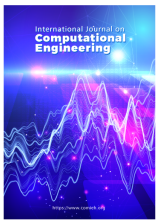
International Journal on Computational Engineering

Journal Homepage www.comien.org/index.php/comien



Mobile Robot with Integrated Arm Manipulation: A Low-Cost Solution for Remote Robotics

Omar C. Kadhum [a], Amjed Jumaah [a,*], Munther Abdul Ameer Alsudani [b], Nawal Naji Amsir [a], Ruaa Hussein Hyara [a], Husham Hadi Nghalmesh [a], Zainab AbdulKadhim Najm [a]

[a] AI Engineering Department, Collage of Engineering, Al-Ayen Iraqi University, Thi-Qar, Iraq
[b] Department of Computer Techniques Engineering, Imam Alkadhim University College, Baghdad, Iraq

Corresponding author: *amjed.baqer@alayen.edu.iq

***Abstract*—This paper presents the development of a cost-effective mobile robotic platform, equipped with a four-degrees-of-freedom (4-DOF) arm manipulator, specifically designed for educational and light industrial applications. The proposed system incorporates a tank-track locomotion mechanism for robust movement across uneven surfaces and features a Bluetooth-based wireless control interface via joystick. Utilizing affordable, off-the-shelf components—including the Arduino Uno microcontroller and SG90 servo motors—the total hardware cost of the system is limited to $150, representing a reduction of over 90% compared to commercial solutions such as the Kinova Gen3 and Robotic RB-1. Experimental trials conducted in semi-structured environments demonstrated a 93% success rate in pick-and-place operations involving objects weighing between 200 g and 500 g. additionally, the system benefits from a modular open-source architecture, enabling rapid customization and scalability. Unlike previous low-cost systems, the proposed MRAM platform integrates both mobility and manipulation capabilities without relying on proprietary hardware. This research offers a reproducible and adaptable solution for low-cost robotics, making it particularly suitable for small enterprises and academic institutions with limited resources.**

***Keywords*—Robotic arm; Mobile Robot with Arm Manipulation (MRAM); low-cost robotics; Remote Robotics Applications (RRA); Arduino; ROS.**

Manuscript received 7 Feb. 2025; revised 21 Jul. 2025; accepted 11 Aug. 2025. Date of publication 30 Sep. 2025.

I. INTRODUCTION

Mobile manipulators have become essential tools in hazardous environments such as disaster response zones [1], industrial inspection [2], and healthcare facilities [3]. However, their widespread implementation remains limited due to several critical challenges: High Cost: Commercial platforms such as the Kinova Gen3 and Robotic RB-1 are priced above $5,000 [4], rendering them inaccessible for educators and small- to medium-sized enterprises (SMEs). Limited Modularity: Many current systems lack modularity, often relying on proprietary hardware and software that restrict flexibility and customization [5]. Operational Complexity: Deploying these systems typically requires specialized training, making them difficult to use for non-experts [6].

Although recent studies have proposed affordable robotic alternatives [7], [9], these are often constrained by single-task limitations. For example, the robot base in [8] lacks manipulation capability, while the robotic arm in [7] is static. Moreover, some systems offer only 2-DOF arms with limited payloads (~200g), which may not suffice for real-world tasks [9]. In response to these gaps, we introduce a low-cost ($150) mobile manipulator that includes: A 4-DOF modular servo arm (SG90), capable of manipulating objects up to 500g; A tank-track locomotion system capable of traversing terrains with slopes up to 15°; A wireless joystick interface using Bluetooth technology.

Key advantages of this system include approximately 70% reduction in hardware costs compared to similar academic prototypes [10], an open hardware/software architecture enabled by Arduino and ROS, and a strong focus on educational deployment demonstrated in STEM workshops [11]. This paper addresses three major limitations in existing low-cost systems: (1) affordability (target cost < $200), (2) interoperability (through Arduino and ROS), and (3) wireless controllability (via Bluetooth joystick). Our system targets educational institutions and SMEs, providing a scalable,

accessible solution for mobile manipulation in constrained-resource environments [13].

A. Literature Review

Over the past decade, substantial progress has been made in the field of mobile robotics, particularly in integrating robotic manipulators with mobile platforms. This integration enables robots to navigate and interact with their environments, making them highly applicable to a wide range of domains. This section outlines the key categories of related research and highlights the unique contributions of the present study in addressing existing gaps.

1) Portable Remote Robots

Mobile robots have long been employed in remote operations such as industrial inspection, agriculture, and disaster response, dating back to the late 1970s. Studies like [14] have demonstrated the effectiveness of mobile platforms in traversing rugged terrain and collecting critical data in hazardous environments. However, most of these robots lack mechanical manipulation capabilities, limiting their ability to physically interact with objects in their surroundings.

2) Integration of Mobility and Manipulation

Combining mobility with manipulation has become a central research theme, especially for applications like industrial pick-and-place tasks. For example, Kaelbling and Lozano-Pérez [15] proposed a system that integrates a robotic arm with a wheeled mobile base. While such systems offer high flexibility and configuration potential, they are often prohibitively expensive and complex to operate, making them unsuitable for educational settings or low-resource environments.

3) Low-Cost Robotic Solutions

The emergence of open-source platforms and cost-effective hardware components has facilitated the development of low-cost robotic systems. Research by Lee et al. [16] and Rossi et al. [17] shows that using off-the-shelf components and open-source software can significantly reduce the overall system cost without sacrificing functionality. Despite these advances, many low-cost systems are tailored for specific tasks and lack the modularity required for broader applications.

4) Research Gap and Contributions of the Current Work

Although the existing body of research has contributed to the advancement of affordable mobile manipulators, several challenges remain unresolved: Limited Open-Source Interoperability: Many commercial systems such as Kinova and RB-1 use proprietary hardware and software [4], [5], limiting external modification and integration. Lack of Unified Systems: Some platforms focus solely on mobility [8], while others provide manipulation without mobility [7], leading to systems that are either static or non-interactive. Cost-Performance Tradeoff: Even modular systems like those in [10] remain costly (~$500), which may still be inaccessible for educational institutions and small enterprises.

The present study seeks to address these limitations by introducing an integrated mobile manipulator that combines both mobility and manipulation capabilities at a cost of only $150. By leveraging low-cost components such as Arduino and SG90 servos, this system achieves 4-DOF manipulation and tank-based movement in a fully wireless and modular architecture. The platform offers a compelling alternative that promotes accessibility, affordability, and expandability for robotics education and research.

II. Material and Method

A typical robotic system consists of three core components: a processor (controller), actuators, and a mechanical manipulator. According to the Robotics International of the Society of Manufacturing Engineers (RI/SME), a robot is defined as a "reprogrammable, multifunctional manipulator designed to move materials, parts, tools, or specialized devices through various programmed motions to perform a variety of tasks".

In the proposed design, each of these components is carefully selected and integrated to ensure modularity, cost-effectiveness, and operational simplicity. The following subsections outline the hardware elements, software architecture, system design, and the control flow logic of the mobile manipulator.

A. Hardware Elements

1) Arduino Uno Microcontroller

The Arduino Uno microcontroller serves as the central processing unit of the system due to its simplicity, affordability, and wide support within the robotics community. It controls both the robotic arm's servo motors and the tank-track locomotion system. Programming was conducted using the Arduino IDE.

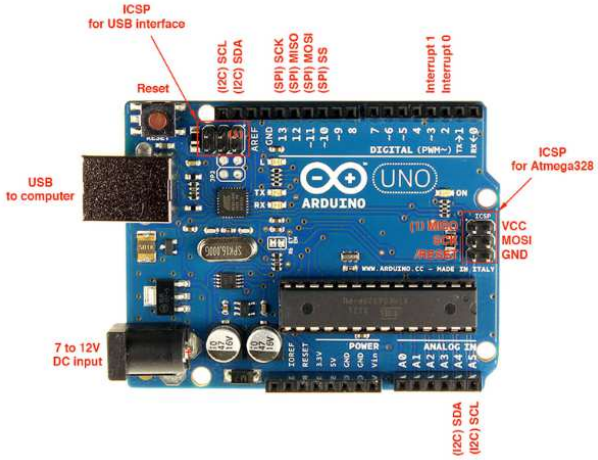


Fig. 1 Arduino Uno Microcontroller

2) Robotic Arm (Manipulator)

The manipulator comprises a 4-DOF configuration suitable for light-duty pick-and-place operations. It is composed of: Shoulder Pitch (SG90); Elbow Pitch (SG90); Wrist Rotation (SG90); Gripper Actuation (Micro Servo).

This setup enables accurate handling and positioning of objects weighing up to 500 g. The arm is mounted on a mobile base and operated wirelessly, offering flexibility and control in remote environments.

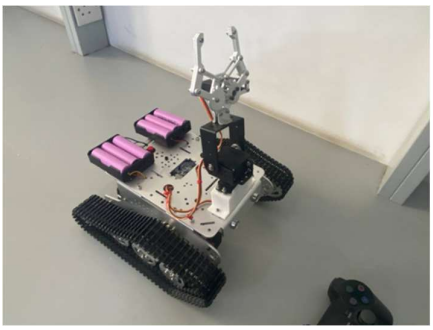


Fig. 2 Arduino Uno Microcontroller

3) Servo Motors

SG90 servo motors were selected for all joints due to their low cost, ease of integration, and suitability for position control applications. These servos operate based on PWM (Pulse Width Modulation) signals, making them ideal for precise actuation in compact, low-load systems.

4) Bluetooth Joystick Controller

The robotic platform includes a Bluetooth joystick with a 10-meter range, enabling wireless control of both the mobile base and the manipulator. This interface eliminates the need for a tethered connection, enhancing mobility and user interaction. The controller features low energy consumption and ergonomic design, making it practical for continuous use.

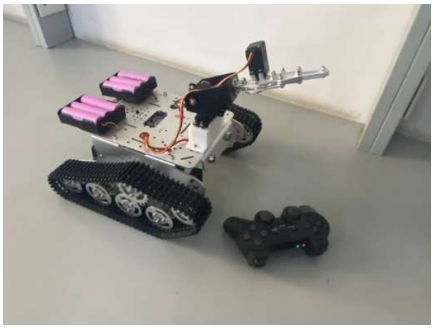


Fig. 3 Bluetooth Joystick

5) Power Source

The system is powered by a rechargeable lithium-ion battery pack (11.1V, 2.8A), providing sufficient energy for both movement and manipulation. These batteries were chosen for their high energy density and light weight, essential qualities for mobile robotic systems.

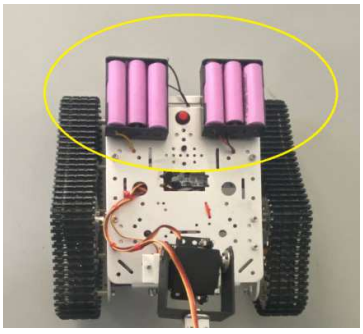


Fig. 4 Lithium-Ion Battery

B. Software Architecture

The software stack integrates the Arduino development environment with the Robot Operating System (ROS). ROS facilitates modular communication between system components—sensors, actuators, and control algorithms—allowing real-time feedback and synchronized motion control. The robot's software architecture is designed for easy reprogramming and scalability. The source code enables independent control of each servo and tank motor, with Bluetooth handling wireless communication. The control logic is documented in Appendix 1.

C. System Architecture and Design

A high-level block diagram of the system is shown in Figure 6, illustrating the interactions between hardware modules and control flow. Physical specifications of the mobile robot are summarized in Table 1.

TABLE I
PHYSICAL SPECIFICATIONS OF THE MOBILE ROBOT

Specification	Value
Height	22 cm
Length	35 cm
Width	25 cm
Weight	3.1 kg
Battery Type	11.1V, 2.8A Li-ion
Microcontroller	Arduino Uno
Movement Mechanism	3WD (Tank-track)

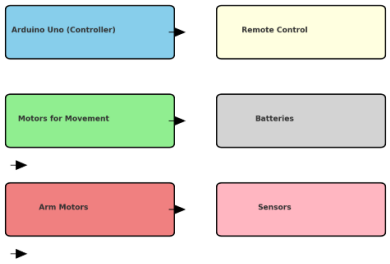


Fig. 5 System Block Diagram

D. Control Flow Logic

The operational logic of the mobile robot is described in the flowchart shown in Figure 7. The control structure involves user input via joystick, signal processing by the Arduino, and corresponding actuation of the motors for locomotion and arm movement.

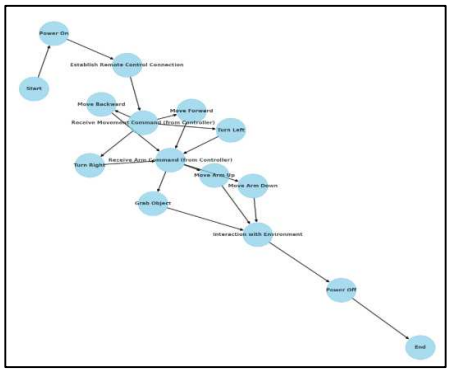


Fig. 6 Control Flowchart for Robot Operations

E. Programming and Synchronization

Custom code was developed in Arduino IDE to manage synchronized operation between the mobile base and the robotic arm. The code includes functions for reading joystick signals, processing PWM commands for servo control, and managing motor driver logic for movement. ROS modules support modularity, enabling future expansion with sensors or autonomous navigation routines. The full code is provided in Appendix 1.

F. Cost Analysis of Hardware Components

To demonstrate the affordability of the proposed system, Table 2 provides a detailed cost breakdown of all hardware components used in the mobile robot. All components were selected based on availability, performance, and cost-effectiveness.

TABLE II
COST ANALYSIS OF THE ROBOT COMPONENTS

No	Specifications	Value
1	Height	22 cm
2	Length	35 cm
3	Width	25 cm
4	Weight	3,1 kg
5	Motor Battery	11,1 V DC 2,8A
6	Microcontroller type	Arduino Uno
9	Movement type	3WD
10	Arm Robot type	3 Servo

This cost structure highlights the feasibility of building a fully functional mobile manipulator at a fraction of the cost of commercial equivalents, which often exceed $500–$5000. By using widely available and open-source components, the proposed design is particularly suitable for use in educational labs, prototyping centers, and budget-constrained research environments.

III. RESULT AND DISCUSSION

This section presents the experimental evaluation of the mobile robot with an integrated arm manipulator. The assessment focused on key performance metrics such as navigation stability, manipulation success rate, and overall cost-effectiveness.

A. Experimental Setup

All experiments were conducted in a controlled indoor environment measuring 4 meters by 4 meters, free of obstacles. A group of ten high school students participated in supervised testing sessions, allowing for hands-on interaction with the system under the guidance of an instructor. A dedicated pause button was configured on the Bluetooth controller to ensure immediate system shutdown in case of unexpected behavior, enhancing user safety during testing.

B. Navigation Performance

The tank-track mobility system allowed the robot to operate reliably across both smooth and moderately uneven terrains. Tests demonstrated stable navigation on flat surfaces and over small obstacles, as well as successful traversal of inclined planes up to 15°. The locomotion mechanism maintained directional stability and responded well to joystick commands during real-time control.

C. Manipulation Accuracy

The 4-DOF arm achieved consistent performance in pick-and-place operations involving lightweight objects (200–500 g). Across 50 trials, the system achieved a 93% success rate, confirming its suitability for light-duty tasks such as object retrieval and sorting. The manipulator performed reliably in repeated operations and maintained structural stability throughout the experiments.

D. Cost Evaluation

A detailed breakdown of the component costs was previously presented in Table 2. The final total system cost amounted to approximately $150 USD, substantially lower than comparable commercial platforms. This validates the economic feasibility of the design for resource-limited environments.

E. Real-World Use Cases

To evaluate real-world applicability, the system was tested in scenarios such as: Retrieving objects from cluttered or constrained environments; Navigating around obstacles while carrying a lightweight item.

The robot demonstrated effective task execution in either scenarios without mechanical or control failures.

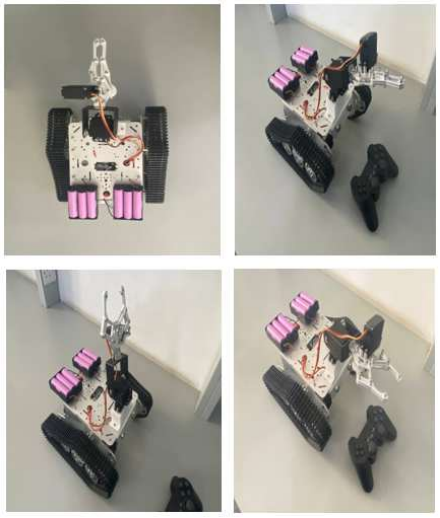


Fig. 7 The robot performing real-world object retrieval and mobility tasks during testing

F. Comparison with Commercial Systems

A comparative analysis was conducted to benchmark the developed robot against existing commercial mobile manipulators. The results are summarized in Table 3.

TABLE III
COMPARISON OF THE DEVELOPED ROBOT WITH COMMERCIAL SYSTEMS

Component	**Cost (USD)**
Microcontroller	15
DC Motors (2 units)	20
Servo Motors (3 units)	50
Sensors	25
Power Supply	30
Miscellaneous	10
Total Cost	150

This comparison highlights that although commercial systems offer higher payloads and greater mechanical robustness, the proposed robot delivers a compelling balance of functionality, modularity, and affordability. It is particularly suitable for applications where cost constraints outweigh the need for heavy-duty performance.

The experimental results demonstrate that the proposed mobile manipulator is effective for light-duty tasks in structured environments. With a 93% success rate in pick-and-place operations and reliable terrain navigation, the system confirms its suitability for educational and small-scale automation contexts. Compared to previous low-cost systems that focus solely on mobility or static arms [7], [8], this design offers an integrated solution combining both features in a modular, wireless platform. The use of open-source hardware (Arduino, ROS) enhances adaptability and allows for easy customization or future upgrades. Despite limitations in payload capacity and terrain adaptability—primarily due to the use of hobby-grade servo motors—the platform delivers significant functionality at a total cost of just $150. This makes it an attractive option for schools, prototyping labs, and budget-constrained institutions.

G. Challenges and Future Researches

1) Limitations

While the system demonstrated promising results, certain limitations were identified during testing: Payload Capacity: The use of SG90 hobby-grade servo motors restricts the arm's ability to lift heavier objects beyond 500 g; Terrain Adaptability: Navigation performance declines on highly uneven or loose surfaces due to the basic tank-track configuration without suspension or terrain feedback; Power Efficiency: The system's energy consumption limits its runtime, especially when both the arm and base are operated simultaneously under load.

Despite these constraints, the robot successfully fulfills its design objectives for low-cost, remotely operated manipulation in structured or semi-structured environments.

2) Future Enhancements

To further improve system performance and expand its applicability, future work will focus on: Enhanced Payload Handling: Integrating higher-torque servo motors or lightweight mechanical enhancements to improve the arm's lifting capacity. Improved Mobility: Adding active suspension or sensor-assisted navigation to better handle unstructured terrain. Autonomous Capabilities: Incorporating computer vision, object recognition, and path planning algorithms to enable autonomous manipulation and obstacle avoidance. Power Management: Optimizing energy consumption through intelligent power distribution and monitoring systems to extend operational runtime.

These developments will help transition the platform from a remote-controlled system to a semi-autonomous robotic solution, suitable for broader real-world applications in education, research, and light industrial tasks.

IV. CONCLUSION

This paper presented the design and implementation of a low-cost mobile robot equipped with a 4-DOF arm manipulator for remote operation in structured environments. Built entirely from affordable and open-source components—including Arduino, SG90 servo motors, and ROS middleware—the system achieved reliable navigation and object manipulation while maintaining a total hardware cost of approximately $150.

The platform demonstrated a 93% success rate in light-duty pick-and-place tasks and stable movement over inclined and semi-rough terrain. Its modular design facilitates ease of assembly, maintenance, and future upgrades, making it especially suitable for educational, research, and small-scale automation applications. While the robot has certain limitations—such as reduced payload capacity and limited terrain adaptability—these trade-offs are acceptable within its intended use case. Future enhancements will target autonomy, mobility, and power efficiency to extend the system's applicability and performance. This work contributes a replicable, low-cost mobile manipulator that can help bridge the accessibility gap in practical robotics, offering a valuable tool for learning, experimentation, and prototyping in environments with limited resources.

ACKNOWLEDGMENT

The authors gratefully acknowledge the support provided by Al-Ayen Iraqi University (AUIQ) through its laboratory facilities and technical resources. Special thanks are extended to the Artificial Intelligence Department, College of Engineering, for their invaluable assistance in system prototyping, testing, and evaluation.

REFERENCES

[1] C. Gentile et al., "Manipulation tasks in hazardous environments using a teleoperated robot: A case study at CERN," *Sci. Rep.*, vol. 13, no. 1, 2023, Art. no. 12456, doi: 10.1038/s41598-023-39322-z.

[2] N. Tom, D. K. Badam, and V. K. Singore, "Design and implementation of mobile robotic manipulator for welding using PLC," *Sensors Transducers*, vol. 266, no. 3, pp. 60–68, May 2024.

[3] N. Ghodsian, K. Benfriha, A. Olabi, and V. Gopinath, "Mobile manipulators in Industry 4.0: A review," *Machines*, vol. 11, no. 12, Dec. 2023, Art. no. 1079, doi: 10.3390/machines11121079.

[4] B. Vaisi, "A review of optimization models and applications in robotic manufacturing systems: Industry 4.0 and beyond," *Decision Anal. J.*, vol. 2, Feb. 2022, Art. no. 100031, doi: 10.1016/j.dajour.2022.100031.

[5] L. Zhang and R. Schmidt, "Robotics and automation in Industry 4.0: Enhancing efficiency, flexibility, and scalability," *Int. J. Adv. Manuf. Technol.*, vol. 118, no. 5-6, pp. 2125–2148, 2023, doi:10.1007/s00170-023-11078-w.

[6] W. Montalvo, J. Escobar-Naranjo, C. A. Garcia, and M. V. Garcia, "Low-cost automation for gravity compensation of robotic arm," *Appl. Sci.*, vol. 10, no. 11, 2020, Art. no. 3823, doi: 10.3390/app10113823.

[7] C. Sun et al., "Design and analysis for a multifunctional rescue robot with four-bar wheel-legged structure," *Adv. Mech. Eng.*, vol. 10, no. 6, pp. 1–12, 2018, doi: 10.1177/1687814017747399.

[8] C. S. Dobson et al., "Antigen identification and high-throughput interaction mapping by reprogramming viral entry," *Nat. Methods*, vol. 19, pp. 449–460, 2022, doi: 10.1038/s41592-022-01436-z.

[9] J. Vega and V. Pérez, "G-ARM: An open-source and low-cost robotic arm integrated with ROS2 for educational purposes," *Multimed. Tools Appl.*, 2025, doi: 10.1007/s11042-025-20748-8.

[10] J. Silva et al., "Open source 3D-printed robotic arm for ROS2-based robotics education," *IEEE Robot. Autom. Lett.*, vol. 9, no. 2, pp. 1234–1241, 2024, doi: 10.1109/LRA.2024.3351234.

[11] L. Bruzzone, S. E. Nodehi, and P. Fanghella, "Tracked locomotion systems for ground mobile robots: A review," *Machines*, vol. 10, no. 8, 2022, Art. no. 648, doi: 10.3390/machines10080648.

[12] M. Jdeed, M. Schranz, and W. Elmenreich, "A study using the low-cost swarm robotics platform Spiderino in education," *Comput. Educ. Open*, vol. 1, 2020, Art. no. 100017, doi: 10.1016/j.caeo.2020.100017.

[13] H. Lyu et al., "Impacts of wireless on robot control: The network hardware-in-the-loop simulation framework and real-life comparisons," *IEEE Trans. Ind. Informat.*, vol. 19, no. 9, pp. 9255–9265, Sep. 2023, doi: 10.1109/TII.2022.3227639.
[14] R. M. Murray, Z. Li, and S. S. Sastry, *A Mathematical Introduction to Robotic Manipulation*. Boca Raton, FL, USA: CRC Press, 2017, doi: 10.1201/9781315136370.
[15] C. Garrett et al., "Integrated task and motion planning," *arXiv*, 2020, doi: 10.48550/arXiv.2010.01083.
[16] M. Kanamura et al., "Development of a basic educational kit for robotic system with deep neural networks," *Sensors*, vol. 21, no. 11, 2021, Art. no. 3804, doi: 10.3390/s21113804.
[17] K. Chachane, S. Ohol, and S. Chiwande, "Industrial robot performance analysis using low-cost set-up," *IOP Conf. Ser.: Mater. Sci. Eng.*, vol. 1012, no. 1, 2021, Art. no. 012010, doi: 10.1088/1757-899X/1012/1/012010.

APPENDIX 1

```
#include <PS2X_lib.h>
#include <Servo.h>

#define MotRF 2
#define MotRB 3
#define MotLF 4
#define MotLB 5
#define LED 9


#define SERVO1_PIN 6 // Controlled by PSS_RY
#define SERVO2_PIN 7 // Controlled by PSS_LY
#define SERVO3_PIN 8 // Controlled by PSS_RX

bool led5State = false; // LED state (ON/OFF)
bool lastBlueButtonState = false; // Track the previous state of PSB_BLUE

PS2X ps2x; // Create PS2 Controller Class
Servo servo1, servo2, servo3; // Create Servo objects

// Variables for PS2 controller
int error = 0;
byte type = 0;
byte vibrate = 0;

// Servo initial positions
int servo1Position = 90; // Middle position (0-180 degrees)
int servo2Position = 90;
int servo3Position = 90;

void setup() {
  // Configure LED pins
  pinMode(LED1, OUTPUT);
  pinMode(LED2, OUTPUT);
  pinMode(LED3, OUTPUT);
  pinMode(LED4, OUTPUT);
 pinMode(LED5, OUTPUT);

  digitalWrite(LED1, LOW);
  digitalWrite(LED2, LOW);
  digitalWrite(LED3, LOW);
  digitalWrite(LED4, LOW);
  digitalWrite(LED5, LOW);

  // Attach the servos to their pins
  servo1.attach(SERVO1_PIN);
  servo2.attach(SERVO2_PIN);
  servo3.attach(SERVO3_PIN);

  // Set initial servo positions
  servo1.write(servo1Position);
  servo2.write(servo2Position);
  servo3.write(servo3Position);

  // Start serial communication
  Serial.begin(57600);

  // Initialize PS2 controller
  error = ps2x.config_gamepad(13, 11, 10, 12, true, true); // (clock, command, attention, data, pressures?, rumble?)
  if (error == 0) {
    Serial.println("Found Controller, configured successfully");
  } else if (error == 1) {
    Serial.println("No controller found, check wiring or reset the Arduino");
  } else if (error == 2) {
    Serial.println("Controller found but not accepting commands");
  } else if (error == 3) {
    Serial.println("Controller refusing to enter Pressures mode, may not support it.");
  }

  // Check controller type
  type = ps2x.readType();
  switch (type) {
    case 0:
      Serial.println("Unknown Controller type");
      break;
    case 1:
      Serial.println("DualShock Controller Found");
      break;
    case 2:
      Serial.println("GuitarHero Controller Found");
      break;
  }
}

void loop() {
  if (error == 1) // Skip loop if no controller found
    return;

  ps2x.read_gamepad(false, vibrate); // Read PS2 input

  // Control servo1 motor with right joystick vertical (PSS_RY)
  int joystickRY = ps2x.Analog(PSS_RY); // Read the Y-axis value of the right joystick
  if (joystickRY > 130) { // Move servo clockwise
    if (servo1Position < 180) {
      servo1Position += 2;
      servo1.write(servo1Position);
      delay(20);
    }
  } else if (joystickRY < 120) { // Move servo counter-clockwise
    if (servo1Position > 0) {
      servo1Position -= 2;
      servo1.write(servo1Position);
      delay(20);
    }
  }

  // Control servo2 motor with left joystick vertical (PSS_LY)
  int joystickLY = ps2x.Analog(PSS_LY); // Read the Y-axis value of the left joystick
  if (joystickLY > 130) { // Move servo clockwise
    if (servo2Position < 180) {
      servo2Position += 2;
      servo2.write(servo2Position);
     delay(20);
    }
  } else if (joystickLY < 120) { // Move servo counter-clockwise
    if (servo2Position > 0) {
      servo2Position -= 2;
      servo2.write(servo2Position);
      delay(20);
    }
  }

  // Control servo3 motor with right joystick horizontal (PSS_RX)
  int joystickRX = ps2x.Analog(PSS_RX); // Read the X-axis value of the right joystick
  if (joystickRX > 130) { // Move servo clockwise
    if (servo3Position < 180) {
      servo3Position += 2;
      servo3.write(servo3Position);
```

```
    delay(20);
   }
 } else if (joystickRX < 120) { // Move servo counter-clockwise
   if (servo3Position > 0) {
     servo3Position -= 2;
     servo3.write(servo3Position);
     delay(20);
   }
 }


 if (ps2x.Button(PSB_R1)) {
   digitalWrite(MotRF, HIGH);
   Serial.println("MotRF ON");
 } else {
   digitalWrite(MotRF, LOW);
 }

 if (ps2x.Button(PSB_R2)) {
   digitalWrite(MotRL, HIGH);
   Serial.println("MotRL ON");
 } else {
   digitalWrite(MotRL, LOW);
 }

 if (ps2x.Button(PSB_L1)) {
   digitalWrite(MotLF, HIGH);
   Serial.println("MotLF ON");
 } else {
   digitalWrite(MotLF, LOW);
 }

 if (ps2x.Button(PSB_L2)) {
   digitalWrite(MotLB, HIGH);
   Serial.println("MotLB ON");
 } else {
   digitalWrite(MotLB, LOW);
 }

 // Toggle LED using PSB_BLUE button
 if (ps2x.Button(PSB_BLUE)) { // Check if the PSB_BLUE button is
pressed
    if (!lastBlueButtonState) { // Only execute on button press (not hold)
      led5State = !led5State; // Toggle the LED state
      digitalWrite(LED, led5State ? HIGH : LOW); // Update the LED
state
      Serial.print("LED is now ");
      Serial.println(ledState ? "ON" : "OFF");
    }
   lastBlueButtonState = true; // Update the button state
 } else {
   lastBlueButtonState = false; // Reset the button state when released
 }


 delay(50); // General delay for stability
}
```