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).

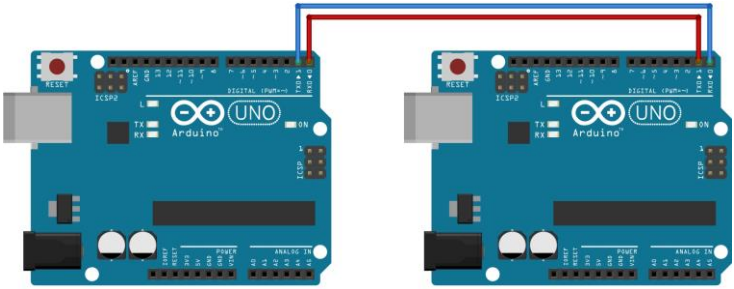

*Рис. 4.2 – Схема підключення Arduino як UART- пристрою*

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.

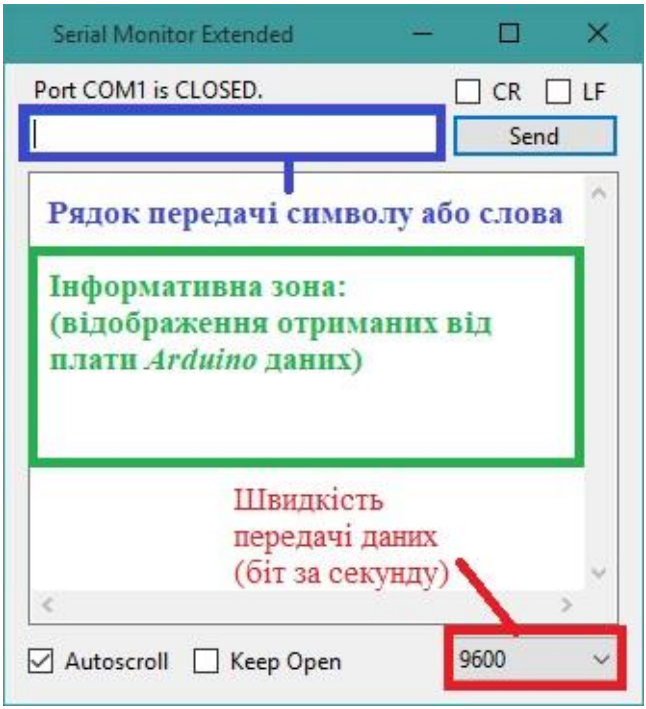


*Рис. 4.3 – Скріншот монітору порту з коментарями щодо основних елементів*

## 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, показано 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. *

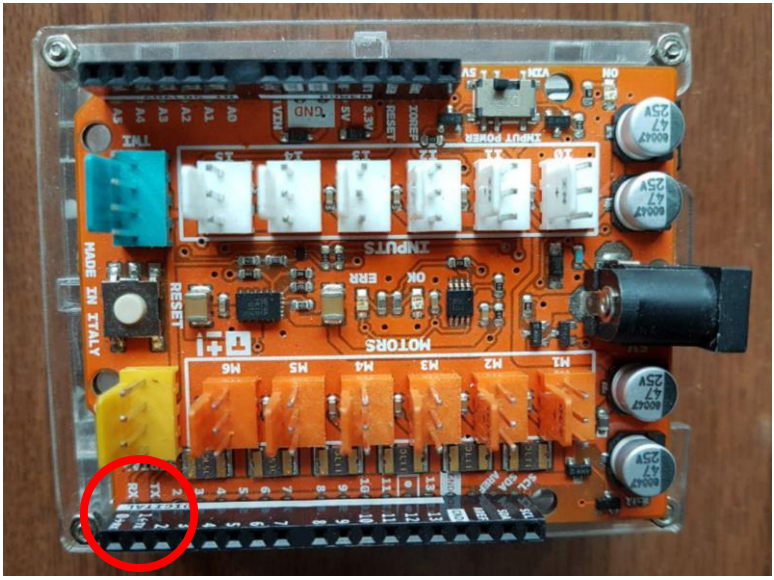


*Рис. 4.4 – Фото плати Braccio Shield з позначеними виводами RX і TX*

### 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 п. 4.2. 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.

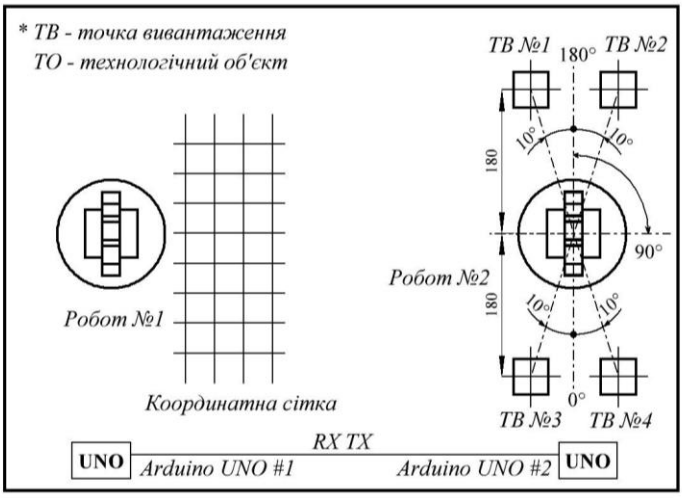


*Рис. 4.6 – Структурна схема лабораторного стенду з RX і TX інтерфейсом*

### 4.3.2. Завдання та приклад виконання
#### 4.3.2.1. Завдання

*Завданням* 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 (ТО) 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 лабораторної роботи за умовним варіантом ***.

| **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 саме їх 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.

### 4.3.2.2. 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*<***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.
**К12.** скласти звіт за вимогами.

### 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.