TEAM 5: WAGA3 ELALB

may god bless our work

About Our Team

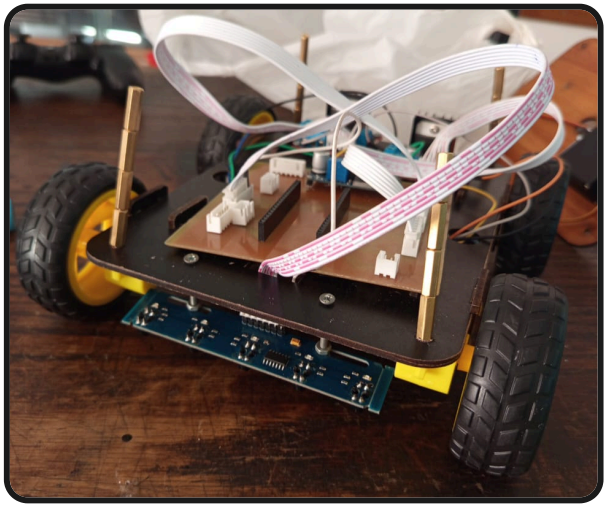
We are a team of 6 undergraduate students making a little robot under the guidance of the IEEE project PROTONS25! We are excited to share our adventure through the wonder, through hell of fire to the rising sea of electricity - every bit of testing and prototype, mechanical workshops and software crashes.

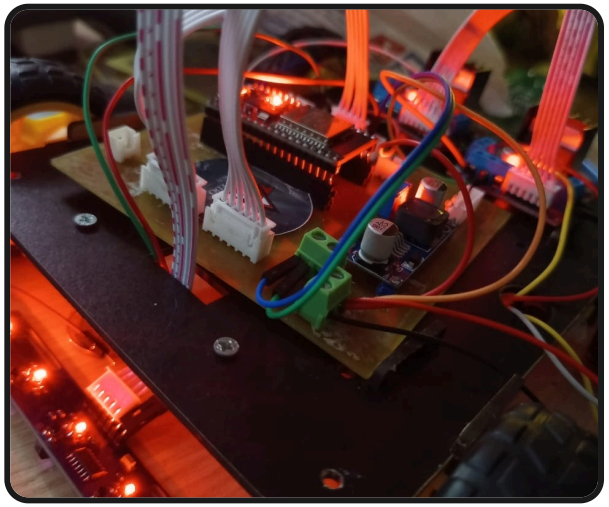
We introduce to you first our specialized teams working in harmony to bring this robotic vision to life!

WAGA3 ELALB TECHNICAL REPORT

TEAM 5: WAGA3 ELALB

We are **TEAM 5: WAGA3 ELALB** from Protons, IEEE AlexSB. Our journey represents the passion and dedication of six undergraduate students united by a common goal - to create an innovative robotic system that pushes the boundaries of what's possible in embedded systems and mechatronics.





Team Members & Roles:

- **Alfares Ahmed** - "The Mad Electrician" (Hardware: Schematic Design & Layout)
- **Omar Mohamed** - "The Cable Possessor" (Hardware: Cable Management & Integration)
- **Ziad Moustafa** - "THE MECHANIC" (Mechanical: Hammer Design & Fabrication)
- **Mohamed Elsaeed** - "The Blonde Screw Connector" (Mechanical: Base Design & Assembly)
- **Amr Emad** - "The Tallest Titan" (Software: Research Functions & Logic)

- **Mohamed Ali** - "King of madhbahah alqalea" (Software: Research Functions & Logic)

Acknowledgments:

We extend our heartfelt appreciation to every mentor who was in Protons not just for helping us but because you let us make friends and you let us be your friends along the way and for their insightful guidance and unwavering support in helping us reach these outcomes. This journey has been incredible. We would also like to express our deepest gratitude to our parents for their continuous encouragement and for granting us access to the learning resources that played a key role in our progress. Without their support, this achievement would not have been possible.

Abstract

This project explores the design and implementation of a highly functional robotic system powered by an ESP32 microcontroller, designed to seamlessly integrate autonomous navigation and manual control capabilities. The robot operates in two distinct modes: **Autonomous**, where it follows a designated line on the ground using an array of five IR sensors, and **Manual**, which allows for direct control through a PS4 controller connected via Bluetooth.

To enhance versatility, the robot features four DC motors strategically selected to balance torque and speed, providing optimal performance under varying loads. one servo for the hammr mechanism.

Hardware System Overview

Our hardware team, led by Alfares Ahmed and Omar Mohamed, designed a robust electronic infrastructure:

🔌 Schematic Design & Layout (Alfares Ahmed)

ESP32-WROOM-32 Development Board 38-Pin serves as the brain, featuring Wi-Fi, and Bluetooth/BLE for versatile connectivity and real-time decision making. Custom PCB designs with optimal component placement.

🔗 Cable Management & Integration (Omar Mohamed)

8 x JST PINS AND CABLES with organized wiring harnesses provide clean connections between sensors, actuators, and the microcontroller, ensuring reliable signal integrity.

⚡ Power & Mobility System

4x DC Geared Motors (148-600 RPM) deliver optimal torque-speed balance, powered by a **12V rechargeable battery system** with clean power distribution.

🛠️ Control Interface

Custom PCBs with L293D Motor Drivers ensure reliable power distribution and motor control, with modular design for easy maintenance and upgrades.

Mechanical Engineering Process

Our mechanical team, featuring Ziad Moustafa and Mohamed Elsaeed, brought physical form to our robotic vision:

🔨 Hammer Design & Fabrication (Ziad Mohamed)

- Designed specialized hammer mechanism for object manipulation
- Engineered impact-resistant components for durability
- Optimized weight distribution for precise striking action
- Integrated servo motor for controlled hammer movement

🏗️ Base Design & Assembly (Mohamed Elsaeed)

- Created stable base platform for all component integration
- Designed two-tiered structure for optimal weight distribution
- Implemented modular assembly system for easy maintenance
- Ensured structural integrity under operational stresses

✂️ Collaborative Fabrication

- Both team members collaborated on cutting and assembling designs
- Precision cutting of wood and acrylic components
- Quality control checks for all mechanical joints
- Iterative testing of structural stability

🎯 Final Mechanical Integration

- Selected wood base for stability and acrylic for lightweight components
- Implemented curved IR sensor array for optimal line detection
- Integrated removable sections for maintenance accessibility
- Ensured modular design for future upgrades and modifications

Software Architecture & Functions

Our software team, led by Amr Emad and Mohamed Ali, implemented sophisticated control systems divided into research packets:

📚 Research & Development Packets

- **Packet 1 - Control Algorithms:** Dual-mode operation with seamless switching between Autonomous and Manual modes
- **Packet 2 - Sensor Integration:** Multi-sensor fusion from 5x IR sensors for robust line detection
- **Packet 3 - Communication Protocols:** Bluetooth-based remote control with real-time input processing

🧠 Autonomous Intelligence Functions

- **Multi-Sensor Fusion:** Data integration from 5x IR sensors for robust line detection
- **Error Correction:** Real-time adjustment based on sensor feedback and positional errors
- **Obstacle Response:** Adaptive behavior patterns for environmental navigation

🎮 Manual Control Functions

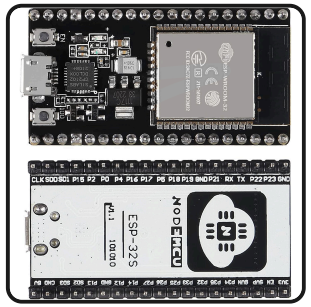
- **PS4 Controller Integration:** Bluetooth-based remote control with real-time input processing
- **Motor Speed Control:** Precise PWM-based velocity management for smooth movement
- **Servo Position Control:** Accurate angular positioning for arm and gripper operations
- **Real-time Feedback:** Status indicators and diagnostic reporting

🛠️ System Management Logic

- **Resource Optimization:** Efficient memory and processing power utilization
- **Power Management:** Battery monitoring and power distribution control
- **Safety Protocols:** Overload protection and emergency stop functionality
- **Modular Code Structure:** Organized, maintainable, and scalable architecture

Hardware Components

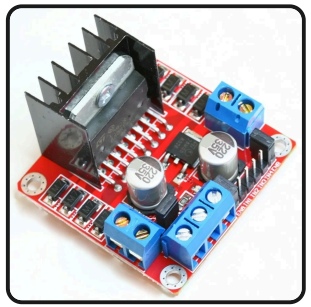
1. The Microcontroller



ESP32-WROOM-32 Development Board 38-Pin

The **ESP32-WROOM-32 Development Board 38-Pin** is a powerful and cost-effective microcontroller ideal for IoT projects. It features **dual-core processing**, **Wi-Fi**, and **Bluetooth/BLE** integration, offering versatile connectivity with **38 pins** running at 240 MHz

2. Servo Motors



2x L298N Motor Drivers were used to control the four DC motors. The L298N modules provide reliable dual H-bridge motor control, allowing for independent speed and direction management of each motor, which is essential for precise movement and maneuverability of the robot.

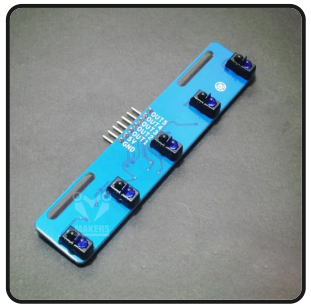
2. Servo Motors

MG996R 360 degrees (for the hammer) after knowing that a plastic servo would not provide the necessary torque and durability.

3. DC Motors

After calculating the required torque, we determined that four DC motors would be necessary to achieve the optimal speed and torque for our design.

4. Line Tracking Sensor Module 5 Channel



We selected the **Line Tracking Sensor Module 5 Channel** because it provides five sensors, which is the optimal number for our design. This configuration allows for accurate line detection and effective navigation.

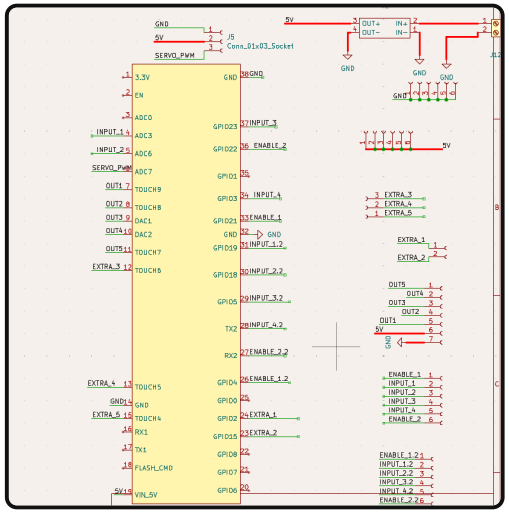
5. Rechargeable Batteries

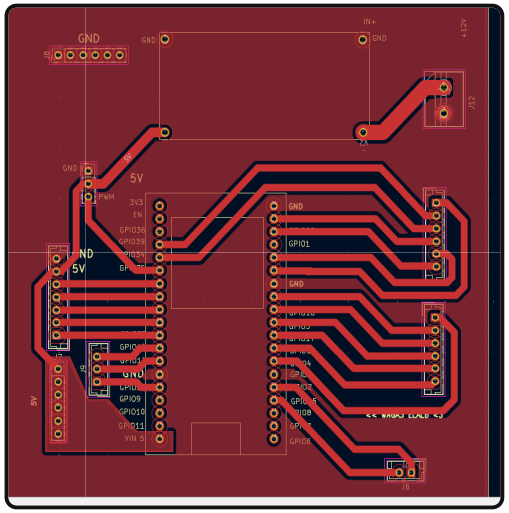
We have a total of three batteries, providing a combined output of 12V, with an additional spare battery on hand in case of any damage to the others.

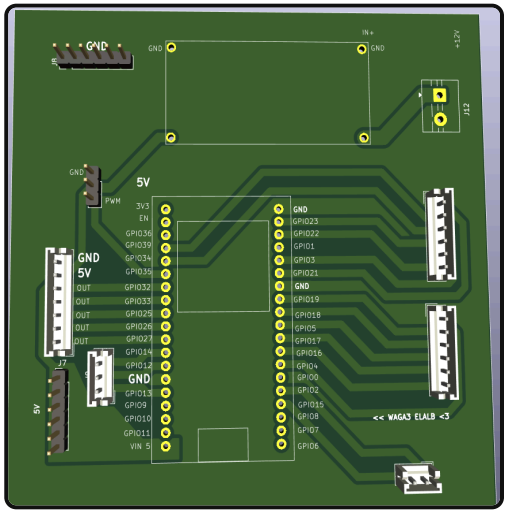
Schematics and PCBs

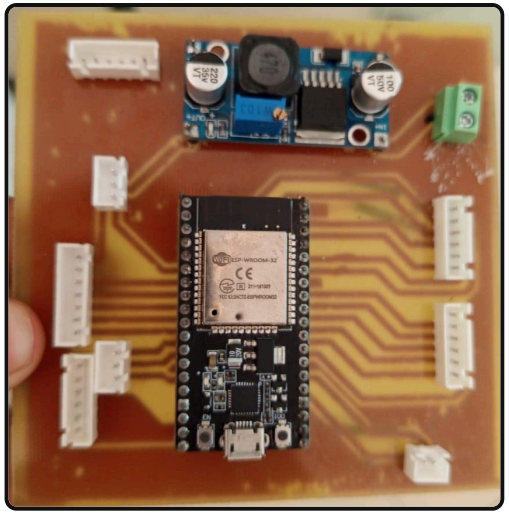
we have made over 10 fabrication and design fixing problems found on the way to the final PCB design not perfect but functional.

The PCB







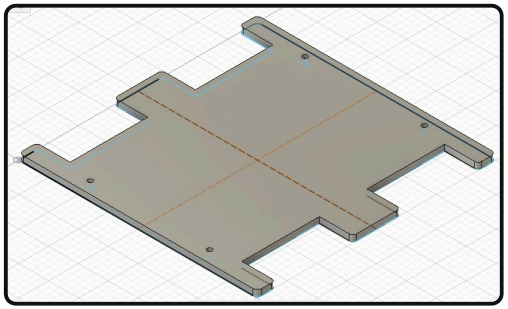


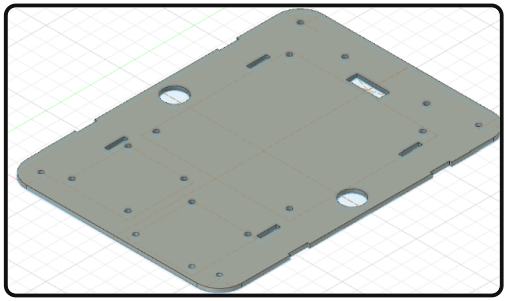
Components:

- Buck Converter
- ESP32 microcontroller
- Line Tracking Sensor Module 5 Channel
- 8 JST connectors for cable management
- 8 JST cables
- 1 screw terminal for the battery connection
- 38 female pin headers
- 4 male pin headers

Mechanical Design

The Base

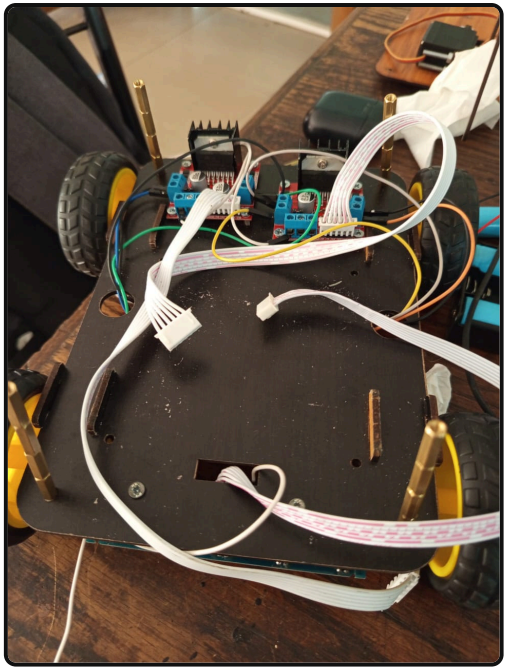




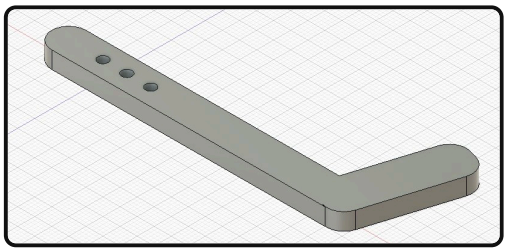
We initially selected this base; however, upon further assessment, we discovered that it is insufficient in size to accommodate the required components.

The base had extra parts at the wheels, so we were concerned the robot might make friction with the wheels.

And then we reached to the **final base** this base represents our most successful design to date. It will be a two-tiered structure, and I would like to detail the placement of each component. The base is exceptionally stable and features a curved section for the infrared (IR) sensors.

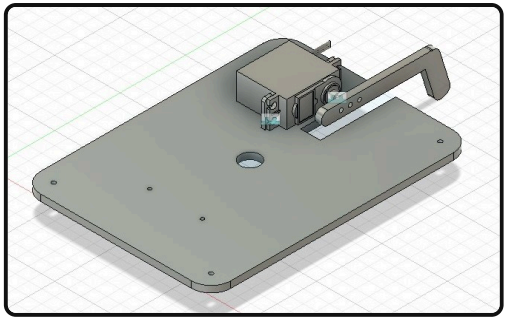


Hammer



We initially focused on practicing the mechanics of gears and exploring how to achieve the most stable design for the gripper.

This is the final result we achieved: we utilized a hammer without a gear system, which is simpler and more effective for our needs.



We added an additional task by creating an arm that holds the hammer with the servo.

Software

In our software development, we initially began by implementing manual code and utilizing the Bluetooth module to establish a connection with a smartphone. However, we soon realized that this approach was too slow for our needs. Consequently, we considered the possibility of integrating a PS4 controller to enhance the control and responsiveness of the robot.

The Final Code

Our final code includes both autonomous mode and manual mode. In autonomous mode, the robot follows a designated line by interpreting the readings from the infrared (IR) sensors. In contrast, manual mode employs user-controlled commands.

and we want to thank omar Mohamed for his help with the software although it wasn't his part he did it **for the team**

```
nst int MOTOR1_IN1 = 3;  //*GPIO3
nst int MOTOR1_IN2 = 23; //*GPIO23
nst int MOTOR1_IN3 = 15; //*GPIO15
nst int MOTOR1_IN4 = 2;  //*GPIO2
nst int MOTOR1_EN1 = 21; //*GPIO21
nst int MOTOR1_EN2 = 22; //*GPIO22

nst int MOTOR2_IN1 = 17; //*GPIO17
nst int MOTOR2_IN2 = 5;  //*GPIO5
nst int MOTOR2_IN3 = 18; //*GPIO18
nst int MOTOR2_IN4 = 19; //*GPIO19
nst int MOTOR2_EN1 = 4;  //*GPIO4
nst int MOTOR2_EN2 = 16; //*GPIO16

nst int IR1 = 32; //*GPIO32
nst int IR2 = 33; //*GPIO33
nst int IR3 = 25; //*GPIO25
nst int IR4 = 26; //*GPIO26
nst int IR5 = 27; //*GPIO27

nst int DEFAULT_SPEED = 200;
nst int TURN_SPEED = 150;
nst int LINE_SPEED = 180;
nst int MAX_LINE_LOST_COUNT = 10;
nst int IR_THRESHOLD = 2000;    // ADC threshold (0-4095). Tune to your
nst int JOYSTICK_DEADZONE = 15; // 0-255 PS4 stick deadzone
nst int SERVO_PIN = 14;         // Choose a free PWM-capable pin
rvo panServo;
```

```
    LINE_FOLLOWING_MODE
};

Mode currentMode = MANUAL_MODE;
unsigned long lastCommandTime = 0;
int lineLostCounter = 0;
bool sensorInverted = false;
unsigned long lastSensorPrint = 0;
unsigned long lastPs4InputTime = 0;

void setupHardware()
{
    // Configure motor control pins as outputs
    pinMode(MOTOR1_IN1, OUTPUT);
    pinMode(MOTOR1_IN2, OUTPUT);
    pinMode(MOTOR1_IN3, OUTPUT);
    pinMode(MOTOR1_IN4, OUTPUT);
    pinMode(MOTOR1_EN1, OUTPUT);
    pinMode(MOTOR1_EN2, OUTPUT);

    pinMode(MOTOR2_IN1, OUTPUT);
    pinMode(MOTOR2_IN2, OUTPUT);
    pinMode(MOTOR2_IN3, OUTPUT);
    pinMode(MOTOR2_IN4, OUTPUT);
    pinMode(MOTOR2_EN1, OUTPUT);
    pinMode(MOTOR2_EN2, OUTPUT);

    // Configure IR sensor pins as inputs (analog capable)
    pinMode(IR1, INPUT);
    pinMode(IR2, INPUT);
```

```
    // Servo
    panServo.attach(SERVO_PIN);
    panServo.write(90);
}

void setMotor1Direction(int in1, int in2, int in3, int in4)
{
    digitalWrite(MOTOR1_IN1, in1);
    digitalWrite(MOTOR1_IN2, in2);
    digitalWrite(MOTOR1_IN3, in3);
    digitalWrite(MOTOR1_IN4, in4);
}

void setMotor2Direction(int in1, int in2, int in3, int in4)
{
    digitalWrite(MOTOR2_IN1, in1);
    digitalWrite(MOTOR2_IN2, in2);
    digitalWrite(MOTOR2_IN3, in3);
    digitalWrite(MOTOR2_IN4, in4);
}

void setMotor1Speed(int speed)
{
    speed = constrain(speed, 0, 255);
    analogWrite(MOTOR1_EN1, speed);
    analogWrite(MOTOR1_EN2, speed);
}

void setMotor2Speed(int speed)
{
    speed = constrain(speed, 0, 255);
    analogWrite(MOTOR2_EN1, speed);
    analogWrite(MOTOR2_EN2, speed);
}
```

```
    // Motor 2: Backward (to go forward)
    setMotor2Direction(LOW, HIGH, LOW, HIGH);
    setMotor2Speed(speed);
}

void moveBackward(int speed)
{
    // Motor 1: Backward (to go backward)
    setMotor1Direction(LOW, HIGH, LOW, HIGH);
    setMotor1Speed(speed);

    // Motor 2: Forward (to go backward)
    setMotor2Direction(HIGH, LOW, HIGH, LOW);
    setMotor2Speed(speed);
}

void turnLeft(int speed)
{
    // Motor 1: Backward (right side - to turn left)
    setMotor1Direction(LOW, HIGH, LOW, HIGH);
    setMotor1Speed(speed);
```

```
// line follower
int readSensorDigital(int pin)
{
    int value = digitalRead(pin);
    return sensorInverted ? !value : value;
}

int readSensorAnalog(int pin)
{
    int value = analogRead(pin); // 0-4095 on ESP32
    bool detected = value >= IR_THRESHOLD;
    return sensorInverted ? !detected : detected;
}

bool isLineDetected()
{
    int detectedCount = 0;
    if (readSensorAnalog(IR1) == 1)
        detectedCount++;
    if (readSensorAnalog(IR2) == 1)
        detectedCount++;
    if (readSensorAnalog(IR3) == 1)
        detectedCount++;
    if (readSensorAnalog(IR4) == 1)
        detectedCount++;
    if (readSensorAnalog(IR5) == 1)
        detectedCount++;

    // Debug: More sensitive detection (changed from 2 to 1 for better detection)
    return (detectedCount >= 1);
}

void lineFollow()
{
    int sensor1 = readSensorAnalog(IR1);
    int sensor2 = readSensorAnalog(IR2);
    int sensor3 = readSensorAnalog(IR3);
```

```
void toggleMode()
{
    if (currentMode == MANUAL_MODE)
    {
        currentMode = LINE_FOLLOWING_MODE;
        Serial.println("SWITCHED TO LINE FOLLOWING MODE");
        Serial.println("Sensors will show readings every 2 seconds");
    }
    else
    {
        currentMode = MANUAL_MODE;
        Serial.println("SWITCHED TO MANUAL MODE");
        stopMotors();
    }
}

// ===== PS4 MANUAL CONTROL =====
void handlePs4ManualControl()
{
    if (!PS4.isConnected())
    {
        return;
    }

    // Read left stick (-128..127). Convert to -255..255
    int lx = PS4.data.analog.stick.lx; // 0..255 with center ~127
    int ly = PS4.data.analog.stick.ly; // 0..255 with center ~127
    int x = lx - 127;
    int y = 127 - ly; // invert so up is +

    if (abs(x) < JOYSTICK_DEADZONE)
        x = 0;
    if (abs(y) < JOYSTICK_DEADZONE)
```

```
    {
        turnLeft(TURN_SPEED);
    }
    else if (PS4.Right())
    {
        turnRight(TURN_SPEED);
    }

    // Toggle modes with Options button
    if (PS4.Options())
    {
        toggleMode();
        delay(200);
    }

    // Servo quick positions
    if (PS4.event.button_down.cross)
    {
        panServo.write(0);
    }
    if (PS4.event.button_down.triangle)
    {
        panServo.write(90);
    }
    if (PS4.event.button_down.circle)
    {
        panServo.write(180);
    }

    // Invert sensors with Share
    if (PS4.event.button_down.share)
    {
        sensorInverted = !sensorInverted;
```

```
aga3-elalb > Untitled-1.c > handlePs4ManualControl()
void setup()
    Serial.println("Use PS4 controller for manual control. Options button toggles modes.");
}

void loop()
{
    // Manual mode with PS4 controller
    if (currentMode == MANUAL_MODE)
    {
        handlePs4ManualControl();
    }

    // Line following mode
    if (currentMode == LINE_FOLLOWING_MODE)
    {
        if (isLineDetected())
        {
            lineFollow();
        }
        else
        {
            lineLostCounter++;
            if (lineLostCounter > MAX_LINE_LOST_COUNT)
            {
                stopMotors();
                Serial.println("Line lost! Switched to Manual Mode");
                lineLostCounter = 0;
            }
            else
            {
                moveForward(LINE_SPEED / 3);
            }
        }
    }
    delay(50);
}
```

Project Timeline

Week 1-2

Project Planning & Research

Initial brainstorming, component research, and team role assignments

Week 3-4

Hardware Design

Schematic design, PCB layout, and component selection

Week 5-6

Mechanical Fabrication

Base construction, hammer mechanism design, and assembly

Week 7-8

Software Development

Manual control implementation, autonomous navigation, and PS4 integration

Week 9-10

Integration & Testing

System integration, debugging, and performance optimization

Week 11-12

Final Polish & Documentation

Final testing, documentation, and presentation preparation

Budget Statistics

Total Budget

2000EGP

Allocated for entire project

Actual Spending

3500EGP

Total expenses incurred

Budget Breakdown by Category

Electronics Components

2000EGP (57%)

- ESP32 Microcontroller
- DC Motors & Servos
- Sensors & PCB Components

Mechanical Materials

700EGP (20%)

- Wood & Acrylic Sheets
- Fasteners & Hardware
- Fabrication Tools

Power System

315EGP (9%)

- Batteries & Chargers
- Power Management

Miscellaneous

490EGP (14%)

- Cables & Connectors
- Transportation
- Documentation

Lessons Learned

💡 Technical Insights

- Early prototyping saves time in later stages
- Modular design facilitates easier debugging
- Power requirements should be calculated with margin
- Sensor placement critically affects performance

👥 Team Collaboration

- Regular communication prevents integration issues
- Version control is essential for collaborative coding
- Documentation saves time during troubleshooting
- Cross-training team members enhances flexibility

📈 Project Management

- Realistic timelines account for unexpected challenges
- Budget contingency is necessary for component failures
- Regular milestones maintain team motivation
- Thorough testing prevents last-minute crises

Team Love: The Heart of Our Success

A great team isn't just about technical skills - it's about the bonds we build and the culture we create together:

💝 Mutual Support System

We created an environment where every member feels valued and supported. Through late-night coding sessions and challenging debugging moments, we always had each other's backs.

🎯 Shared Vision & Goals

United by our passion for robotics and innovation, we aligned our individual strengths toward common objectives, celebrating every milestone together.

🤝 Trust & Communication

Open dialogue and mutual respect formed our foundation. We trusted each other's expertise while maintaining honest communication about challenges and solutions.

🎉 Joy in the Journey

From spontaneous dance breaks to shared meals during long work sessions, we found joy in the process, making memories that extend beyond the project itself.

🌟 Growth Mindset

We embraced challenges as opportunities to learn, supporting each other's growth while collectively pushing our boundaries of what we thought possible.

🤗 Family Atmosphere

Beyond teammates, we became friends and family - checking on each other's well-being, celebrating personal achievements, and building relationships that will last long after the project ends.

"Individual commitment to a group effort - that is what makes a team work, a company work, a society work, a civilization work." - Vince Lombardi

Core Values

In any successful team, it's essential to have each other's backs and stand together through thick and thin. Our journey has truly been about collaboration and growth rather than just the destination. Along the way, we've faced challenges, celebrated victories, and created unforgettable memories.

From late-night brainstorming sessions fueled by snacks to spontaneous dance breaks, we've shared laughter and leaned on each other's strengths. Each team member brought unique skills and perspectives, making our experience even more enriching.

Here are some of our favorite photos capturing the teamwork, smiles, and moments that defined our adventure together! ☺