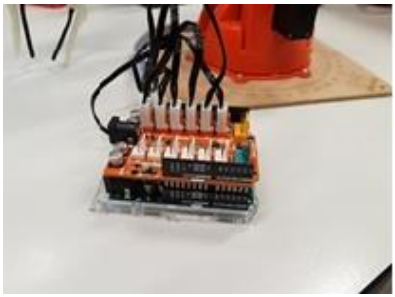
записується у мікроконтролер *Arduino Uno*. Модуль *Braccio Shield* може керувати 6 серводвигунами (М1 – М6) одночасно.

Рис. 2.5. – Внутрішня будова серводвигуна



а)

б)



в)

Рис. 2.6. – Загальний вигляд мікроконтролера Arduino Uno та модуля Braccio Shield (а), під'єднання модуля Braccio Shield до Arduino Uno (б), з'єднані модуль Braccio Shield та Arduino Uno (в)

Також є можливість підключити до модуля 6 датчиків (в роз'єми *INPUTS* за рис. 2.7). Дана функція розширює можливості застосування робота.

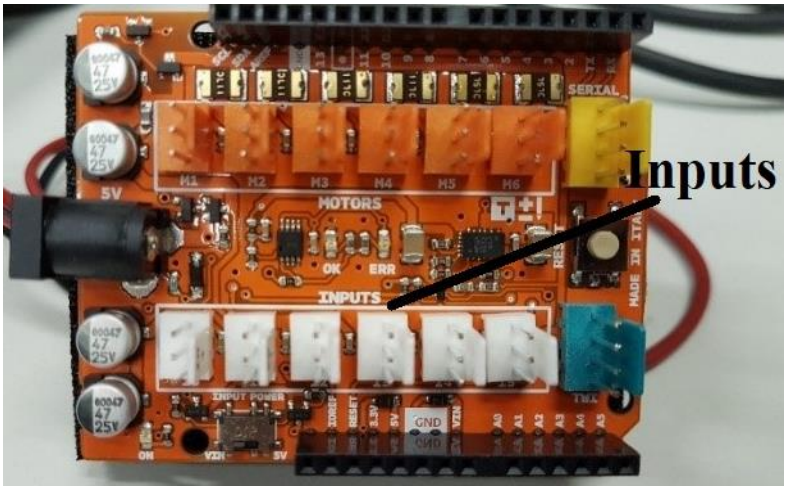


Рис. 2.7. – Загальний вигляд плати розширення Braccio Shield

Напруга на модуль керування двигунами подається з окремого блоку живлення, який забезпечує 5 В постійного струму силою до 5000 mA. Теоретично система може працювати від потужного акумулятора.

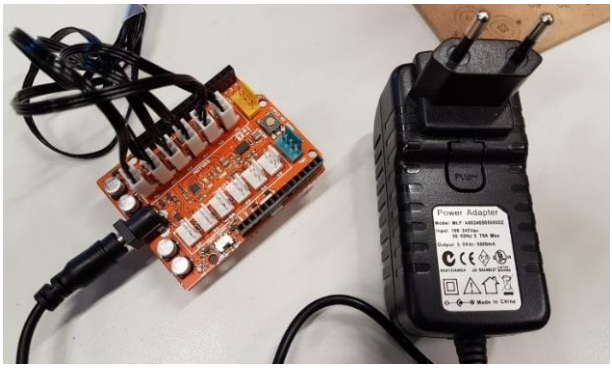


Рис. 2.8. – Модуль Braccio Shield та блок живлення модуля

2.2. Технічні характеристики робота *TinkerKit Braccio*

2.2.1. Деталі та склад

Ланки робота можна зібрати в різні конфігурації (рис. 2.9), що реалізується завдяки різним деталям та взаємозамінним модулям. В комплект поставки включені нижче приведені деталі, плати та двигуни, а саме:

• 21 пластикових деталей;

• 2 серводвигуни *SR 311*;

• 4 серводвигуни *SR 431*;

• 1 *Braccio shield* для *Arduino Uno*;

• 1 блок живлення 5В, 5А;

• 1 протектор кабелів.

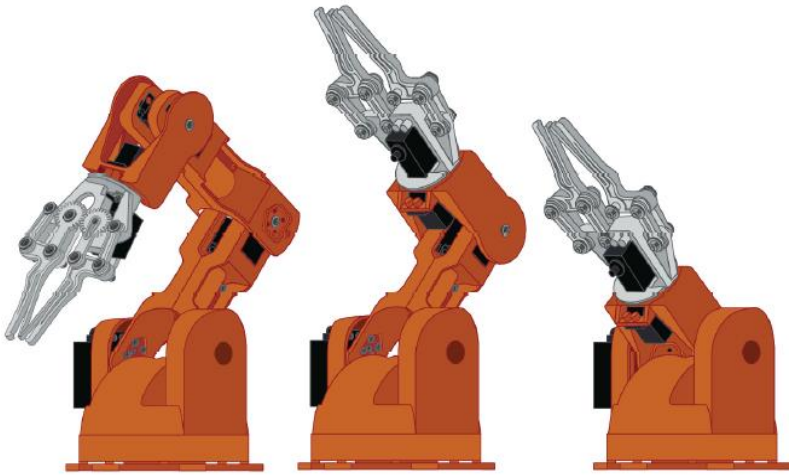


Рис. 2.9. – Можливі конфігурації побудови робота TinkerKit Braccio

Характеристики повністю зібраного робота *TinkerKit Braccio* приведені в табл. 2.1.

Таблиця 2.1. – Характеристики робота TinkerKit Braccio (у конфігурації з 6 ланками)

№	Найменування	Параметр	Величина
1	Вага	грам	792
2	Робоча відстань	см	80
3	Висота	см	52
4	Ширина основи	см	14
5	Ширина захвату	мм	90
6	Довжина кабелю	см	40
7	Вантажопідйомність при робочій відстані 32 см	грам	150
8	Максимальна маса	грам	400

2.3. Контролер *Arduino Uno*

2.3.1. Загальні відомості про контролер *Arduino Uno*

Arduino Uno – це пристрій на основі мікроконтролера *ATmega328*. У його склад входить: 14 цифрових входів / виходів (з них 6 можуть використовуватися в якості ШІМ-виходів; ШІМ – широтно-імпульсна модуляція), 6 аналогових входів, кварцевий резонатор на 16 МГц, роз'єм *USB*, роз'єм живлення, роз'єм *ICSP (*від англ. *In Circuit Serial Programming* - внутрішньосхемне програмування*)* і кнопка скидання. Для початку роботи з пристроєм досить просто подати живлення від *AC/DC*-адаптера або батарейки, або підключити його до комп'ютера за допомогою *USB*-кабелю.

Загальний вигляд контролера *Arduino Uno* представлено на рис. 2.10.

Рис. 2.10. – Контролер Arduino Uno

Характеристики контролера *Arduino Uno* приведені у табл. 2.2.

Таблиця 2.2. – Характеристики контролера Arduino Uno

№	Найменування	Параметр	Величина
1	Мікроконтролер	–	ATmega 328
2	Робоча напруга	В	5
3	Напруга живлення	В	7–12
4	Максимальний струм одного вивода	мА	40
5	Flash-пам’ять	КБ	32
6	SRAM	КБ	2

2.3.2. Живлення контролера *Arduino Uno*

Напруга живлення для *Arduino Uno* подається від *USB* або від зовнішнього джерела живлення.

В якості зовнішнього джерела живлення (не *USB*) може використовуватися мережевий *AC / DC* - адаптер або акумулятор / батарея. Штекер адаптера необхідно вставити у відповідний роз'єм живлення на платі. У разі живлення від акумулятора / батареї їхні контакти необхідно під'єднати до виводів *Gnd* та *Vin* роз'єму *POWER*.

Нижче перераховані виводи живлення, розташовані на платі:

VIN – напруга, що надходить в *Arduino Uno* безпосередньо від зовнішнього джерела живлення (не пов'язане з 5В від *USB* або іншим стабілізованою напругою). Через цей вивід можна як подавати зовнішнє живлення, так і споживати струм, коли пристрій живиться від зовнішнього адаптера;

5V – на вивід надходить напруга 5В від стабілізатора напруги на платі незалежно від того, як живиться пристрій: від адаптера (7─12В), від *USB* (5В) або через вивід *VIN* (7–12В). Живити пристрій через виводи *5V* або *3V3* не рекомендується, оскільки в цьому випадку не використовується стабілізатор напруги, що може призвести до виходу плати з ладу;

3.3V – 3.3В, напруга, що надходять від стабілізатора напруги на платі. Максимальний струм, споживаний від цього виводу, становить 50 мА;

GND – вивід землі або мінус у замкнутому колі;

IOREF – цей вивід надає платам розширення інформацію про робочу напругу мікроконтролера *Arduino Uno*. Залежно від напруги, зчитаного з виводу *IOREF*, плата розширення може переключитися на відповідне джерело живлення або задіяти перетворювачі рівнів, що дозволить їй працювати як з 5В, так і з 3.3В пристроями.

2.3.3. Входи і виходи контролера *Arduino Uno*

Нижче представлені позначення виходів мікроконтролера *Arduino Uno,* що відповідають рис. 2.10.

З використанням функцій *pinMode (), digitalWrite ()* і *digitalRead ()* кожен з 14 цифрових виводів може працювати в якості входу або виходу. Рівень напруги на виводах обмежений 5В. Максимальний струм, який може віддавати або споживати один вивід, становить 40 мА. Нище представлений опис роз’ємів.

Послідовний інтерфейс: роз'єми *0* (*RX*) і ***1* (*TX*).** Використовуються для отримання (*RX*) і передачі (*TX*) даних по послідовному інтерфейсу. Ці виводи з'єднані з відповідними виводами мікросхеми *ATmega8U2*, яка виконує роль перетворювача *USB-UART* (*UART* – від англ. *Universal Asynchronous Receiver/Transmitter* – універсальний асинхронний приймач/передавач).

Зовнішні переривання: роз'єми *2* та *3*. Можуть слугувати джерелами переривань, що виникають при фронті, спаді або при низькому рівні сигналу на цих виводах. Для отримання додаткової інформації див. функцію *attachInterrupt ()*.

ШІМ: роз'єми *3, 5, 6, 9, 10* та *11*. За допомогою функції *analogWrite ()* можуть виводити 8-бітові аналогові значення в вигляді ШІМ-сигналу.

Інтерфейс *SPI*: роз'єми *10* (*SS*), *11* (*MOSI*), *12* (*MISO*), *13* (*SCK*). Із застосуванням бібліотеки *SPI* дані виводи можуть здійснювати зв'язок по інтерфейсу *SPI*.

Світлодіод: роз'єм *13*. Вбудований світлодіод, приєднаний до виводу *13*. При відправці значення *HIGH* світлодіод вмикається, при відправці *LOW* – вимикається.

В *Arduino Uno* є 6 аналогових входів (*A0* – *A5*), кожен з яких може представити аналогову напругу у вигляді 10-бітного числа (1024 різних значень). За замовчуванням вимірювання напруги виконується в інтервалі від 0 до 5 В.

Крім перерахованих на платі існує ще кілька виводів, а саме:

***AREF* –** опорна напруга для аналогових входів. Може бути задіяний функцією *analogReference ();*

***Reset* –** формування низького рівня (*LOW*) на цьому виводі призведе до перезавантаження мікроконтролера. Зазвичай цей вивід слугує для функціонування кнопки скидання на платах розширення.

2.3.4. Програмне забезпечення *Arduino IDE*

IDE (від англ. *Integrated Development Environment* – інтегроване середовище розробки) – це додаток або група додатків (середовище), призначених для створення, налаштування, тестування та обслуговування програмного забезпечення.

Інтегроване середовище розробки характеризується наявністю складної функціональності, включаючи редагування і компіляцію вихідного коду, створення програмних ресурсів, створення баз даних і т.д.

Для програмування контролерів серії *Arduino* фірмою-розробником створено програмне забезпечення, що відповідає основним вимогам типового середовища *IDE*. Це не потужне програмне забезпечення, а проста функціональна програма, яка дозволяє писати, компілювати і завантажувати програму в мікроконтролер.

Середовище розробки *Arduino IDE* являє собою текстовий редактор програмного коду, область повідомлень, вікно виведення тексту (консоль), панель інструментів і кілька меню. Для завантаження програм і зв'язку середовище розробки підключається до апаратної частини контролерів *Arduino*.

Програми для контролерів *Arduino* пишуться на звичайній мові програмування *C++*, доповненою простими і зрозумілими функціями для керування введенням / виведенням на контактах.

Arduino IDE містить множину попередньо створених бібліотек. Бібліотеки додають додаткову функціональність скетчам (від англ. *sketch* – ескіз, скетч, чорновик), наприклад, при роботі з апаратною частиною або при обробці даних. Одна або кілька директив "*#include"* будуть розміщені на початку коду скетчу з подальшою компіляцією бібліотек і разом зі скетчем. Завантаження бібліотек вимагає додаткового місця в пам'яті контролерів *Arduino*.

Для роботи з платами *Arduino* та модуля *Braccio Shield* необхідна спеціальна програма *Arduino IDE* (рис. 2.11). *Arduino*

IDE – це середовище розробки програм для платформ на базі мікропроцесорів *Arduino* (інше визначення див. вище). За допомогою цього компілятора розроблені програми записуються у пам'ять мікропроцесора.

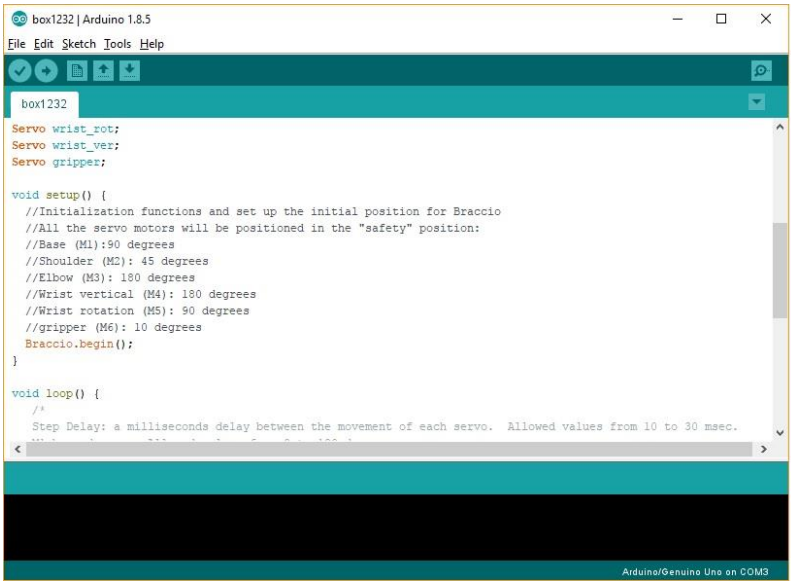


Рис. 2.11. – Вікно середовища розробки програм Arduino IDE

2.4. Попередня підготовка обладнання для роботи з середовищем програмування *Arduino IDE* та роботом *TinkerKit Braccio*

2.4.1. Встановлення *Arduino IDE*

Для початку роботи із лабораторним стендом, який описано в п. 2.6, необхідно встановити на комп'ютер інтегроване середовище програмування контролерів *Arduino* – *Arduino IDE*. Програма доступна для наступних операційних систем (ОС):

- *Windows* (починаючи з версії *XP* до *10*);
- *Linux* (*x32*, *x64* та *ARM*);

- *Mac OS X*.

Для цього необхідно перейти за посиланням: https://www.arduino.cc/en/Main/Software та завантажити інсталятор останньої версії програми.

Скріншот вікна браузера за вказаним посиланням представлений на рис. 2.12.

Рис. 2.12. – Скріншот вікна браузера за посиланням для завантаження інтегрованого середовища програмування Arduino IDE

Після вибору інсталятора для відповідної операційної системи у браузері відобразиться вікно, у якому пропонується завантаження безкоштовного інсталятора або завантаження із благодійним внеском.

Обирається пункт "*just download*", що показано на рис. 2.13 і виділено красною рамкою у нижньому правому куті, для отримання інсталятора без благодійного внеску.

Рис. 2.13. – Скріншот вікна завантаження інсталятора

Після натискання кнопки "*just download*" обирається папка розміщення інсталятора, щоб при завершенні завантаження знайти файл та запустити його для встановлення *Arduino IDE*. На прикладі рис. 2.14 обрано папку "*Рабочий стол*".

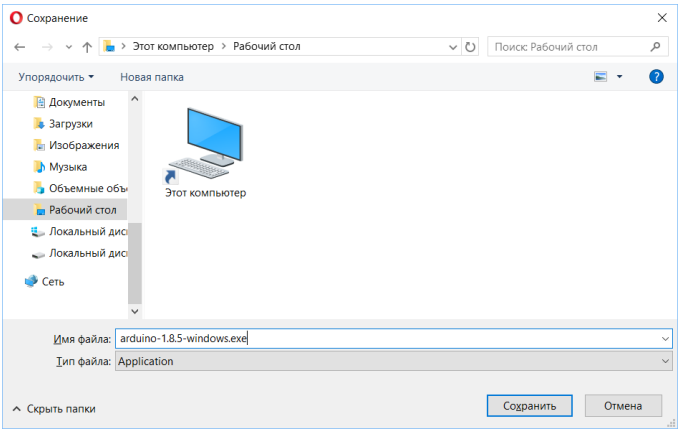


Рис. 2.14. – Скріншот вікна завантаження інсталятора

При запуску інсталятора відобразиться вікно (рис. 2.15), на якому представлені ліцензійні права розробника додатка

Arduino IDE. Встановлення програмного середовища виконується натисканням кнопки "*I Agree*", що виділено червоною рамкою у правому нижньому куті на рис. 2.15.

Наступний крок – вибір потрібних компонентів додатку *Arduino IDE* для встановлення їх на комп'ютер. Для цього виділяються позначками всі компоненти, як показано на рис. 2.16, і натискається кнопка "*Next*", що виділено на рис. 2.16 красною рамкою у правому нижньому куті.

Далі обирається папка, в яку буде встановлено програмне забезпечення. Для прикладу на рис. 2.17 показаний стандартний шлях встановлення програмного забезпечення на локальному диску *C:*. Він виконується натисканням кнопки "*install*", що виділено на рис. 2.17 красною рамкою у правому нижньому куті.

Під час встановлення *Arduino IDE* комп'ютер може запросити доступ на встановлення драйверів для портів. За такої ситуації необхідно надати дозвіл програмі на їх встановлення. На рис. 2.18 показано завершення встановлення додатку, і завершення роботи з інсталятором натисканням кнопки "*Close*".

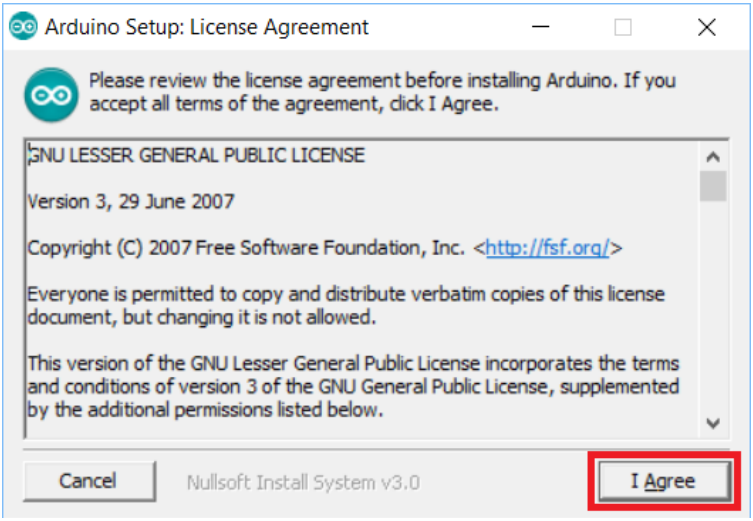


Рис. 2.15. – Скріншот вікна погодження з ліцензійними правами Arduino IDE

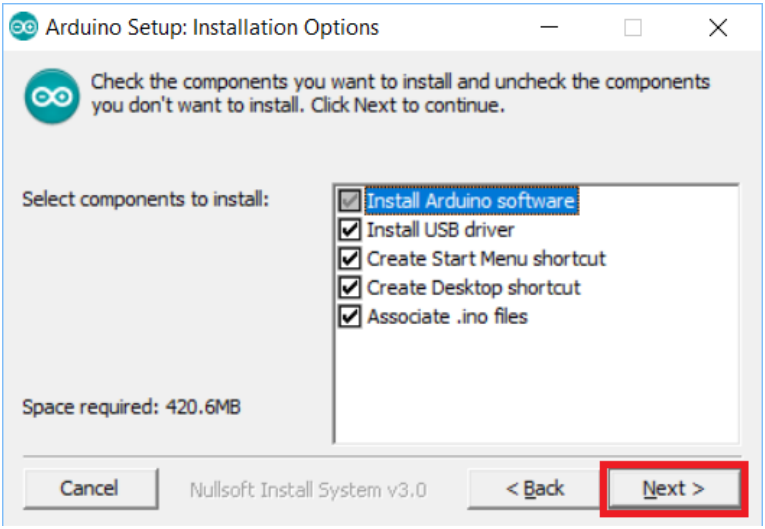


Рис. 2.16. – Скріншот вікна вибору компонентів Arduino IDE для їх встановлення

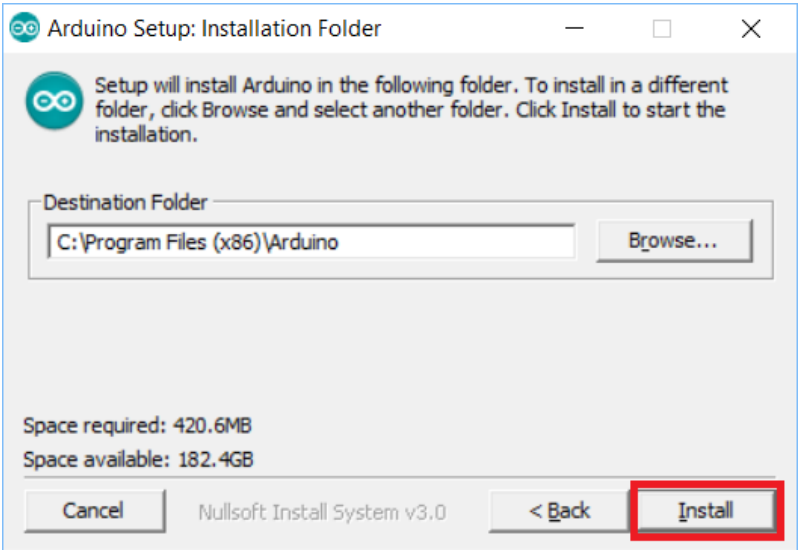


Рис. 2.17. – Скріншот вікна вибору папки для встановлення програми Arduino IDE

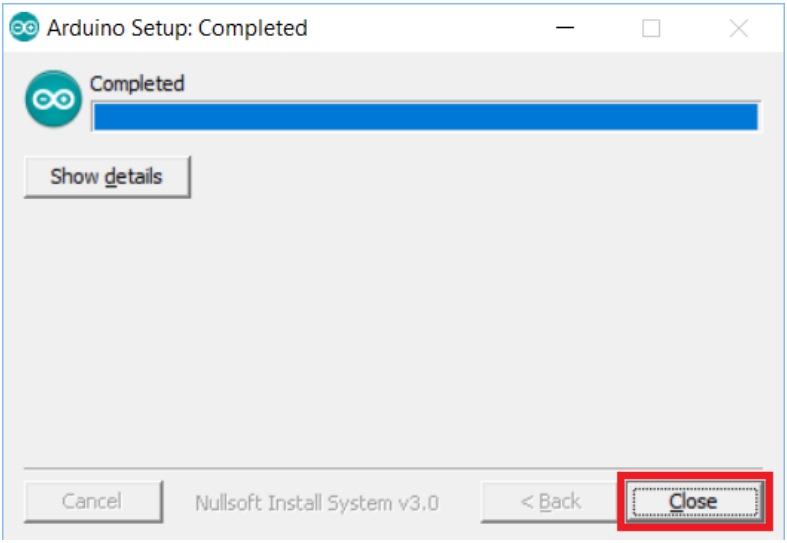


Рис. 2.18. – Скріншот завершення встановлення Arduino IDE

Після виконання попередніх кроків на робочому столі має з'явитись ярлик "*Arduino*", який показано на рис. 2.19.

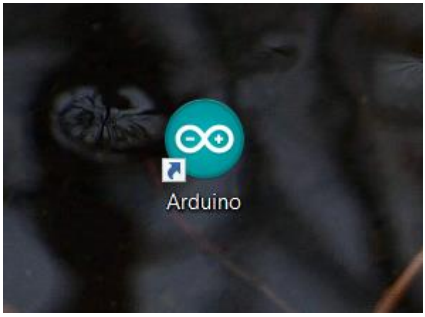


Рис. 2.19. – Ярлик додатку Arduino IDE

2.4.2. Запуск *Arduino IDE*

Після завантаження та встановлення додатку *Arduino IDE* його запуск виконується подвійним кліком миші на ярлику "*Arduino*", після чого з'явиться вікно *Arduino IDE* (рис. 2.19).

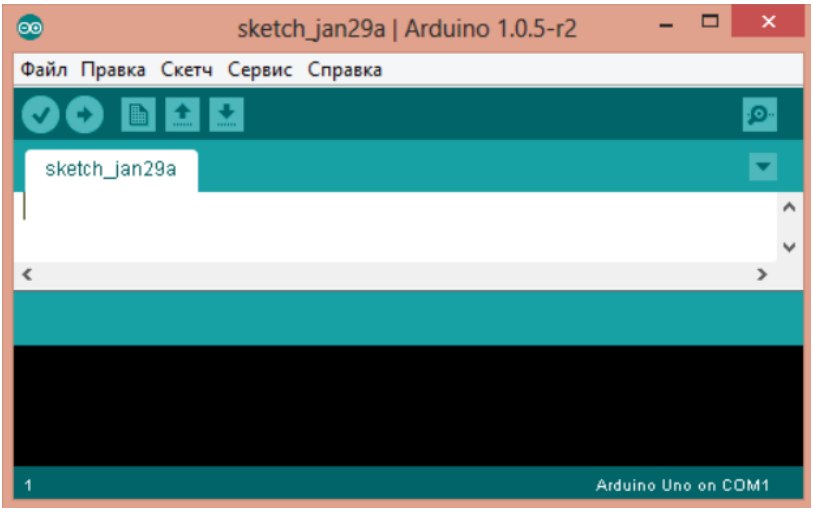


Рис. 2.19. – Вікно додатку Arduino IDE

2.4.3. Підключення *Arduino Uno* до комп'ютера

Після встановлення додатку *Arduino IDE* потрібно підключити *Arduino Uno* до комп'ютера.

Для цього необхідно з'єднати *Arduino Uno* з комп'ютером через *USB*-кабель. На платі *Arduino Uno* загориться світлодіод "*ON*" і почне блимати світлодіод "*L*" (рис. 2.20). Це означає, що на плату подано живлення і мікроконтролер *Arduino Uno* почав виконувати "*прошиту*" за замовчуванням програму "*Blink*" (миготіння світлодіодом).

Рис. 2.20. – Сигнали про живлення Arduino UNO

Для налаштування *Arduino IDE* на роботу з *Arduino Uno* необхідно знати, який номер COM-порту присвоїв комп'ютер *Arduino Uno*. Для цього потрібно зайти в "*Диспетчер пристроїв Windows*" (рис. 2.21) і розкрити вкладку "*Порти (COM и LPT)*".

При цьому операційна система розпізнала плату *Arduino Uno* як *COM*-порт, підібрала для неї правильний драйвер і призначила цього *COM*-порту номер *7*. Якщо підключити до комп'ютера іншу плату *Arduino*, то операційна система призначить їй інший номер. Тому, при підключені декількох плат *Arduino* дуже важливо не заплутатися в номерах *COM*-портів.

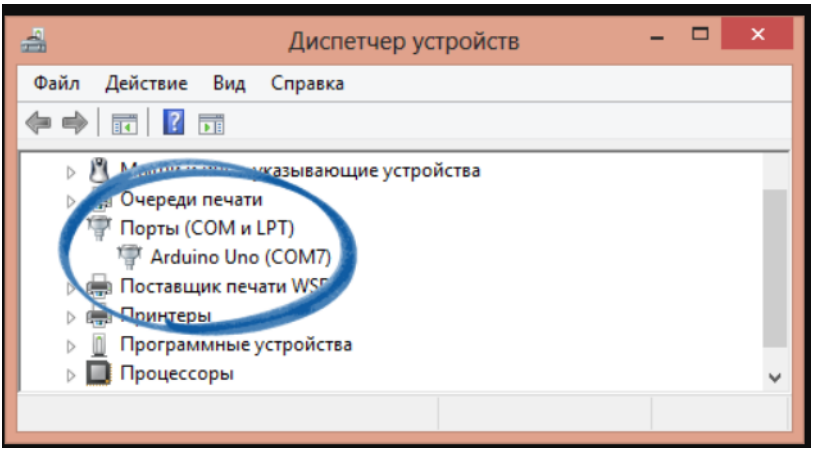


Рис. 2.21. – Диспетчер пристроїв Windows

2.4.4. Налаштування *Arduino IDE* на роботу з *Arduino Uno*

Для вказаного в заголовку необхідно забезпечити знаходження плати *Arduino Uno* в середовищі *Arduino IDE* в його *COM*-порту "*COM7*".

Для цього необхідно перейти в меню "*Сервис*" → *"Последовательный порт"* і обрати порт "*COM7*" (рис. 2.22). Тепер *Arduino IDE* знає, що плата знаходиться на порту "*COM7*".

Для правильної компіляції коду необхідно встановити, що середовище програмування *Arduino IDE* працює з *Arduino Uno*. Для цього перейти в меню "*Сервис*" → "*Плата*" і вибрати плату "*Arduino Uno*" (рис. 2.23).

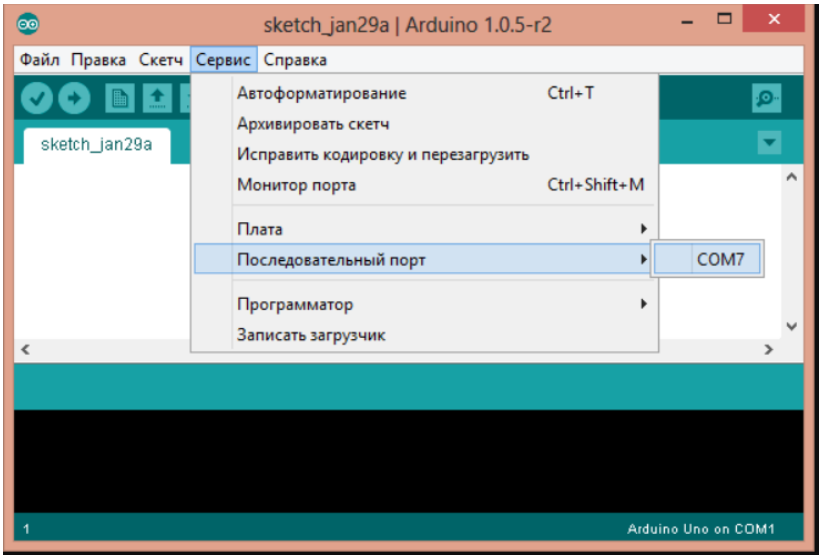


Рис. 2.22. – Вибір COM-порту

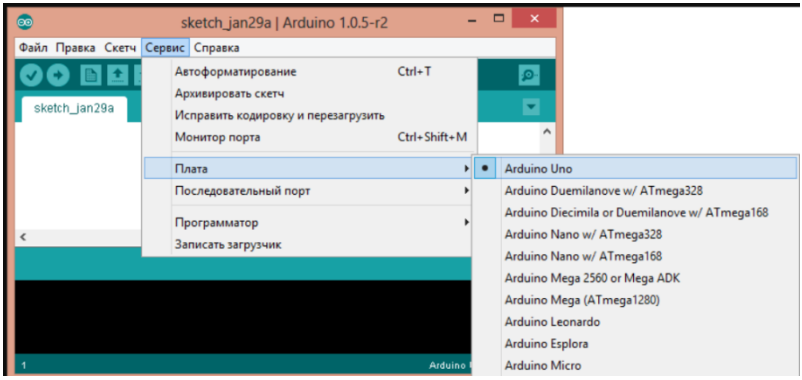


Рис. 2.23. – Вибір плати Arduino UNO

2.4.5. Завантаження першого скетча

Arduino IDE містить готові приклади попередньо вирішених задач для можливого швидкого інтегрування даних прикладів в основну програму.

Для завантаження першої програми необхідно вибрати скетч під назвою “*Blink*” (рис. 2.24).

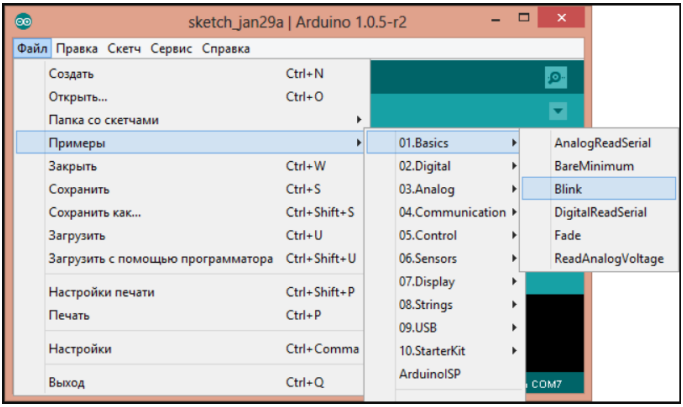


Рис. 2.24. – Вибір програми "Blink"

Для завантаження програми "*Blink*" необхідно натиснути кнопку "*Загрузить*" (рис. 2.25).

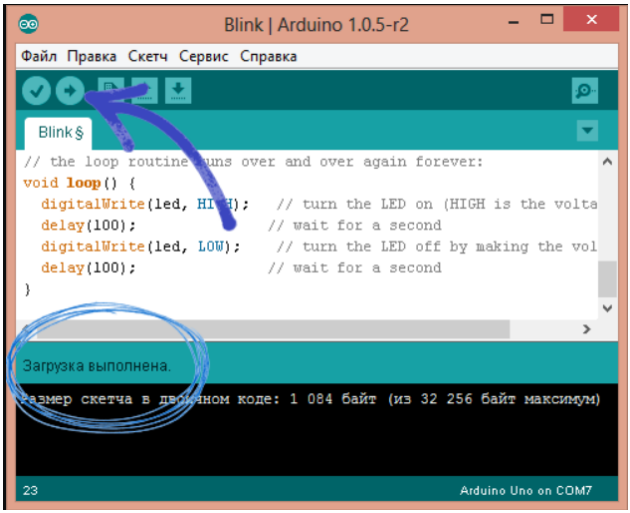


Рис. 2.25. – Завантаження програми "Blink"

2.4.6. Встановлення бібліотеки *Braccio*

Для забезпечення функціонування стенду (див. далі рис 2.31) та керування його складовими необхідно використовувати

бібліотеку “*Braccio*”, для завантаження якої необхідно перейти за посиланням https://github.com/arduino-org/arduino-library-braccio (рис. 2.26).

Для підключення завантаженої бібліотеки “*Braccio*” необхідно виконати наступні дії: “*Скетч*” → “*Подключить библиотеку*” → “*Добавить .ZIP библиотеку…*” (рис. 2.27).

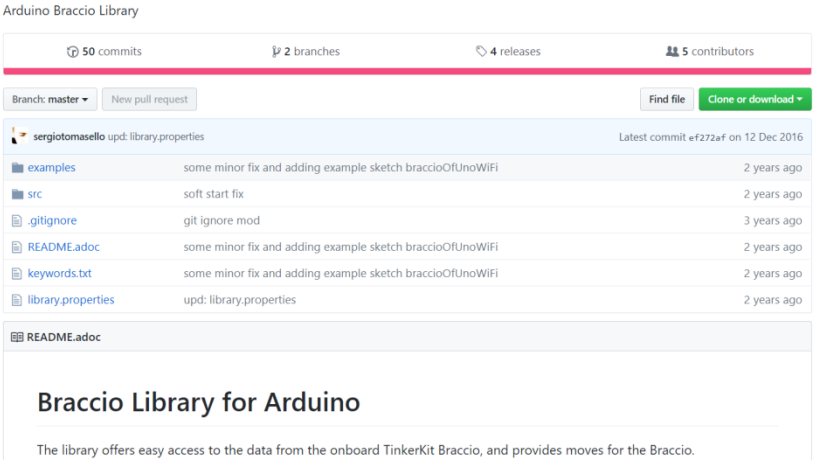


Рис. 2.26. – Ресурс з бібліотекою “Braccio” для TinkerKit Braccio

Програмний код має стандартну структуру лише в частині, де кожному двигуну надається своє значення, а синтаксис відрізняється. Приклад скетчу на просте переміщення в циклі з коментарями показано на рис. 2.28. В червоній рамці виділені коментарі, а в структурі *Braccio.ServoMovement* прописані значення кутів двигунів та затримки роботи між двигунами.

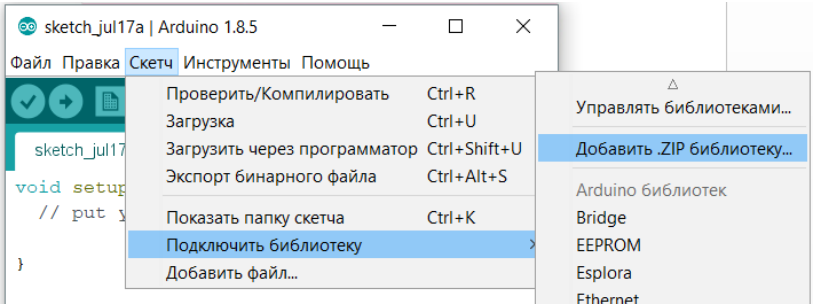


Рис. 2.27. – Підключення бібліотеки "braccio" для TinkerKit Braccio

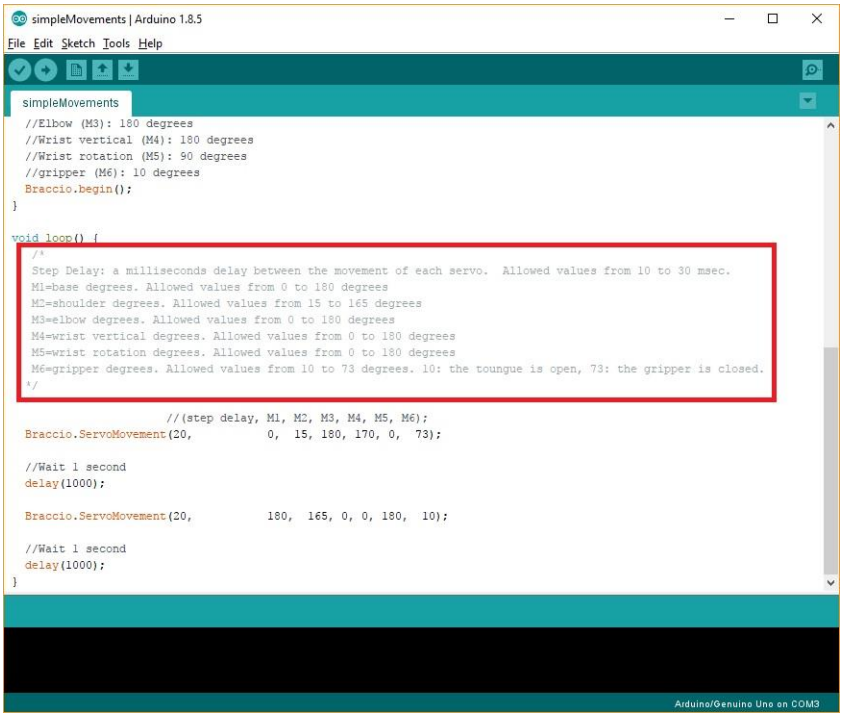


Рис. 2.28. – Приклад простого коду для робота

2.5. Структура та зміст програми

Програма відпрацювання переміщень ланок робота складається з декількох складових:

- підключення бібліотек (*#include<ім'я файлу>*);
- функція ініціалізації (*void setup ()*);
- основна функція (*void loop()*).

За допомогою команди ***#include<ім'я файлу>*** виконується підключення двох допоміжних бібліотек (<***Braccio.h***> та <***Servo.h***>). Бібліотека <***Braccio.h***> забезпечує коректну роботу плати *Arduino Uno* з платою драйвером *Braccio Shield* та передачу керуючих ШІМ-сигналів до серводвигунів. Для коректної роботи і точного відпрацювання кутів серводвигунами необхідно підключити бібліотеку <***Servo.h***>.

У функції ініціалізації ***void setup()*** ініціалізується і запускається основна функція з бібліотеки <***Braccio.h***> за допомогою команди ***Braccio.begin()***. При спрацюванні цієї функції починається подача живлення і керуючих ШІМ-сигналів до серводвигунів робота.

Основна функція ***void loop ()*** забезпечує виконання програмного коду в циклі, тобто програма буде повторно виконуватись до завершення. Для відпрацювання конкретних кутів окремих ланок робота у функції ***void loop ()*** використовується структура ***Braccio.ServoMovement(…).*** У даній структурі першою прописується тривалість затримки кроку, яка визначає кількість мілісекунд між покроковим відпрацюванням кута кожного серводвигуна. Наступні значення визначають кути повороту вала кожного серводвигуна, починаючи з М1 і завершуючи М6.

2.6.Структура та зміст лабораторного стенду

Лабораторний стенд складається з таких складових груп:

- елементи управління;
- виконавчі механізми;
- допоміжні компоненти.

В групу ***елементи управління*** входять наступні компоненти:

- плати *Arduino Uno* (2 шт.);

- драйвери *Braccio Shield* (2 шт.);
- панель управління (для ручного керування 1шт.).

До групи ***виконавчі механізми*** входить два робота моделі TinkerKit Braccio. Опис роботів представлений в п. 2.1.1.

Група ***допоміжні компоненти*** складається з ТО (технологічних об'єктів) і станини, на якій встановлені роботи і ***елементи управління.***

На рис. 2.29 зображено лабораторний стенд з розміщенням його складових елементів.

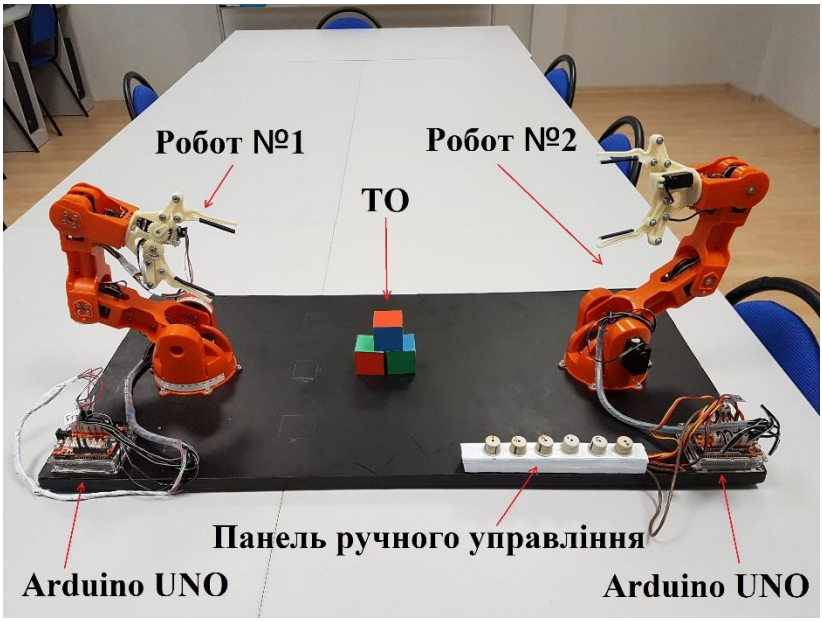


Рис. 2.29. – Лабораторний стенд

Виконавчі механізми стенду працюють за керуючими сигналами, які надходять з ***елементів управління***, а саме підсилений, за допомогою драйверів, керуючий сигнал від контролера *Arduino Uno* до серводвигунів.

Приведення стенду в робочий стан складається з наступних кроків:

- розробка програми;
- завантаження програми у контролер;
- підключення компонентів до джерела живлення;
- запуск і перевірка роботи програми.

Під час кроку ***розробка програми*** розробляється код траєкторії рухів ланок МС та перевіряється правильність написання програми у *програмному середовищі* ***Arduino IDE.*** Виконується компіляція коду.

Наступним кроком є ***завантаження програми у контролер***, під час якого завантажується код у контролер з ПК через порт *USB.*

Підключення компонентів до джерела живлення передбачає встановлення роботів у безпечне положення та підключення блоків живлення до мережі.

УВАГА! НЕ ДОПУСКАЄТЬСЯ ОДНОЧАСНЕ ЖИВЛЕННЯ ПЛАТИ *Arduino Uno* ЧЕРЕЗ *USB* ТА СПЕЦІАЛЬНОГО РОЗ'ЄМУ!

Останнім кроком є ***запуск і перевірка програми.*** Результатом є демонстрація правильності роботи програми за варіантом індивідуального завдання.

2.7. Приклад виконання завдання та програми простих переміщення ланок МС робота *TinkerKit Braccio*

2.7.1. Завдання

Запрограмувати переміщення ланок робота зі стартової (початкової) позиції в кінцеву для абстрактного варіанту ***:

Варіант №	Кути ланок стартової позиції							Кути ланок кінцевої позиції					
	M1	M2	M3	M4	M5	M6		M1	M2	M3	M4	M5	M6
***	0	15	180	170	0	73		180	165	0	0	180	10

2.7.2. Загальні рекомендації

Загальна схема виконання завдання включає наступні основні етапи Е:

Е1. Встановлення та запуск додатку, бібліотек та ініціалізація двигунів під відповідну бібліотеку (див. далі кроки К1 – К5 п. 2.7.3);

Е2. Запис стартового, поточного та кінцевого положень ланок МС робота (кроки К6 – К8 п. 2.7.3);

Е3. Складання коду програми за варіантом індивідуальних завдань;

Е4. Підключення плати *Arduino Uno* до ПК та завантаження програми в нього;

Е5. Перевірка працездатності складеного коду програми на лабораторному стенді (крок К11).

2.7.3. Виконання завдання

Завдання виконується наступною послідовністю кроків К:

К1. запустити додаток (програмне середовище *Arduino IDE)*;

К2. створити новий скетч (файл > новий, або *Ctrl + N* тобто створення нового скутча);

К3. підключити бібліотеки за допомогою команди (*#include*< ***Braccio.h*** > та *#include*< ***Servo.h*** >);

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 бібліотеку "*Braccio.h*" за допомогою функції ***void setup ()*** та команди ***Braccio.begin();***

К6. записати стартове положення (стартову позицію) ланок маніпулятора робота у функції ***void loop ()*** за допомогою команди ***Braccio.ServoMovement*** **(20, 90, 90, 90, 90, 90, 73);**

та задати затримку в 1 секунду за допомогою команди ***delay*****(1000);**

К7. записати початкове положення під №1 ланок маніпуляційної системи робота у функції ***void loop ()*** за допомогою команди:

Braccio.ServoMovement **(20, 0, 15, 180, 170, 0, 73);**

та задати затримку в 1,5 секунди за допомогою команди ***delay*****(1500);**

К8. записати кінцеве положення під №2 ланок маніпуляційної системи робота у функції ***void loop ()*** за допомогою команди:

Braccio.ServoMovement **(20, 180, 165, 0, 0, 180, 10);** та задати затримку в 0,5 секунди за допомогою команди ***delay*****(500);**

К9. підключити плату *Arduino UNO* до ПК та обрати відповідний *COM*-порт;

К10. завантажити складену за варіантом індивідуального завдання програму в *Arduino UNO*;

К11. перевірити роботу програмного коду на стенді.

Приклад частини програмного коду має вигляд:

Braccio.ServoMovement(20, 0, 15, 180, 170, 0, 73);

Це означає:

- 20 мілісекунд затримка кроку;
- М1 відпрацьовує кут 0°;
- М2 відпрацьовує кут 15°;
- М3 відпрацьовує кут 180°;
- М4 відпрацьовує кут 170°;
- М5 відпрацьовує кут 0°;
- М6 відпрацьовує кут 73°.

Нижче представлений програмний код для прикладу, у якому виконується відпрацювання роботом трьох позицій, що умовно названі “стартова”, “кобра” та “обернена кобра”. Символ “//” активує рядок коментаря, який не впливає на програмний код, а має описовий, інформативний характер.

Загальний вид процюючого програмного коду наступний:

```
#include <Braccio.h> //  команда #include <Braccio.h>,
                         підключає бібліотеку для роботи з
                         роботами
#include <Servo.h> //    команда #include <Servo.h, підключає
                         бібліотеку для роботи з
                         серводвигунами

// ініціалізація окремих серводвигунів під бібліотеку

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)

void setup() {
  // Ініціалізація функцій і встановлення ланок робота в
     початкову позицію для бібліотеки Braccio ()
  // Всі серводвигуни будуть встановлені в "безпечну"
     позицію:
  // База – ланка L2, двигун (M1): 90 градусів
  // Плече – ланка L3, двигун (M2): 45 градусів
  // Ланка – L4, двигун (M3): 180 градусів
  // Ланка – L5, двигун (M4): 180 градусів
  // Кисть – ланка L6, двигун (M5): 90 градусів
  // Схват – (gripper) ланка L8, двигун (M6): 10 градусів

  Braccio.begin(); // запуск допоміжної функції
}
// void loop() – це основна функція програми, яка буде
виконуватись в циклі
(повторюватись після завершення виконання)
void loop() {
```

/* коментар починається зі знаку “/*” та закінчується знаком “*/”; між даними знаками немає необхідності починати кожен рядок коментарів зі знаку “//”

Затримка кроку: кількість мілісекунд між покроковим відпрацюванням кутів валом кожного серводвигуна.

Допустипі значення від 10 до 30 мілісекунд із кроком 1 мілісекунда.

M1=База, ланка L2. Допустимі значення від 0 до 180 градусів

M2=Плече, ланка L3. Допустимі значення від 15 до 165 градусів

M3=Ланка L4. Допустимі значення від 0 до 180 градусів

M4=Ланка L5. Допустимі значення від 0 до 180 градусів

M5=Кисть, ланка L6. Допустимі значення від 0 до 180 градусів

M6=Схват, ланка L8. Допустимі значення від 10 до 73 градусів. (10 - схват відкритий, 73 - схват закритий).

*/

// встановлення ланок маніпуляційної системи робота у "стартове" вертикальне положення всіх ланок, схват закритий

//(Затримка кроку, M1, M2, M3, M4, M5, M6);

Braccio.ServoMovement(20, 90, 90, 90, 90, 90, 73);

//Затримка 1 секунда

delay(1000); // затримка в мс (1с = 1000 мс) між блоками
// команд до затримки та після.
// Рекомендовано запрограмувати тривалість затримки
//достатньою для сприйняття кожним студентом (бригадою
//студентів) відпрацювання команд, наприклад, 5, 10 секунд
//тощо. Можливий інтервал програмних затримок: 0 – 32767
//мс (за технічною документацією Arduino Uno).

// встановлення ланок маніпуляційної системи робота у положення #1 "кобра" (термін умовний), схват закритий

//(Затримка кроку, M1, M2, M3, M4, M5, M6);

Braccio.ServoMovement(20, 0, 15, 180, 170, 0, 73);

//Затримка 1,5 секунди

delay(1500);

// Див. вище

```
  // встановлення ланок маніпуляційної системи робота у положення #2 "обернена кобра"(термін умовний), схват закритий
  //(Затримка кроку, M1, M2, M3, M4, M5, M6);
  Braccio.ServoMovement(20,          180,  165, 0, 0, 180,  10);

  //Затримка 0,5 секунди
  delay(500);
}
//Див. вище
```

Вказане вище має екранну форму, що подана на рис. 2.30.

Example_UA §

```
#include <Braccio.h> // <-- команда #include <Braccio.h>, підключає бібліотеку для роботи з роботами
#include <Servo.h> // <-- команда #include <Servo.h, підключає бібліотеку для роботи з серводвигунами

// ініціалізуємо окремі серводвигуни під бібліотеку //

Servo base; // серводвигун M1 ланка L1 (база)
Servo shoulder;// серводвигун  M2 ланка L2 (плече)
Servo elbow;// серводвигун M3 ланка L3
Servo wrist_rot;// серводвигун M4 ланка L4
Servo wrist_ver;// серводвигун M5 ланка L5 (кисть)
Servo gripper;// серводвигун M6 ланка L6 (схват)

void setup() {
  // Ініціалізація функцій і встановлення ланок робота в початкову позицію для бібліотеки Braccio ()
  // Всі серводвигуни будуть встановлені в "безпечну" позицію:
  // База ланка L1 двигун (M1):90 градусів
  // Плече ланка L2 двигун (M2): 45 градусів
  // Ланка L3 двигун (M3): 180 градусів
  // Ланка L4 двигун (M4): 180 градусів
  // Кисть ланка L5 двигун (M5): 90 градусів
  // Схват (gripper) ланка L6 двигун (M6): 10 градусів

  Braccio.begin(); // запуск допоміжної функції
}
// void loop() це основна функція програми яка буде виконуватись в циклі
(повторюватись після завершення виконання)
void loop() {
  /*
   Затримка кроку: кількість мілісекунд між покрокового відпрацювання кута кожного серводвигуна.
   Допустипі значення від 10 до 30 мілісекунд.
   M1=База ланка L1. Допустимі значення від 0 до 180 градусів
   M2=Плече ланка L2. Допустимі значення від 15 до 165 градусів
   M3=Ланка L3. Допустимі значення від 0 до 180 градусів
   M4=Ланка L4. Допустимі значення від 0 до 180 градусів
   M5=Кисть ланка L5. Допустимі значення від 0 до 180 градусів
   M6=Схват ланка L6. Допустимі значення від 10 до 73 градусів. (10: схват відкритий, 73: схват закритий)
  */

  // встановлення маніпуляційної системи робота у "стартове" вертикальне положення, схват закритий
  //////////////////////(Затримка кроку, M1, M2, M3, M4, M5, M6);
  Braccio.ServoMovement(20,             90, 90, 90, 90, 90, 73);

  //Затримка 1 секунда
  delay(1000);

  // встановлення маніпуляційної системи робота у положення #1 "кобра", схват закритий
  //////////////////////(Затримка кроку, M1, M2, M3, M4, M5, M6);
  Braccio.ServoMovement(20,           0,  15, 180, 170, 0,  73);

  //Затримка 1,5 секунди
  delay(1500);

  // встановлення маніпуляційної системи робота у положення #2 "обернена кобра", схват закритий
  //////////////////////(Затримка кроку, M1, M2, M3, M4, M5, M6);
  Braccio.ServoMovement(20,           180,  165, 0, 0, 180,  10);

  //Затримка 0,5 секунди
  delay(500);
```

Рис. 2 30. – Екранна форма програмного модуля представленого вище прикладу

2.8. Порядок виконання роботи

1. Вивчити теоретичні відомості.
2. Виконати попередню підготовку обладнання як вказано в п. 1.4.
3. Запустити додаток *Arduino IDE* та підключити плату *Arduino Uno* до комп'ютера за допомогою *USB* кабелю.

Схема з'єднання комп'ютера із контролером *Arduino Uno* представлена на рис. 2.31.

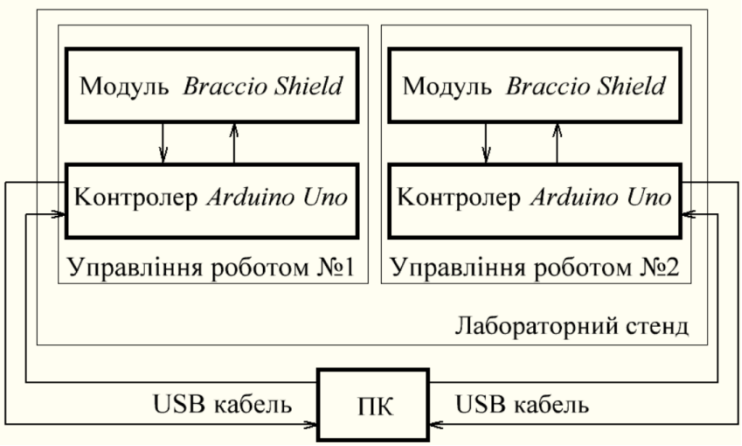


Рис. 2.31. – Схема з'єднання комп'ютера із контролером Arduino Uno

4. Написати програму переміщення ланок МС у циклі згідно із варіантом індивідуальних завдань, вказаних в табл. 2.3.
5. Завантажити програму на плату *Arduino Uno* та від'єднати її від ПК.
6. Поставити ланки робота у безпечне положення та подати живлення на *Braccio Shield* за допомогою блока живлення через відповідний роз'єм.
7. Продемонструвати відпрацювання переміщення ланок МС робота, заданих варіантом.
8. Зняти відео та завантажити його в хмарне сховище (наприклад, в Google drive).
9. Створити QR-посилання на відео.

10. Оформити зміст звіту.

2.9. Зміст звіту

1. Назва та мета роботи.

2. Короткі теоретичні відомості щодо робота моделі *TinkerKit Braccio*.

3. Основні відомості щодо контролера *Arduino Uno.*

4. Схема, склад лабораторного стенду та опис його роботи.

5. Основні відомості про структуру та зміст програми.

6. Індивідуальне завдання за варіантом лабораторної роботи.

7. Програма за варіантом лабораторної роботи.

8. Представити QR–посилання відеодемострації працездатності коду за варіантом індивідуальних завдань.

9. Висновки по роботі.

2.10. Варіанти індивідуальних завдань

1 . 3 . – *Т а б л и ц я*

Варіант №	Кути ланок стартової позиції					Кути ланок кінцевої позиції				
	М1	М2	М3	М4	М5	М1	М2	М3	М4	М5	М6
1	10	90	30	10	90	160	90	110	50	180	60
2	95	90	85	60	90	0	90	130	90	180	60
3	20	90	50	90	90	180	90	120	0	180	60
4	35	90	10	85	90	145	90	90	0	180	60
5	70	90	80	45	90	180	90	130	125	180	60
6	120	90	120	60	90	10	90	0	120	180	60
7	30	90	20	75	90	160	90	100	160	180	60
8	160	90	60	30	90	15	90	95	0	180	60
9	45	90	110	70	90	90	90	60	15	180	60
10	60	90	180	160	90	180	90	90	40	180	60
11	60	90	130	110	90	0	90	50	80	180	60
12	55	90	15	95	90	155	90	110	0	180	60
13	110	90	65	45	90	20	90	150	135	180	60
14	170	90	95	120	90	10	90	0	40	180	60
15	0	90	100	80	90	180	90	180	120	180	60
16	150	90	75	150	90	20	90	0	0	180	60
17	5	90	25	160	90	175	90	90	60	180	60
18	180	90	150	55	90	15	90	120	135	180	60
19	85	90	85	115	90	160	90	0	15	180	60
20	130	90	15	145	90	30	90	160	75	180	60

2.11. Контрольні питання

1. Надати короткий опис робота *Braccio TinkerKit*.
2. Накреслити структурну схему робота *Braccio TinkerKit*.
3. Склад серводвивигуна. Принцип керування серводвигуном.
4. Поняття *Arduino IDE*.
5. Короткий опис контролера *Arduino Uno*.
6. Опис входів і виходів контролера *Arduino Uno*.
7. Опис складових програми для робота за вказаним вище завданням.
8. Основні складові лабораторного стенду.
9. Стислий опис функціонування лабораторного стенду.

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.

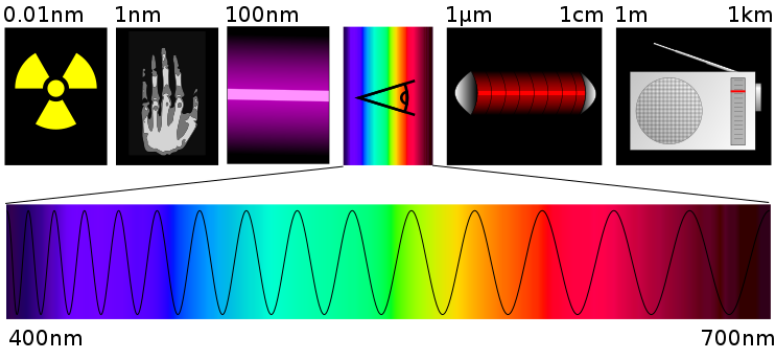


Рис. 3.1. – Видиме світло на електромагнітній шкалі

Як і будь-які інші електромагнітні хвилі, світло характеризується частотою, довжиною хвилі, поляризацією й 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 датчику (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*, *STL*. Виходи продемонстровані на рис. 3.3.

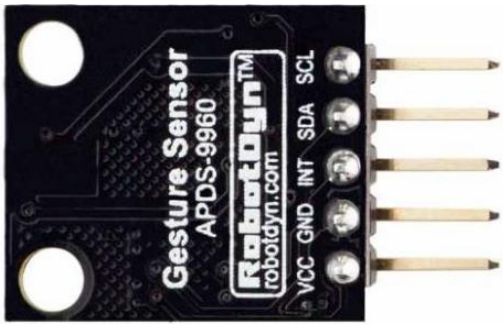


Рис. 3.3. – Виходи датчика кольору RobotDyn APDS-9960

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. 2.4 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***.

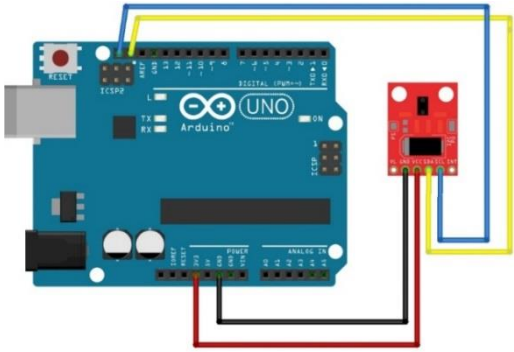


Рис. 3.4. – Схема підключення датчика кольору RobotDyn APDS-9960 до плати Arduino Uno

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

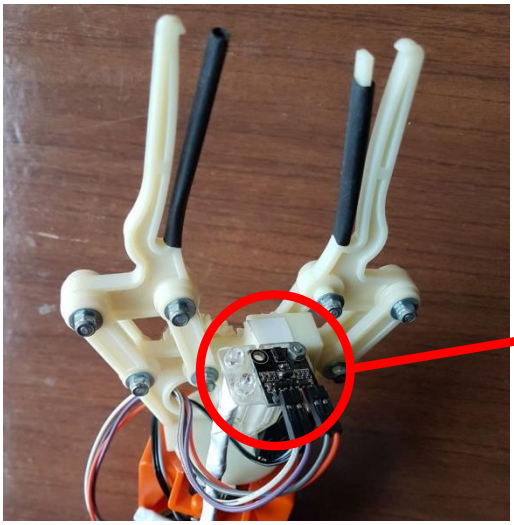


Рис. 3.5. – Розміщення датчика кольору RobotDyn APDS-9960 на роботі TinkerKit Braccio

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 маленьких 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 функціїї 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.

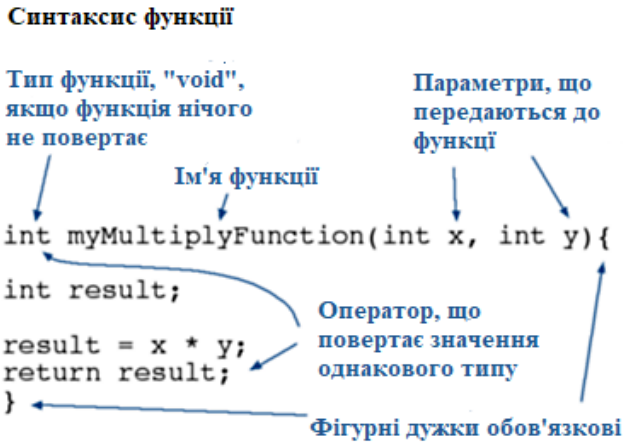


Рис. 3.6. – Синтаксис функції при програмування Arduino Uno

Сама функція може приймати значення певного типу. Цей тип вказується перед ***ім'ям функції***. Значення, яке приймає функцію, 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. 2.7).

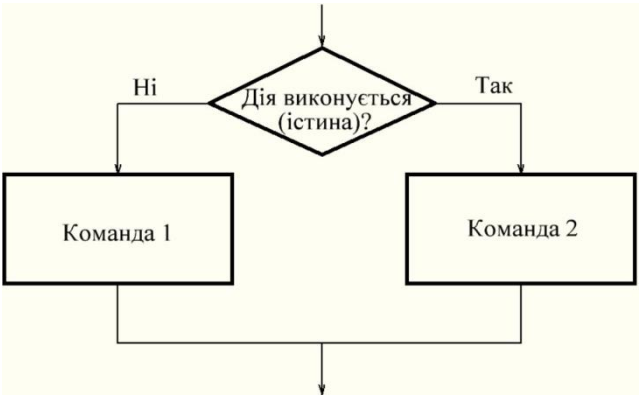


Рис. 3.7. – Приклад умовного оператора

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*", що 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 — 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 №2 моделі *TinkerKit Braccio* (див. рис. 3.29), 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 окружностей, 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.

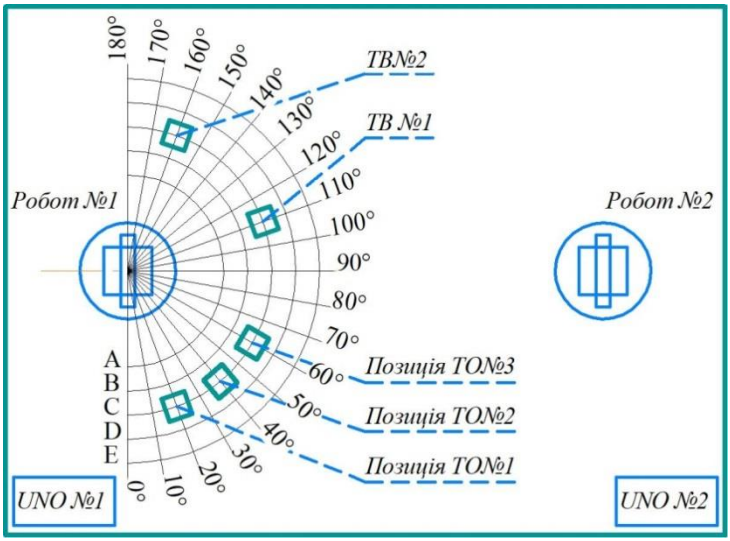


Рис. 3.8. – Структурна схема лабораторного стенду з активним роботом №1 з точками ТО та ТВ

В наведеному нижче прикладі надано код (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 – С 20°, ТО №2 – С 40°, ТО №3 – С 60°, а позиції ТВ №1 та ТВ №2 відповідно С 110° та С 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).

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

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 2.9. 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. 2.10) 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 кожним 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.

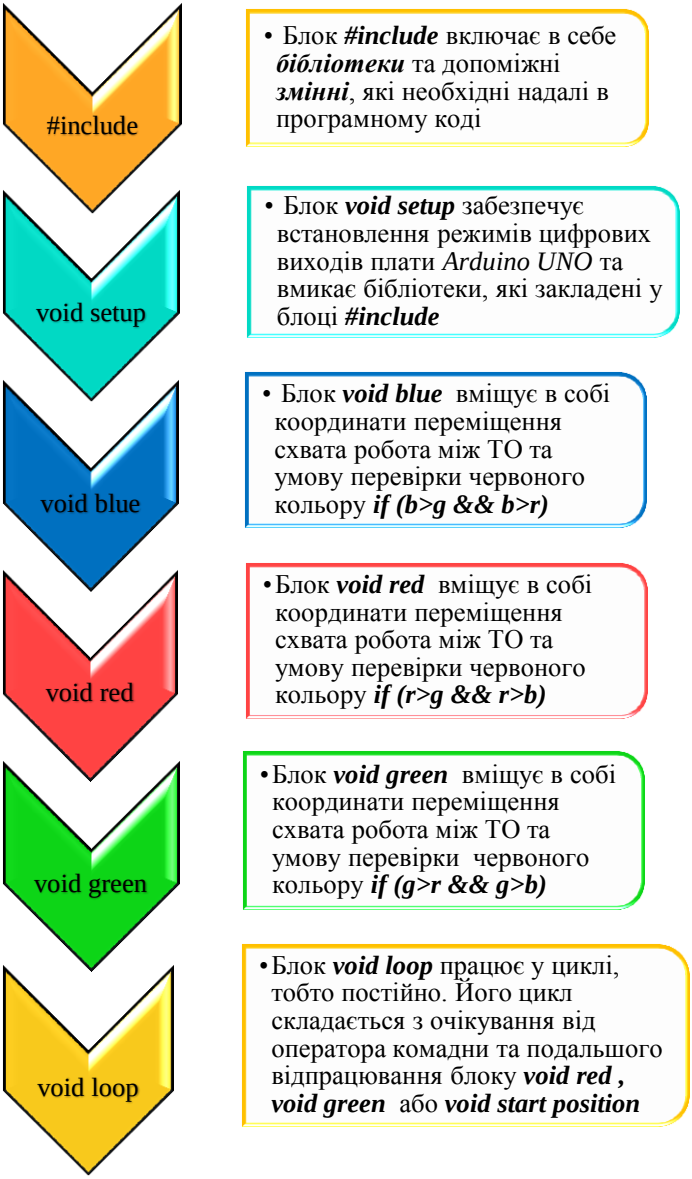


Рис. 3.9. – Структурна схема послідовностей складових елементів коду УП функціонування робота

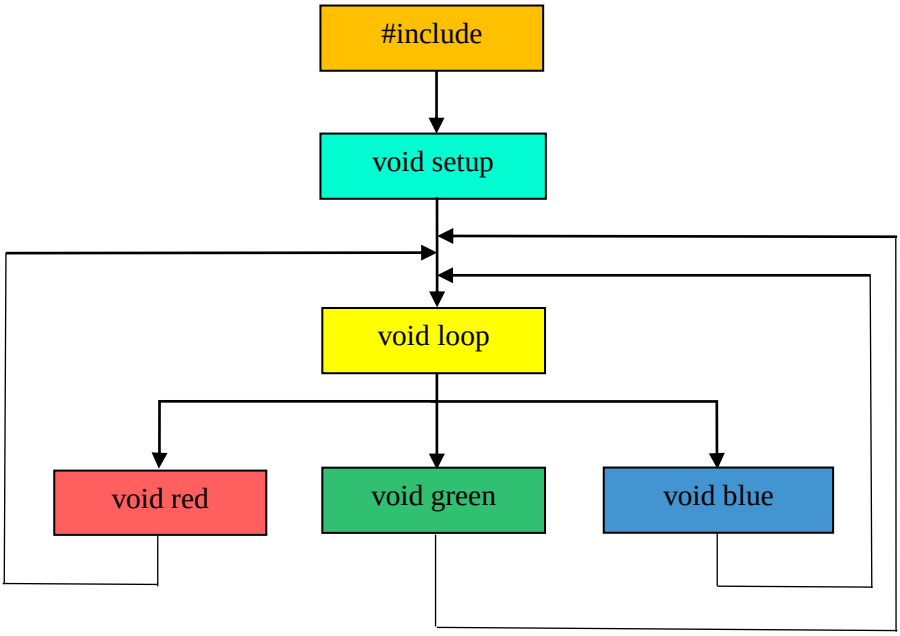


Рис. 3.10. – Схема-алгоритм складових коду УП функціонування робота

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.

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

Нижче представлено програмний код наведеного нижче прикладу. При цьому кольоровий фон відповідає 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 не виконанні 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

11. Ознайомитись та засвоїти теоретичні відомості.
12. Перевірити підключення датчика кольору *RobotDyn APDS-9960* до плати *Arduino UNO.*
13. Запустити додаток *Arduino IDE* та підключити плату *Arduino UNO* до комп'ютера за допомогою *USB*-кабелю.
14. Завантажити програму-приклад на плату *Arduino UNO* та відкрити монітор порту для комунікації з ПК (виставити *бітрейт* 115200**).**
15. Розташувати ТО на стенді у визначених місцях (позначено на основі стенду позначками).
16. Встановити всі ланки робота у безпечне положення та подати живлення на *Braccio Shield* за допомогою блока живлення через відповідний роз'єм.
17. Дочекатись ініціалізації датчика кольору: на моніторі порту буде виведено повідомлення "Device initialized**!**".
18. Подати команду на запуск функції пошуку ТО прописом відповідним символом та вводу клавішою *Enter* на ПК, спостерігати за ходом виконання програми.
19. 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.;
20. 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*.
21. Продемонструвати відпрацювання роботом позицій, заданих за варіантом.
22. Записати відео роботи програми пошуку ТО за кольором, завантажити відео у хмарне сховище (довільний вибір) та згенерувати *QR*-код посилання на відео.
23. 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 №1**

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°

Продовження табл. 3.2

1	С 20°	С 40°	С 60°	**Червоний** С 130°	**Синій** С 180°
2	С 20°	С 40°	С 60°	**Синій** С 130°	**Зелений** С 180°
3	С 20°	С 40°	С 60°	**Зелений** С 130°	**Червоний** С 180°
4	С 20°	С 40°	С 60°	**Червоний** С 130°	**Зелений** С 180°
5	С 20°	С 40°	С 60°	**Синій** С 130°	**Червоний** С 180°
6	С 40°	С 60°	С 80°	**Зелений** С 100°	**Синій** С 150°
7	С 40°	С 60°	С 80°	**Червоний** С 100°	**Синій** С 150°
8	С 40°	С 60°	С 80°	**Синій** С 100°	**Зелений** С 150°
9	С 40°	С 60°	С 80°	**Зелений** С 100°	**Червоний** С 150°
10	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.*

Лабораторна робота №4

ОСНОВИ ПРОГРАМУВАННЯ КОМПЛЕКСУ З ДВОХ РОБОТІВ *Braccio* В ПРОГРАМНОМУ СЕРЕДОВИЩІ *Arduino IDE*

Мета роботи – отримати практичні навички програмування узгодженого функціонування двох роботів *Braccio TinkerKit* в середовищі *Arduino IDE*.

4.1. Теоретичні відомості

4.1.1. Основі відомості про обмін даними між мікроконтролерами *Arduino UNO*

UART (англ. Universal asynchronous receiver/transmitter – універсальний асинхронний приймач/передавач) – тип асинхронного приймача-передавача, компонентів комп'ютерів та периферійних пристроїв, що передає дані між паралельною та послідовною формами. *UART* звичайно використовується спільно з іншими комунікаційними стандартами, такими як *RS-232*.

Дані *UART* передаються послідовним кодом у наступному форматі, що схематично представлені на рис. 4.1.

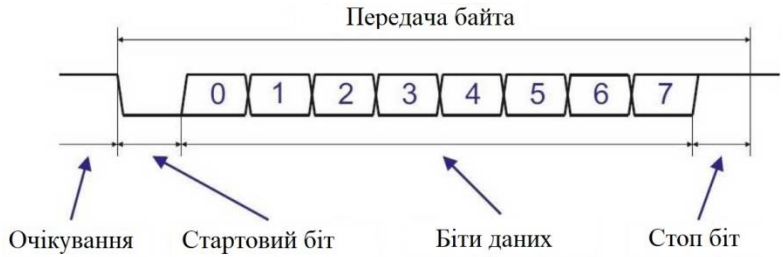


Рис. 4.1 – Передача даних послідовним кодом UART

При цьому кожен біт передається за рівні проміжки часу. Час передачі одного біта визначається швидкістю передачі. Швидкість передачі вказується в бодах (біт на секунду). Крім

бітів даних інтерфейс *UART* використовує в потоці біти синхронізації: стартовий і стоповий. Таким чином, для передачі байту інформації потрібно 10 бітів, а не 8. Похибка тимчасових інтервалів передачі бітів повинна бути не більше 5% (рекомендується не більше 1,5%).

Існують варіанти з різною кількістю бітів даних, бітів синхронізації, може бути доданий біт контролю парності і т.п. Але ці формати використовуються рідко.

Основні пункти на які варто звернути увагу при:

- в неактивному режимі вихід *UART* знаходиться в високому стані (HIGH state);
- передача байта починається зі стартового біта (низького рівня);
- передача байта закінчується стоповим бітом (високого рівня);
- дані передаються молодшим бітом вперед;
- для передачі байта потрібно 10 бітів;
- час передачі одного байта розраховується виходячи з швидкості передачі і кількості бітів (10).

Часто використовуються такі стандартні швидкості передачі інтерфейсу *UART* (табл. 4.1):

Таблиця 4.1 - Стандарти швидкостей передачі даних

Швидкість передачі, бод	Час передачі одного біта, мкс	Час передачі байту, мкс
4800	208	2083
9600	104	1042
19200	52	521
38400	26	260
57600	17	174
115200	8,7	87

Обмін інформацією через *UART* відбувається в дуплексному режимі, тобто передача даних може відбуватися одночасно з прийомом. Для цього в інтерфейсі *UART* є два сигнали:

- *TX* – вихід для передачі даних (*MASTER*);
- *RX* – вхід для прийому даних *(SLAVE).*

При з'єднанні двох *UART* пристроїв вихід *TX* першого пристрою з'єднується з входом *RX* другого, а сигнал *TX* другого *UART* підключається до входу *RX* першого (рис. 4.2).

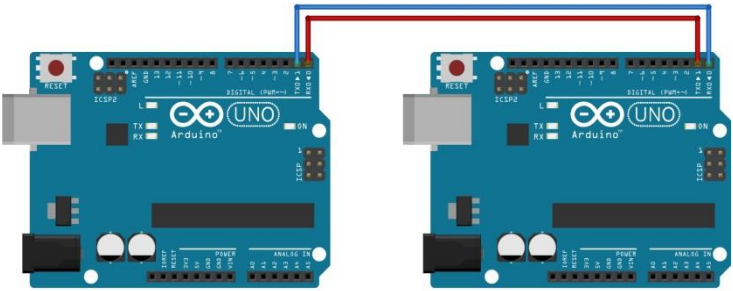


Рис. 4.2 – Схема підключення Arduino як UART- пристрою

Плата *Arduino UNO* має один порт *UART*, сигнали якого підключені до контактів 0 (сигнал RX) і 1 (сигнал TX). Сигнали мають логічні рівні *TTL* (0 ... 5 В). Через ці контакти (0 і 1) можна підключити до плати інший пристрій, який має інтерфейс *UART*.

Крім функції зв'язку з іншими контролерами порт *UART* плати *Arduino UNO* використовується для завантаження в контролер програми з комп'ютера. Для цього до цих же сигналів (*RX* і *TX*) підключені відповідні контакти мікросхеми *ATmega16U2* - перетворювача інтерфейсу *USB/UART*. Мікросхема перетворювача підключена через резистори, що мають опір 1 кОм. Таким чином, при вільних контактах 0 і 1 плати *Arduino* сигнали з мікросхеми ATmega16U2 надходять на контролер *ATmega328*. А якщо до плати підключити зовнішній *UART* пристрій, то його сигнали будуть мати пріоритет, тому що *ATmega16U2* підключена через резистори.

Перетворювач інтерфейсу *ATmega16U2* дозволяє підключати плату *Arduino* до комп'ютера через *USB* порт. На комп'ютер встановлюється драйвер. Він створює на комп'ютері

віртуальний *COM* порт. Через драйвер відбувається обмін даними.

4.1.2. Налагодження програм за допомогою послідовного порту на *Arduino*

Вплив та спостереження ходу виконання програми забезпечує монітор порту, а саме:

- за допомогою послідовного порту і функцій класу *Serial* можна передати на комп'ютер інформацію про стан програми;
- за допомогою монітора послідовного порту *Arduino IDE* або іншої програми можливо ці дані побачити на екрані комп'ютера;
- цими програмними засобами можна передати дані в програму *Arduino IDE* і вплинути на її роботу.

За рахунок функціоналу, який доступний у моніторі порту, можливо безпосередньо передавати дані до плати *Arduino.* Це забезпечує оперативний контроль та моніторинг ходу виконання програми. Монітор порту показаний на рис. 4.3.

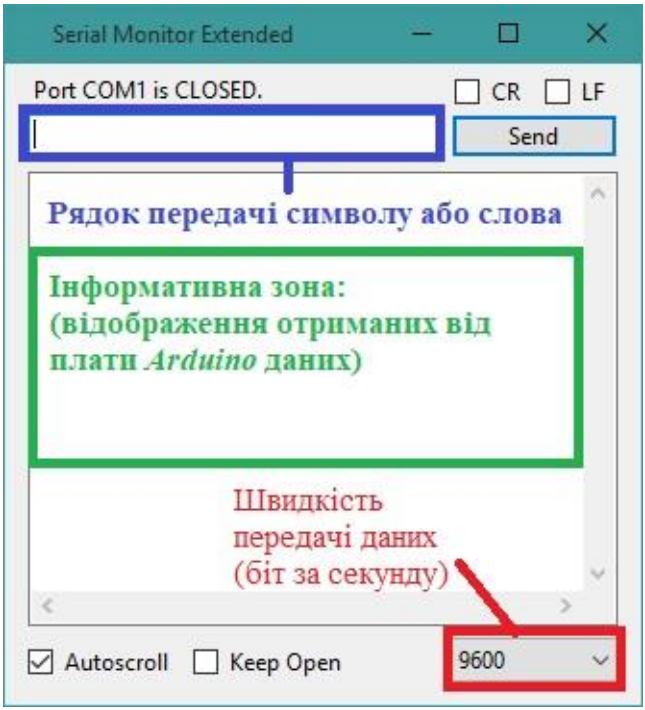


Рис. 4.3 – Скріншот монітору порту з коментарями основних елементів

4.2. Підготовка обладнання до виконання лабораторної роботи

Перед виконанням лабораторної роботи необхідно виконати наступні кроки:

1) переконатися що живлення плат *Arduino UNO* від'єднано від мережі;

2) взяти спеціальні сигнальні дроти з конекторами та візуально перевірити на наявність зовнішніх механічних пошкоджень;

УВАГА! ПРИ ВИЯВЛЕННІ ЗОВНІШНІХ МЕХАНІЧНИХ ПОШКОДЖЕНЬ (ПОШКОДЖЕННЯ ІЗОЛЯЦІЇ, СЛІДИ НАДРІЗІВ, СЛІДИ ПОТЕМНІННЯ ТА ІН.)

ПРОІНФОРМУВАТИ ЛАБОРАНТА ЧИ ВИКЛАДАЧА ТА ОТРИМАТИ НОВІ ДРОТИ.

3) за допомогою сигнальних дротів з'єднати дві плати *Arduino UNO* між собою, як наприклад, на рис. 4.2. Переконатися що конектори приєднані у виводи надійно і правильно;

4) подати живлення на плати, дочекатись візуального світлового сигналу від світлодіодів *RX* і *TX* на платі (у разі виникнення труднощів чи помилок після виконання вище описаних кроків звернутися до лаборанта чи викладача).

На рис. 4.4 показані виводи *RX* і *TX*, які дублюються на платі *Braccio Shield.*

* Нагадування - плата *Braccio Shield* під'єднується до контролера *Arduino UNO* та дублює його виводи. *

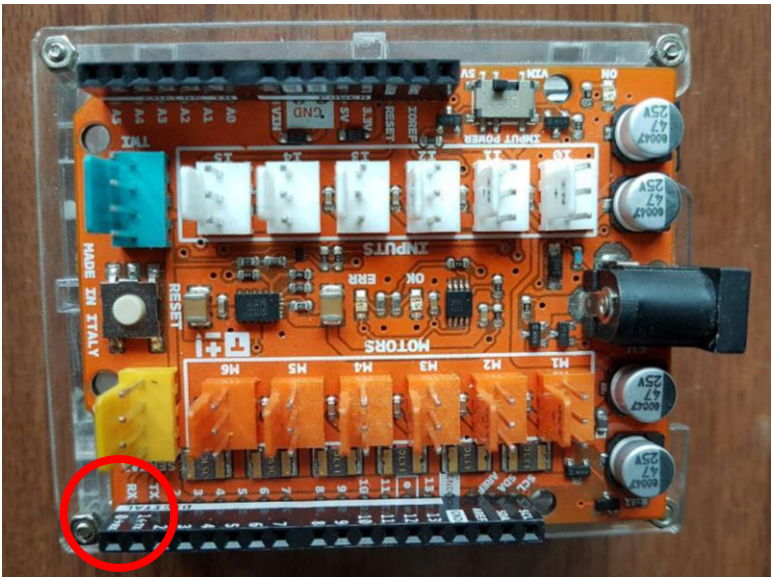


Рис. 4.4 – Фото плати Braccio Shield з позначеними виводами RX і TX

4.3. Приклад виконання завдання та програми

4.3.1. Структура стенду та рекомендації щодо роботи зі ним

Лабораторний стенд в даній лабораторній роботі включає в себе такі додаткові елементи:

- дроти *RX TX* інтерфейсу;
- кнопки управління.

Кнопки стаціонарно під'єднані до контролерів і у прикладі програмного коду використовуються як активатори старту роботи за визначеним алгоритмом. Також для коректної роботи необхідно під'єднані контролери між собою через відповідні роз'єми, як зазначено вище у пункті 3.2. Плати *Arduino UNO* передають сигнали за ієрархією *Master-Slave*, тобто одна з них головна (*Master*) передає керуючий сигнал на підпорядковану (*Slave*). Підпорядкована плата не може контролювати головну, але може надсилати дані.

На рис. 4.6 показана структурна схема стенду.

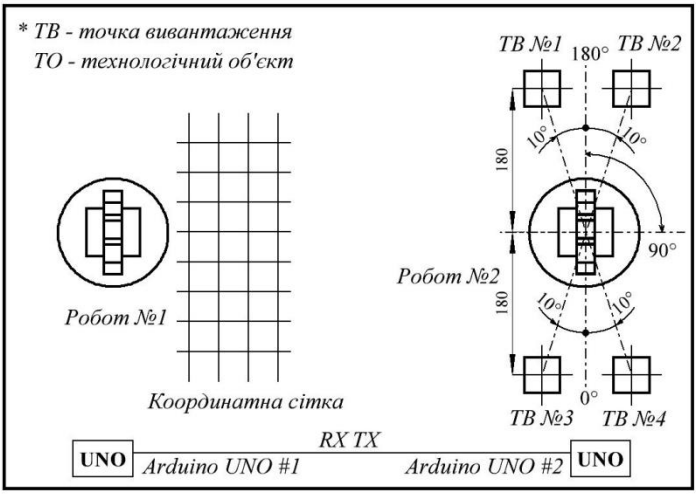


Рис. 4.6 – Структурна схема лабораторного стенду з RX і TX інтерфейсом

4.3.2. Завдання та приклад виконання

Завдання даної лабораторної роботи є:

- організувати взаємну роботу двох роботів моделі *TinkerKit Braccio* за допомогою інтерфейсу *UART (RX TX)* в ієрархії *Master-Slave*;

- запрограмувати роботів таким чином, щоб *Master* подавав керуючий сигнал на *Slave* про передачу або прийом куба схватом.

У даній лабораторній роботі робот №1 передає ТО роботу №2, після чого останній розміщує ТО в точці вивантаження (ТВ). Розміщення ТО на початковій позиції залежить від варіанта індивідуальних завдань і визначений координатною сіткою на стенді. Нижче представлений приклад виконання.

Варіант №	**Робот №1**	**Робот №2**
	Координати початкової позиції	**Точка вивантаження**
***	**B18**	**ТВ №1**

У даному варіанті прикладі координати початкової позиції ТО B18, а точка вивантаження ТВ №1. Ці точки використовувались у попередніх лабораторних роботах, тому для зручності їх було обрано для прикладу.

Завдання полягає у виконанні наступних умов:

- створення комунікації між платами *Arduino;*
- робот №1 захоплює ТО у точці B18;
- робот №2 отримує ТО від робота №1;
- робот №1 переміщує ТО у ТВ №1.

4.3.2.1. 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< **SoftwareSerial.h"** >*);

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 виходи №8 та №9 як виходи RX та TX за допомогою команди *SoftwareSerial softSerial(8, 9);*;

К6. ініціалізувати бібліотеку для ПР за допомогою функції ***void setup()*** та команди ***Braccio.begin();***

К7. створити функції ***void start()*** для початку переміщення ТО;

К8. створити функцію ***void loop ()*** та організувати в ній роботу двох ПР, з урахуванням координат початкової позиції робота №1 та точки вивантаження робота №2, згідно із варіантом індивідуальних завдань;

К9. підключити плату *Arduino Uno* до ПК та обрати відповідний *COM*-порт;

К10. завантажити програму в *Arduino Uno*;

К11. перевірити роботу коду на стенді.

4.4. Приклад виконання завдання та програми взаємодії двох плат *Arduino UNO*

У даній програмі будуть використовуватися нові команди і типи змінних.

pinMode(ім'я виходу, OUTPUT); - визначає як саме буде працювати вихід на прийом сигналів **(INPUT)** чи на передачу даних **(OUTPUT).** Приклади:

pinMode(11, OUTPUT); - вихід, ім'я якого 11, працює на передачу сигналів;

pinMode(12, INPUT); - вихід, ім'я якого 12, працює на прийом сигналів;

КОД РОБОТА №1

```
// Блок #include, підключення бібліотек
#include <SoftwareSerial.h>
#include <Braccio.h>
#include <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)

SoftwareSerial softSerial(8, 9); // визначаємо виходи №8 та №9 як RX, TX>

int incomingByte; // оголошуємо додаткову змінну
```

```
//Блок void setup
void setup(){

Serial.begin(9600); // встановлюємо швидкість передедачі данх
softSerial.begin(9600); // ініціалізуємо порт
Braccio.begin(); // початок роботи серводвигунів
}
```

```
//Блок void start
void start(){
  Braccio.ServoMovement(15,          90,  45, 160, 160, 0,  10);
  delay(100); // затримка часу у 0,1с

  Braccio.ServoMovement(15,          180,  90, 160, 180, 0,  10);
  delay(100); // затримка часу у 0,1с
```

```
Braccio.ServoMovement(15,          180,  90, 170, 180, 0,  10);
delay(100); // затримка часу у 0,1с

Braccio.ServoMovement(15,          180,  90, 170, 180, 0,  70);
delay(100); // затримка часу у 0,1с

Braccio.ServoMovement(15,          180,  90, 160, 180, 0,  70);
delay(100); // затримка часу у 0,1с

Braccio.ServoMovement(15,          88,  75, 170, 130, 90,  70);
delay(700); // затримка часу у 0,7с

Braccio.ServoMovement(15,          90,  75, 170, 130, 90,  10);
delay(100); // затримка часу у 0,1с

Braccio.ServoMovement(15,          90,  45, 160, 160, 0,  10);
delay(100); // затримка часу у 0,1с
 }
//Блок void loop
void loop(){
if (Serial.available()){   // перевіряємо отримання команд від
ПК
 incomingByte = Serial.read();
 if (incomingByte == 's') {  // відправляємо отриману команду
ПК на UART
Serial.println("Початок переміщення");
start();
delay(1000);
Serial.println('x');
softSerial.write('x');
 }
}
}
```

КОД РОБОТА №2

```
//Блок #include підключення бібліотек
#include <SoftwareSerial.h>
#include <Braccio.h>
#include <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)

SoftwareSerial softSerial(8, 9); // визначаємо виходи №8 та №9 як RX, TX>
int incomingByte; // оголошуємо додаткову змінну
```

```
//Блок void setup
void setup(){

Serial.begin(9600); // встановлюємо швидкість передедачі данх
softSerial.begin(9600); // ініціалізуємо порт
Braccio.begin(); // початок роботи серводвигунів
}
```

```
//Блок void start

void start(){
  Braccio.ServoMovement(15,          90,  135, 20, 20, 90,  10);


  delay(100); // затримка часу у 0,1с
  Braccio.ServoMovement(15,          90,  95, 47, 58, 90,  10);
```

```
  delay(3500); // затримка часу у 3,5с
  Braccio.ServoMovement(15,          90,  95, 47, 58, 90,  70);
  delay(1000); // затримка часу у 1с
  Braccio.ServoMovement(15,          90,  95, 70, 80, 90,  70);
  delay(100); // затримка часу у 0,1с
  Braccio.ServoMovement(15,          170,  110, 20, 0, 180,  70);
  delay(100); // затримка часу у 0,1с
}
```

```
//Блок void loop

void loop(){

if (softSerial.available()){   // перевіряємо отримання команд від ПК

int com = softSerial.read();   // читаємо один символ з буфера програмного порту та зберігаємо його змінну com


if (com == 'x'){ // діємо згідно команди
start(); // функція преміщення ТО
delay(1000);
}
}
}
```

4.5. Варіанти індивідуальних завдань

Таблиця 4.2. – Варіанти індивідуальних завдань

Варіант №	Робот №1		Робот №2
	Координати початкової позиції		Точка вивантаження
1	B18		ТВ №1
2	B6		ТВ №2
3	B10		ТВ №3

4	**B8**		**ТВ №4**
5	**B14**		**ТВ №1**
6	**B16**		**ТВ №2**
7	**B7**		**ТВ №3**
8	**B9**		**ТВ №4**
9	**B11**		**ТВ №1**
10	**B15**		**ТВ №2**
11	**B13**		**ТВ №3**
12	**B12**		**ТВ №4**
13	**B17**		**ТВ №1**
14	**B19**		**ТВ №2**
15	**B4**		**ТВ №3**
16	**B9**		**ТВ№1**
17	**B8**		**ТВ №2**
18	**B10**		**ТВ №1**
19	**B12**		**ТВ №1**
20	**B7**		**ТВ №4**
21	**B11**		**ТВ №3**
22	**B13**		**ТВ №1**
23	**B17**		**ТВ №4**
24	**B18**		**ТВ №2**
25	**B19**		**ТВ №3**

4.6. Контрольні питання

1. Надати короткий опис датчика **UART**;
2. Ієрархія Master-Slave, особливості контролю і передачі даних;
3. Функція Serial.begin;
4. Функція Serial.print;
5. Функція Serial.write;
6. Описати склад програми для робота.