# FlexiArm: A 6-DOF App-Controlled Industrial Manipulator

Course Code: ETE 320, Course Title: Microprocessor and Microcontroller Sessional

Mueez Mejbah (2108010), Belayet Hossain Saimun (2108032), Sammaul Islam Siam (2108040),
Mohammad Mesbahul Alam (2108043), Saklain Shuvo (2108044), Md. Hasibul Islam Jihad (2108048)

*Department of Electronics and Telecommunication Engineering*
*Chittagong University of Engineering and Technology (CUET)*
Chattogram, Bangladesh

***Abstract*—This paper presents FlexiArm, a 6-degree-of-freedom robotic manipulator controlled via Bluetooth communication. The system integrates Arduino Uno microcontroller, MG996R servo motors, PCA9685 PWM driver, and HC-05 Bluetooth module with a custom Android application. The implementation demonstrates wireless robotic control with pose management capabilities, allowing storage and automated playback of up to 10 programmed positions. Experimental validation confirms successful pick-and-place operations, smooth trajectory execution, and reliable wireless operation within a 15-meter range. The system achieved position accuracy of ±2° with payload capacity of 200 grams, providing an effective platform for educational robotics and small-scale automation applications.**

***Index Terms*—Robotic manipulator, Arduino Uno, Bluetooth control, PCA9685, servo motors, pose management, wireless control**

## I. Introduction

Robotic manipulators have become fundamental components in modern automation systems, with applications spanning manufacturing, assembly operations, and educational research [1]. The 6-degree-of-freedom (6-DOF) configuration represents a versatile architecture that enables complete positional and rotational control in three-dimensional space [2].

The development of FlexiArm was motivated by the need for an accessible educational platform that demonstrates practical concepts in robotic control, wireless communication, and human-machine interfaces. Existing commercial robotic arms typically require significant financial investment, while simpler educational kits often lack the sophistication needed to demonstrate real-world automation principles. This project bridges that gap by implementing industrial-grade features using readily available components.

The primary objectives of this work include: (1) implementation of a functional 6-DOF manipulator using servo-based actuation, (2) wireless control via Bluetooth communication protocol, (3) development of pose memory and sequence playback functionality, (4) integration of PCA9685 servo driver for modular PWM generation, and (5) creation of an intuitive Android application interface for system operation. The target cost constraint of under $150 ensures accessibility for educational institutions while maintaining sufficient capability for demonstrating fundamental robotics principles.

## II. Methodology

### A. System Architecture

The system employs a hierarchical control architecture comprising five functional layers: user interface (Android application), communication (HC-05 Bluetooth module), processing (Arduino Uno microcontroller), actuation control (PCA9685 PWM driver), and mechanical execution (servo motors and linkages). This modular design enables independent development and testing of each subsystem while maintaining clear interfaces between components.

Figure 1 presents the complete system flow diagram illustrating the data flow from the MIT App Inventor interface through Bluetooth communication to the Arduino Uno/Mega, which processes commands and controls the robotic arm's six servo motors via the PCA9685 PWM driver. The diagram shows the bidirectional communication for connection status and real-time position feedback.

Figure 2 illustrates the completed robotic manipulator showing the six degrees of freedom: base rotation (6), shoulder joint (5), elbow joint (4), wrist pitch (3), wrist roll (2), and gripper mechanism (1).

### B. Mechanical Design and Assembly

The mechanical structure consists of six serially-connected joints forming a kinematic chain from base to end-effector. Each joint is actuated by a dedicated MG996R servo motor providing 180° range of motion. The design prioritizes lightweight construction to minimize torque requirements while maintaining structural rigidity during operation.

Assembly proceeded systematically: (1) base platform construction and mounting of rotation servo, (2) shoulder and elbow linkage assembly ensuring planar alignment, (3) three-axis wrist mechanism integration for end-effector orientation control, (4) gripper mechanism attachment with servo-driven jaw actuation, (5) secure servo mounting with mechanical coupling to respective joints, and (6) cable routing through

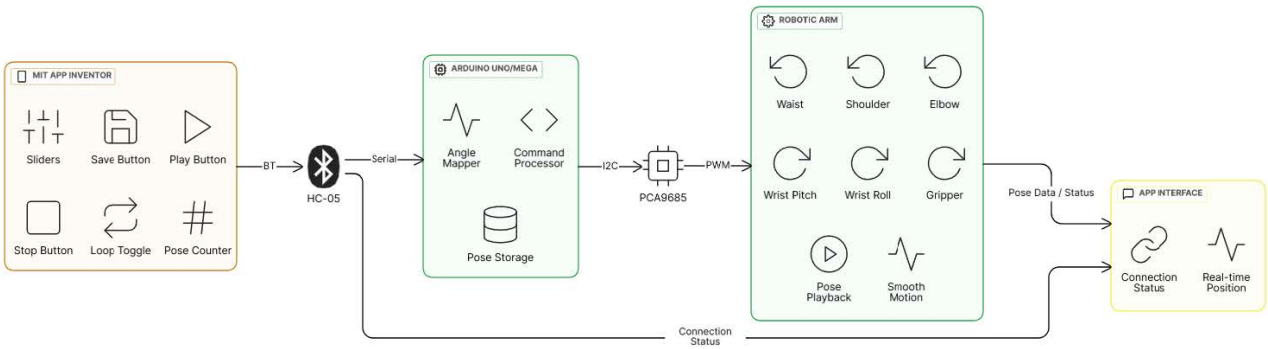


Fig. 1. System flow from Android app to robotic arm via Bluetooth and Arduino.

arm structure with appropriate strain relief to prevent wire fatigue during repeated motion cycles.

### C. Electronic Hardware Implementation

*1) Component Selection and Specifications:* Table I presents the key components utilized in this implementation. The Arduino Uno was selected based on its ATmega328P microcontroller (16 MHz, 32KB Flash memory) providing sufficient computational resources for real-time servo control and Bluetooth communication [3]. Despite having fewer I/O pins compared to larger boards, the Uno proved adequate when paired with the PCA9685 driver which requires only two pins (A4/SDA, A5/SCL) for I2C communication to control all six servos.

TABLE I
SYSTEM COMPONENT SPECIFICATIONS

| Component | Specification | Qty |
|---|---|---|
| Arduino Uno | ATmega328P, 16 MHz | 1 |
| PCA9685 Driver | 16-channel PWM, $I^2C$ | 1 |
| MG996R Servo | High-torque servo, ∼11 kg-cm @ 6 V | 3 |
| SG90 Servo | Micro servo, ∼1.8 kg-cm @ 5 V | 3 |
| HC-05 Bluetooth | Wireless control, 10–20 m range | 1 |
| Power Supply | 5–6 V, 2 A | 1 |

The MG996R servos offer 11 kg-cm stall torque at 6V with metal gear construction for durability [4]. The HC-05 Bluetooth module operates at 2.4 GHz with configurable baud rates supporting reliable command transmission [8]. The PCA9685 provides 16 independent PWM outputs with 12-bit resolution, significantly reducing Arduino processing load by handling all timing-critical PWM generation in hardware [7].

*2) Circuit Design and Wiring:* Figure 3 presents the complete circuit diagram. The electronic integration connects all six servo motors to the PCA9685 servo driver via its PWM output channels. The PCA9685 communicates with Arduino Uno through I2C protocol using pins A4 (SDA) and A5 (SCL), substantially reducing pin requirements compared to direct servo connections.

A critical design consideration was power management. Servos can draw up to 2.5A each during peak operation, necessitating a dedicated 5V 10A external power supply. The Arduino receives power via USB connection. Decoupling capacitors (470µF) were installed in parallel with each servo to filter voltage transients and prevent brown-out conditions during simultaneous servo movements. The HC-05 Bluetooth module connects to the hardware serial port (pins 0 and 1, RX/TX) enabling bidirectional communication with the Android application [8], [9].

### D. Software Development

*1) Arduino Firmware Implementation:* The Arduino firmware implements several key functional modules. The Adafruit PCA9685 library [7] handles I2C communication and PWM signal generation. Command parsing processes incoming Bluetooth serial data, interpreting single-digit commands (0-6) as slider position updates and character commands ('S', 'P', 'R', etc.) as pose management operations.

Motion planning incorporates trajectory smoothing through incremental position updates. Rather than commanding immediate jumps to target angles, the algorithm advances servo positions by 1° increments with 10ms delays between steps [2]. This approach prevents jerky movements, reduces mechanical stress, and improves motion quality.

Pose management utilizes a two-dimensional integer array storing up to 10 poses, each containing six servo positions. The save function records current positions to the next available array slot. The playback function iterates through all saved poses sequentially, moving servos to each stored configuration. Loop mode enables continuous sequence repetition until explicitly stopped. Memory constraints of the Arduino Uno (2KB SRAM) limit pose storage to 10 positions, though this proved sufficient for demonstration purposes.

*2) Android Application Development:* The Android application was developed using MIT App Inventor [6], a visual programming environment enabling rapid prototyping. Figure 4 shows the user interface comprising six vertical sliders for individual servo control (0-180°), connection status indicator, and pose management controls.


Fig. 2. FlexiArm robotic manipulator showing six independent joints: (1) Gripper, (2) Wrist roll, (3) Wrist pitch, (4) Elbow, (5) Shoulder, (6) Base rotation

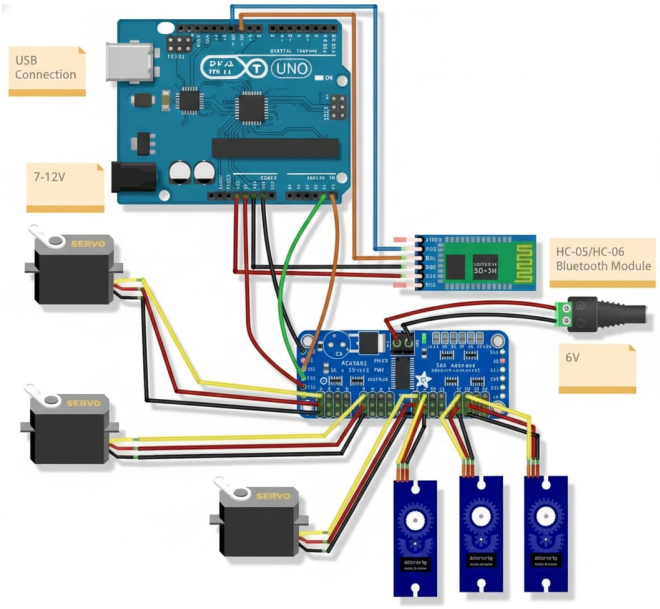


Fig. 3. System circuit diagram showing Arduino Uno, PCA9685 servo driver, HC-05 Bluetooth module, and servo motor interconnections

The application implements Bluetooth device discovery and pairing using MIT App Inventor's BluetoothClient component [9]. When a user adjusts a slider, the application transmits the servo number concatenated with the position value via Bluetooth. Pose management buttons send single-character commands: 'S' for save, 'P' for play, 'R' for reset. The loop toggle transmits "LoopON" or "LoopOFF" strings to enable continuous sequence playback.

Event-driven programming handles slider position changes, button clicks, and Bluetooth connection status updates. Visual feedback includes color-coded connection status (green for connected, red for disconnected) and a pose counter displaying the number of saved positions.

*3) MIT App Inventor Block Programming Implementation:* The application logic was implemented using MIT App Inventor's visual block programming interface [6]. The following figures demonstrate the key functional components.

Figure 5 illustrates the initial screen navigation block that opens the main control interface (Screen2) when the "about" button is clicked, providing seamless transition to the robot control panel.

Figure 6 demonstrates the pose management system implementation. The SaveButton stores the current configuration by transmitting "S\n" command and incrementing the pose counter. The PlayButton initiates sequence playback with "P\n" command, while StopButton sends "Stn\n" to halt execution. The ResetButton clears all saved poses by transmitting "R\n" and resetting the counter to zero.

Figure 7 illustrates the loop functionality implementation. When the LoopSwitch is toggled, the application transmits either "LoopON\n" or "LoopOFF\n" commands to enable or disable continuous pose sequence repetition.

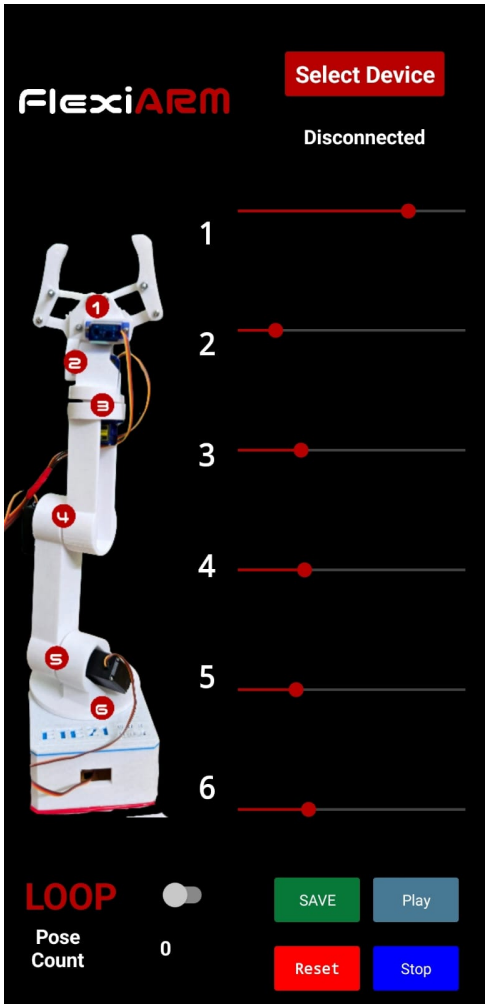


Fig. 4. Android application interface showing servo control sliders, Bluetooth connection status, and pose management panel


Fig. 5. Screen navigation block for main control interface access

Figure 8 shows the Bluetooth connection logic implemented using MIT App Inventor's BluetoothClient component [9]. The ListPicker1 component displays available Bluetooth devices from the HC-05 module's AddressesAndNames. Upon selection, the application attempts connection and updates the status label with color-coded feedback (green for "Connected", red for "Disconnected").

Figure 9 presents the implementation of six independent slider controls for each servo motor (Slider1 through Slider6). Each slider's PositionChanged event triggers Bluetooth transmission of the joint number concatenated with the slider position value, enabling real-time servo angle control from 0° to 180°. This provides intuitive manual control of the base, shoulder, elbow, wrist pitch, wrist roll, and gripper joints.

The visual block programming approach provides several advantages including accessible logic representation for non-programmers, rapid prototyping capabilities, built-in Bluetooth communication components [9], event-driven architecture for responsive interaction, and straightforward modification for feature expansion.

## III. Results and Observations

### A. Functional Validation

Comprehensive testing validated system functionality across multiple operational scenarios. Individual servo testing confirmed full 0-180° range of motion with consistent response to commanded positions. Position accuracy was measured at ±2°, limited by the inherent resolution of the servo motors rather than control system limitations.

Bluetooth communication testing demonstrated reliable connection establishment and command transmission. Maximum operational range reached approximately 15 meters in open space, reducing to 8-10 meters with wall penetration. Command latency measured between 200-500ms from slider adjustment to observable servo motion, acceptable for manual control applications. No packet loss was observed during normal operation, though connection stability degraded beyond the maximum range threshold.

Integrated system testing evaluated coordinated multi-joint movements. The arm successfully executed pick-and-place operations with objects up to 200 grams mass. Pose memory functionality reliably stored and recalled 10 distinct positions. Sequence playback demonstrated consistent repeatability over multiple cycles. Continuous operation testing ran automated sequences for extended periods (30+ minutes) without observable degradation in performance or position accuracy.

### B. Performance Metrics

Quantitative performance evaluation yielded the following metrics:

- **Workspace volume:** Approximately 0.15 m³ reachable envelope
- **Position accuracy:** ±2° angular resolution per joint
- **Command latency:** 200-500ms typical response time
- **Payload capacity:** 200g at maximum extension
- **Wireless range:** 15m line-of-sight, 8-10m with obstacles
- **Pose storage:** 10 positions successfully saved and recalled
- **Motion quality:** Smooth trajectory execution at 1°/10ms increment rate
- **Power consumption:** 3-4A total during coordinated movements

The incremental motion control strategy (1° steps with 10ms delays) effectively eliminated jerky movements and produced visually smooth trajectories. This approach proved essential for maintaining mechanical integrity during extended operation.

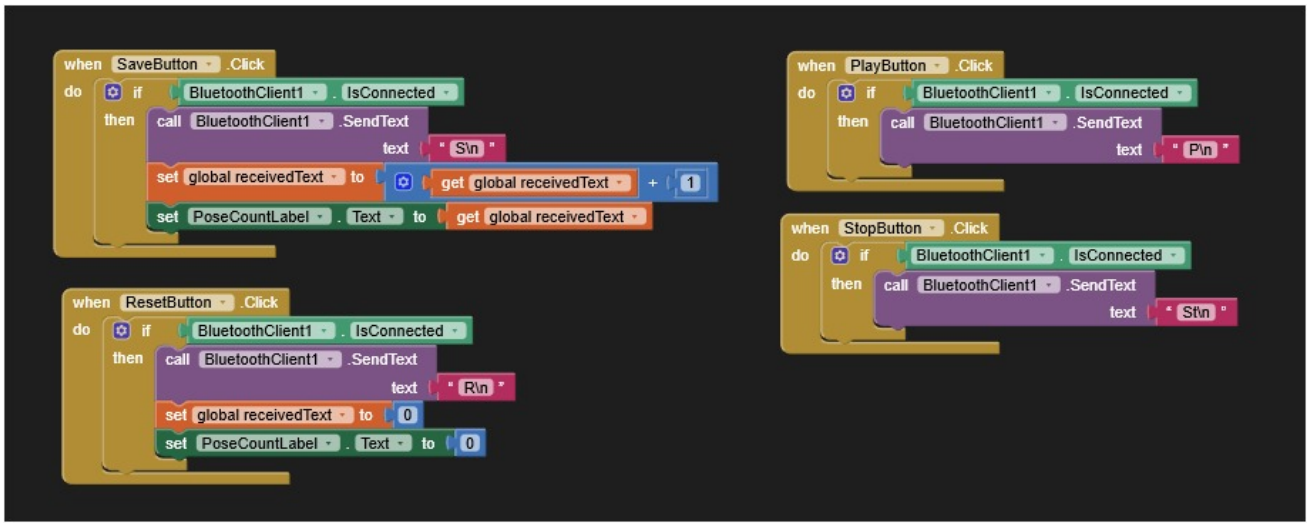


Fig. 6. Pose management blocks implementing Save, Play, Stop, and Reset functionality

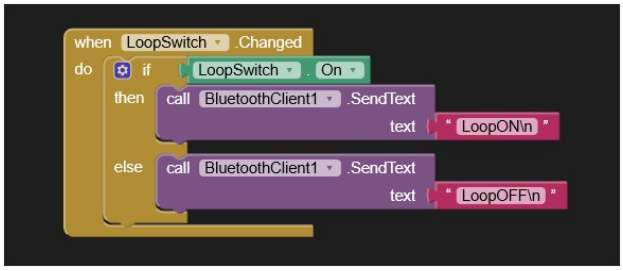


Fig. 7. Loop control block enabling continuous sequence repetition

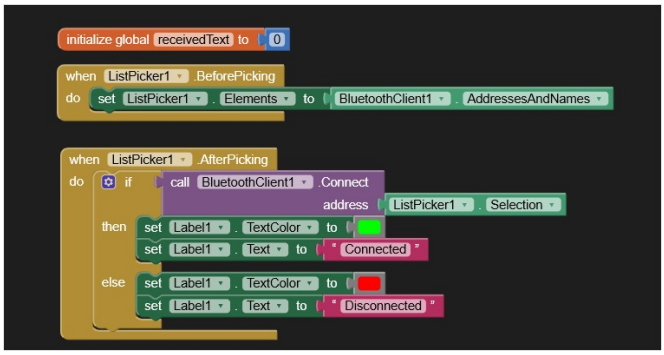


Fig. 8. Bluetooth connection blocks for device pairing and connection management

### C. Practical Demonstrations

Several demonstration tasks validated real-world applicability. Pick-and-place operations successfully transferred small objects (batteries, small boxes) between designated locations. Drawing tasks with an attached pen traced simple 2D geometric patterns, demonstrating position control accuracy. Preset position sequences transitioned smoothly between programmed poses, validating the utility of pose memory for repetitive automation tasks.

The wireless control capability proved particularly valuable during testing, enabling safe observation of arm behavior from a distance during initial calibration and potentially problematic movement sequences.

### D. Challenges Encountered and Solutions

Several technical challenges emerged during development:

**Voltage stability:** Initial testing revealed Arduino resets during simultaneous servo movements due to voltage drops on shared power rails. Implementation of separate power supplies (USB for Arduino, 5V 10A for servos) with decoupling capacitors completely resolved this issue.

**Bluetooth connectivity:** Intermittent connection drops occurred during early testing. Improved antenna placement and implementation of connection status monitoring with automatic reconnection logic in both Arduino and Android code substantially improved reliability.

**Mechanical binding:** Wrist joints occasionally bound when fully extended. Adjustment of servo mounting tolerances and application of appropriate lubrication eliminated this problem.

**Cable management:** Wire routing through rotating joints initially caused tangling and strain. Systematic cable routing with strain relief loops and zip-tie anchoring provided a permanent solution.

These challenges and their solutions provided valuable insights into practical robotics system integration beyond theoretical design considerations.

## IV. Conclusion

This work successfully implemented a functional 6-DOF robotic manipulator with wireless Bluetooth control and advanced pose management capabilities. The integration of Arduino Uno microcontroller, PCA9685 servo driver, MG996R servo motors, HC-05 Bluetooth module, and custom Android application resulted in a cohesive system capable of performing coordinated manipulation tasks.

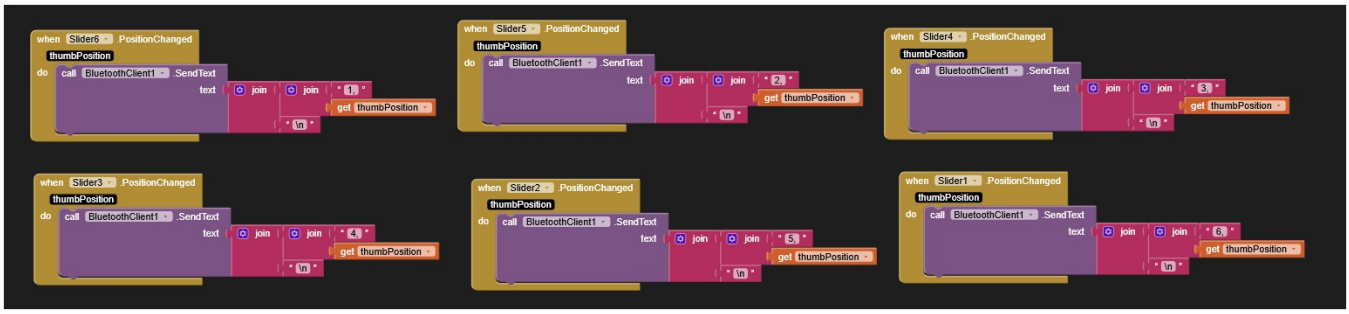


Fig. 9. Servo slider control blocks for individual joint position control of all six servos

The project validated several key technical approaches. The PCA9685 servo driver proved essential for reducing pin requirements and offloading PWM generation from the microcontroller [7]. The incremental motion control algorithm successfully produced smooth trajectories while preventing mechanical stress [2]. The pose memory implementation demonstrated practical utility for repetitive automation sequences despite memory constraints of the Arduino Uno platform [3].

Experimental results confirmed position accuracy of ±2°, payload capacity of 200g, and reliable wireless operation within 15-meter range. The system successfully executed pick-and-place operations, stored and recalled 10 programmed poses, and maintained stable operation during extended testing periods.

Limitations include restricted payload capacity, absence of position feedback (open-loop control), lack of collision detection, and manual tuning requirements. Future enhancements could address these through encoder integration for closed-loop control [5], force sensing capabilities, improved mechanical design with precision components [1], EEPROM storage for persistent pose memory beyond RAM limitations, and vision system integration for autonomous operation.

If performance requirements exceed current capabilities, several mitigation strategies are available: upgrading to higher-torque actuators for increased payload, implementing stepper motors with encoders for improved precision [2], expanding to Arduino Mega for additional memory and I/O capacity, or integrating external position sensing for closed-loop control [5].

The project demonstrates that sophisticated robotic systems with industrial-grade features can be constructed using accessible components and open-source platforms [3], [6], providing effective educational value across mechanical design [1], [2], electronic integration [7], embedded programming, and mobile application development domains. FlexiArm serves as a practical bridge between simplified educational kits and professional industrial manipulators, offering hands-on experience with automation concepts directly applicable to real-world systems.

## REFERENCES


[1] S. B. Niku, *Introduction to Robotics: Analysis, Control, Applications*, 3rd ed. Hoboken, NJ: Wiley, 2020.

[2] J. J. Craig, *Introduction to Robotics: Mechanics and Control*, 4th ed. Pearson, 2018.

[3] Arduino, "Arduino Uno Rev3," https://store.arduino.cc/products/arduino-uno-rev3, accessed Dec. 2025.

[4] TowerPro, "MG996R High Torque Metal Gear Servo Motor," Datasheet, 2019.

[5] M. W. Spong, S. Hutchinson, and M. Vidyasagar, *Robot Modeling and Control*, 2nd ed. Wiley, 2020.

[6] MIT App Inventor, "MIT App Inventor Documentation," http://appinventor.mit.edu/explore/, accessed Dec. 2024.

[7] Adafruit Industries, "Adafruit 16-Channel 12-bit PWM/Servo Driver - I2C interface - PCA9685," https://www.adafruit.com/product/815, accessed Dec. 2024.

[8] HC-05 Bluetooth Module, "HC-05 Serial Bluetooth Products User Manual," Datasheet, 2011.

[9] MIT Media Lab, "MIT App Inventor Bluetooth Communication," http://ai2.appinventor.mit.edu/reference/components/connectivity.html#BluetoothClient, accessed Dec. 2025.


## TEAM CONTRIBUTIONS

TABLE II
INDIVIDUAL TEAM MEMBER CONTRIBUTIONS

| Student ID | Contribution |
|---|---|
| 2108010 | Conceptualization and project planning; Component sourcing and procurement; Materials preparation and inventory management |
| 2108032 | Methodology development; Mechanical assembly and fabrication; Hand gripper design, construction and integration |
| 2108040 | Resources management; Component collection and organization; Hardware assembly support |
| 2108043 | Software development and validation; Arduino firmware implementation; System integration and comprehensive testing |
| 2108044 | Resources coordination; Component procurement and organization; Assembly assistance |
| 2108048 | Documentation - Original draft preparation; Technical writing and review; Report editing and final manuscript preparation |