**1. Project Title**

**Remote Stepper Motor Controller System Using ESP32, IR Remote, and OLED Display**

**2. Abstract**

The “Remote Stepper Motor Controller System” is an embedded project designed for precise and user-friendly control of a 28BYJ-48 stepper motor using an ESP32 microcontroller, an IR remote, and a 0.96-inch OLED display. This system allows remote directional control, adjustable speed settings, and cycle-based movement using an intuitive IR interface, all displayed in real time on the OLED screen. The aim of the project is to demonstrate interactive hardware control using common microcontroller components while making room for future upgrades like Wi-Fi control, limit switches, and partial angle movement.

**3. Introduction**

Stepper motors are widely used in CNC machines, robotics, and automation due to their precision and repeatability. This project focuses on controlling a 28BYJ-48 stepper motor using an ESP32 board, enabling various functions via an IR remote. An OLED screen provides real-time feedback, making the system suitable for applications where visual cues are critical. The project is ideal for beginners and intermediate enthusiasts exploring IoT, embedded systems, and mechatronics.

**4. Objectives**

* To control the direction and speed of a stepper motor remotely.
* To visualize motor status and commands using an OLED display.
* To enable cycle-based movements for predefined steps.
* To allow EEPROM-based persistent speed and direction settings.

**5. Hardware Components**

|  |  |
| --- | --- |
| Component | Description |
| ESP32 Dev Board | Core microcontroller with Wi-Fi and Bluetooth support |
| 28BYJ-48 Stepper Motor | Unipolar motor suitable for small automation |
| ULN2003 Motor Driver | Controls current to the stepper coils |
| IR Remote + Receiver Module | User interface for remote control |
| 0.96” OLED Display (SSD1306) | Displays motor status, speed, direction |
| Jumper Wires, Breadboard, Power Supply | For wiring and setup |

**6. Software Used**

* **Arduino IDE** (for code development and upload)
* **Wokwi Simulator** (optional for virtual testing)
* **Fritzing** (optional for creating circuit diagrams)

**7. Pin Configuration**

|  |  |
| --- | --- |
| Component | ESP32 GPIO |
| Stepper Motor IN1 | GPIO 14 |
| Stepper Motor IN2 | GPIO 12 |
| Stepper Motor IN3 | GPIO 13 |
| Stepper Motor IN4 | GPIO 15 |
| IR Receiver | GPIO 4 |
| OLED SDA | GPIO 21 |
| OLED SCL | GPIO 22 |

**8. System Workflow**

1. **System Boot-up**:
   * Displays project intro and contributor names.
   * Loads speed and direction settings from EEPROM.
2. **IR Command Reception**:
   * Uses IRremote library to receive command values.
   * Commands include direction control, speed adjustment, cycle count, pause/resume.
3. **Motor Movement**:
   * Directional buttons (Forward/Backward) control continuous movement.
   * Number keys (0–9) allow selecting number of cycles (mapped to 1–10).
   * Uses AccelStepper library for smooth movement.
4. **Display Feedback**:
   * OLED shows current command, speed, direction, and motor status.
   * Includes smooth progress bar while executing multi-cycle movement.
5. **Persistent Settings**:
   * Saves speed and direction in EEPROM upon every valid command.

**9. Code Highlights**

* **EEPROM Integration**: Saves last speed and direction to avoid reset loss.

|  |
| --- |
| * // EEPROM Addresses * #define EEPROM\_SPEED\_ADDR 0 * #define EEPROM\_DIR\_ADDR 4 * void saveToEEPROM() { * EEPROM.writeInt(EEPROM\_SPEED\_ADDR, speed); * EEPROM.writeBool(EEPROM\_DIR\_ADDR, previousDirectionForward); * EEPROM.commit(); * } * void loadFromEEPROM() { * speed = EEPROM.readInt(EEPROM\_SPEED\_ADDR); * previousDirectionForward = EEPROM.readBool(EEPROM\_DIR\_ADDR); * if (speed < 100 || speed > 1000) speed = 300; * } |

* **OLED Display**: Provides real-time updates with command feedback and a progress bar.

|  |
| --- |
| * void displayStatus(const char\* action) { * display.clearDisplay(); * display.setCursor(0, 0); * display.setTextSize(1); * display.println(action); * display.setCursor(0, 20); * display.print("Speed: "); display.println(speed); * display.setCursor(0, 30); * display.print("Direction: "); * display.println(previousDirectionForward ? "Forward" : "Backward"); * display.setCursor(0, 50); * display.print("State: "); * display.println(motorActive ? "Running" : "Stopped"); * display.display(); * } * void showCommandFeedback(int cycles, bool dir) { * display.clearDisplay(); * display.setCursor(0, 0); * display.setTextSize(1); * display.print("Cmd: Move "); * display.print(cycles); * display.print(" Cycle "); * display.println(dir ? "Forward" : "Backward"); * display.display(); * delay(1000); * } |

* **Cycle Mode**: User can press a number key (0 = 1 cycle, 1 = 2 cycles…) followed by Forward/Backward.

|  |
| --- |
| * void moveNCycle(int n, bool dir) { * stepper.setSpeed(dir ? abs(speed) : -abs(speed)); * int steps = 4096; // one cycle for 28BYJ-48 * stepper.move(steps \* (dir ? n : -n)); * while (stepper.distanceToGo() != 0) { * stepper.run(); * } * // Display final movement summary * display.clearDisplay(); * display.setCursor(0, 0); * display.setTextSize(1); * display.print("Moved "); * display.print(n); * display.print(" Cycle "); * display.println(dir ? "Forward" : "Backward"); * display.display(); * delay(1000); * } |

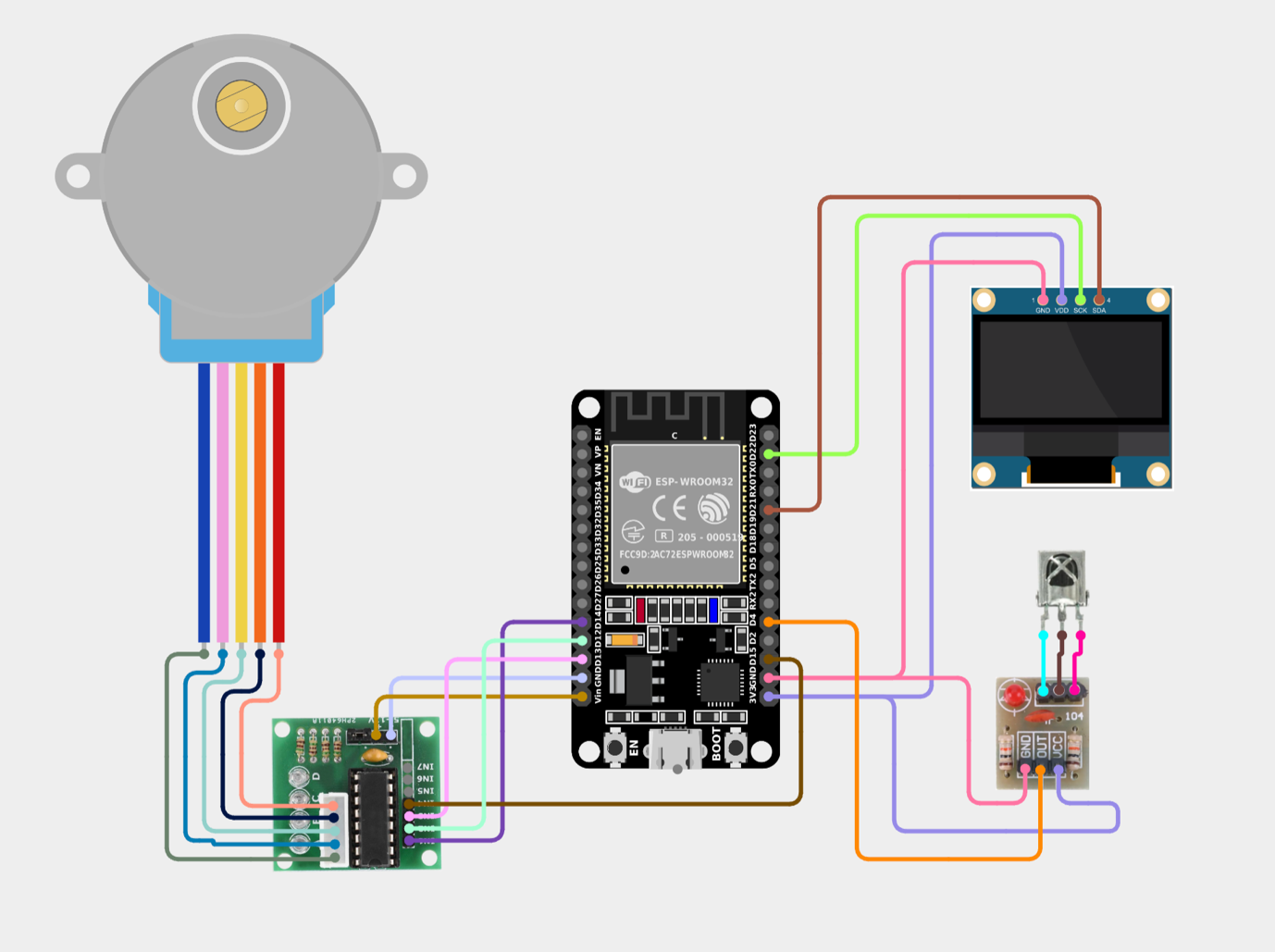
* **Project Name and Contributors:** At the time of starting project it shows project name and the name of contributors

|  |
| --- |
| * void showProjectIntro() { * display.clearDisplay(); * display.setTextSize(1); * display.setCursor(0, 0); * display.println("Remote Stepper Motor"); * display.println("Controller System"); * display.display(); * delay(1500); // Show project title * display.clearDisplay(); * display.setCursor(0, 0); * display.setTextSize(1); * display.println("Presented by:"); * display.println("-------------"); * display.println("- Abdullah(383)"); * display.println("- Khaled(391)"); * display.println("- Shanto(379)"); * display.println("- Tawhid(395)"); * display.display(); * delay(2000); // Show contributor names * } |

* **Key logics:**

|  |  |
| --- | --- |
| * void loop() { * if (IrReceiver.decode()) { * int command = IrReceiver.decodedIRData.command; * Serial.print("IR Command: "); Serial.println(command); * // showCommandFeedback(command,dir); * if (command >= 0 && command <= 9) { * stepRequest = command + 1; // Consider 0 as 1, 1 as 2, ..., 9 as 10 * cycleAwaitDirection = true; * displayStatus("Press F/B for Cycle"); * } * else if (cycleAwaitDirection && (command == forward || command == backword)) { * bool dir = (command == forward); * showCommandFeedback(stepRequest, dir); // Show detailed feedback * moveNCycle(stepRequest, dir); // Just move requested cycles once * cycleAwaitDirection = false; * stepRequest = 0; * }else { * switch (command) { * case 17: * moveForward = true; * moveBackward = false; * motorActive = true; * previousDirectionForward = true; * idleToggleState = false; * displayStatus("Moving Forward"); * break; * case 16: * moveBackward = true; * moveForward = false; * motorActive = true; * previousDirectionForward = false; * idleToggleState = false; * displayStatus("Moving Backward"); * break; * case 19: * speed += 100; * if (speed > 1000) speed = 1000; * stepper.setSpeed(speed); * displayStatus("Speed Increased"); * break; * case 18: * speed -= 100; * if (speed < 100) speed = 100; * stepper.setSpeed(speed); * displayStatus("Speed Decreased"); * break; | case 11:  moveForward = false;  moveBackward = false;  motorActive = false;  idleToggleState = false;  stepper.stop();  displayStatus("Motor Stopped");  break;  case 28:  if (!idleToggleState) {  motorActive = false;  idleToggleState = true;  displayStatus("Paused (Idle)");  } else {  motorActive = true;  if (previousDirectionForward) {  moveForward = true;  moveBackward = false;  } else {  moveBackward = true;  moveForward = false;  }  idleToggleState = false;  displayStatus("Resumed from Idle");  }  break;  default:  displayStatus("Unknown Cmd");  break;  }  }  saveToEEPROM();  IrReceiver.resume();  delay(300);  }  if (motorActive) {  if (moveForward) {  stepper.setSpeed(abs(speed));  stepper.runSpeed();  } else if (moveBackward) {  stepper.setSpeed(-abs(speed));  stepper.runSpeed();  }  }  } |

**10. Circuit Diagram**



**11. How to Run**

1. Connect all hardware components as per pin mapping.

2. Upload the Arduino code to ESP32 using Arduino IDE.

3. Power the ESP32 and motor.

4. Use IR remote to:

- Select cycles (0-9)

- Move forward/backward

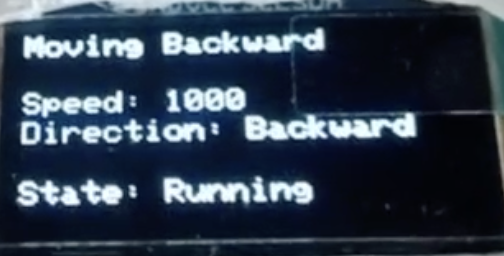
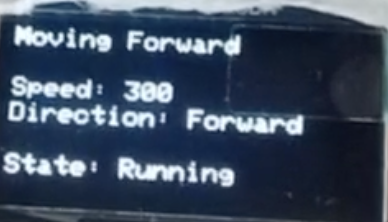
- Adjust speed

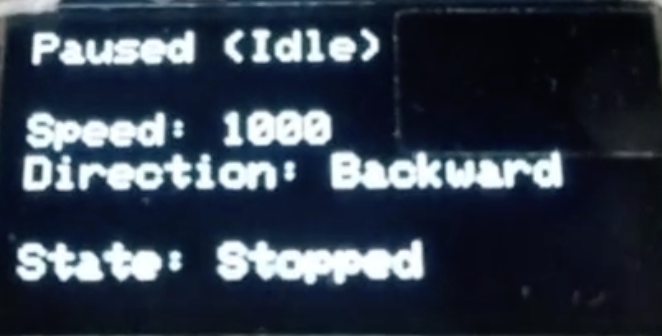
- Start/Stop motor

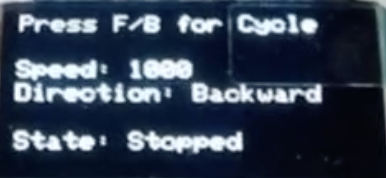
- Pause/Resume

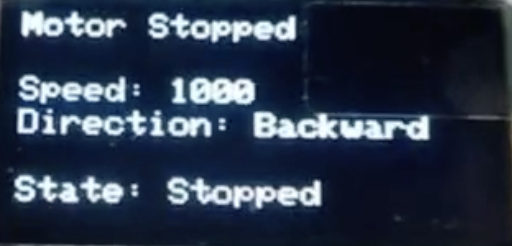
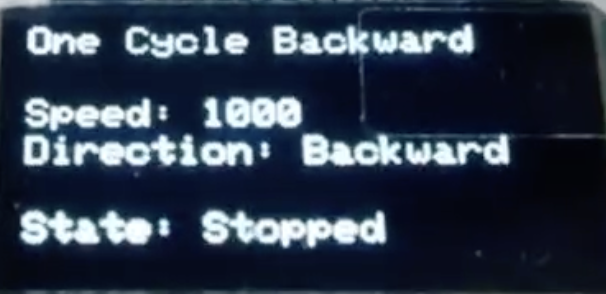
**12. Future Improvements**

* ✅ Add **Wi-Fi-based web control** using ESP32’s capabilities.
* ✅ Add **auto-homing and limit switches** for safer operations.
* ✅ Add support for **partial angle rotation** (e.g., 90°, 180°).
* ✅ Implement **serial command interface** for desktop control.

**13. Screenshots / Demo**







**14. Developed By**

**Team JU CSE**

* Abdullah (383)
* Khaled (391)
* Shanto (379)
* Tawhid (395)

**15. License**

This project is licensed under the **MIT License**.

See the LICENSE file for more details.

**16. Useful References**

* [28BYJ-48 Stepper Motor Documentation](https://lastminuteengineers.com/28byj48-stepper-motor-arduino-tutorial/)
* [Adafruit SSD1306 OLED Library](https://learn.adafruit.com/monochrome-oled-breakouts)
* [AccelStepper GitHub](https://www.airspayce.com/mikem/arduino/AccelStepper/)
* [IRremote Library GitHub](https://github.com/Arduino-IRremote/Arduino-IRremote)
* [Wokwi Simulator](https://wokwi.com/)

**17. Appendices**

**A. IR Remote Codes Mapping**

|  |  |
| --- | --- |
| Button Label | Decimal Value |
| FORWARD | 17 |
| BACKWARD | 16 |
| SPEED + | 19 |
| SPEED - | 18 |
| STOP | 11 |
| PAUSE/RESUME | 28 |
| Numbers 0-9 | 0–9 |

**B. OLED Display Behavior**

* Startup screen
* Status updates: Moving Forward, Motor Stopped, Paused
* Smooth animated progress bar in cycle mode