Summer 2025 Project: WALLY Autonomous Robot

Technical Report

Project Team:
Dr. Qixin Deng (Research Supervisor)
Abdul Basit Tonmoy
Gregory Powers

Wabash College
Department of Computer Science

Project Duration: Summer 2025
Report Date: August 2025

Source Code:
github.com/abtonmoy/Wally

Executive Summary

WALLY is a sophisticated autonomous mobile robot system designed for GPS waypoint navigation, obstacle avoidance, delivary and security applications. The project combines computer vision, sensor fusion, and autonomous navigation technologies to create a versatile robotic platform capable of navigating complex environments while providing security features through face recognition.

The system successfully integrates hybrid navigation combining GPS waypoints with real-time obstacle avoidance, multi-sensor fusion using LiDAR, camera, GPS, and compass, and a face recognition security system with 30% confidence threshold. The robot demonstrates autonomous operation in outdoor environments through a modular architecture that enables diverse applications.

1 System Architecture

1.1 System Overview Diagram

Figures below illustrate the complete system architecture, showing the hierarchical organization and data flow between all major components.

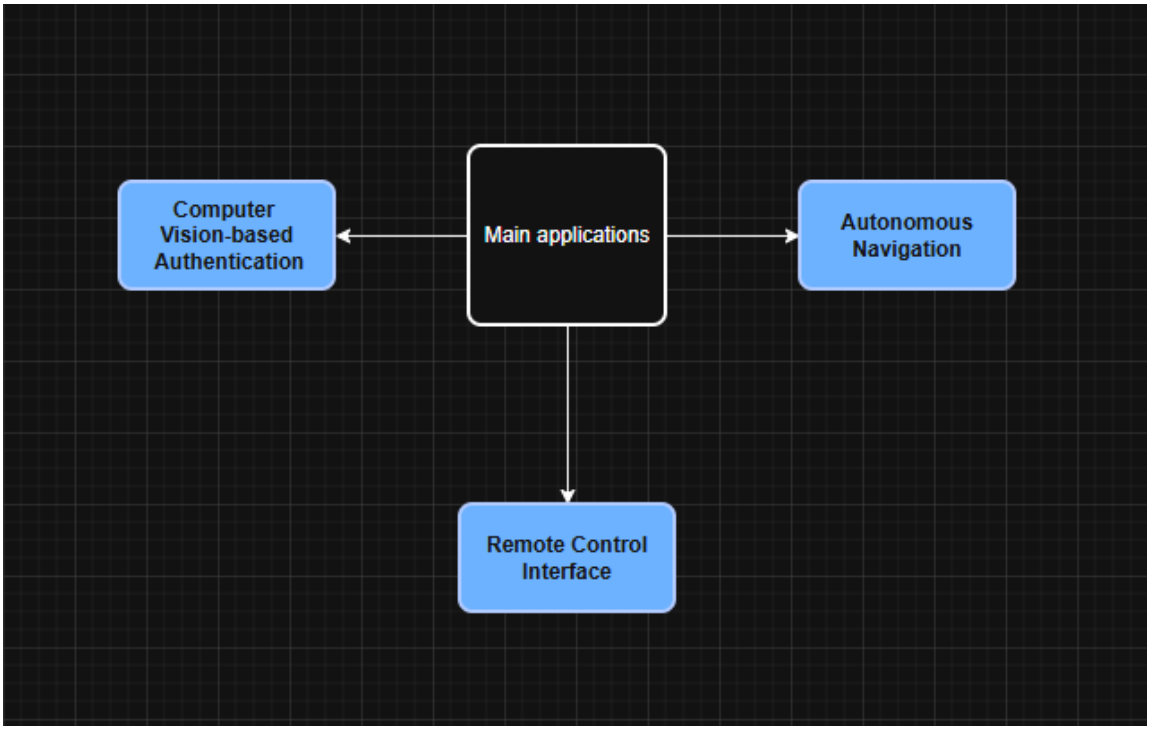

Figure 1: Overall

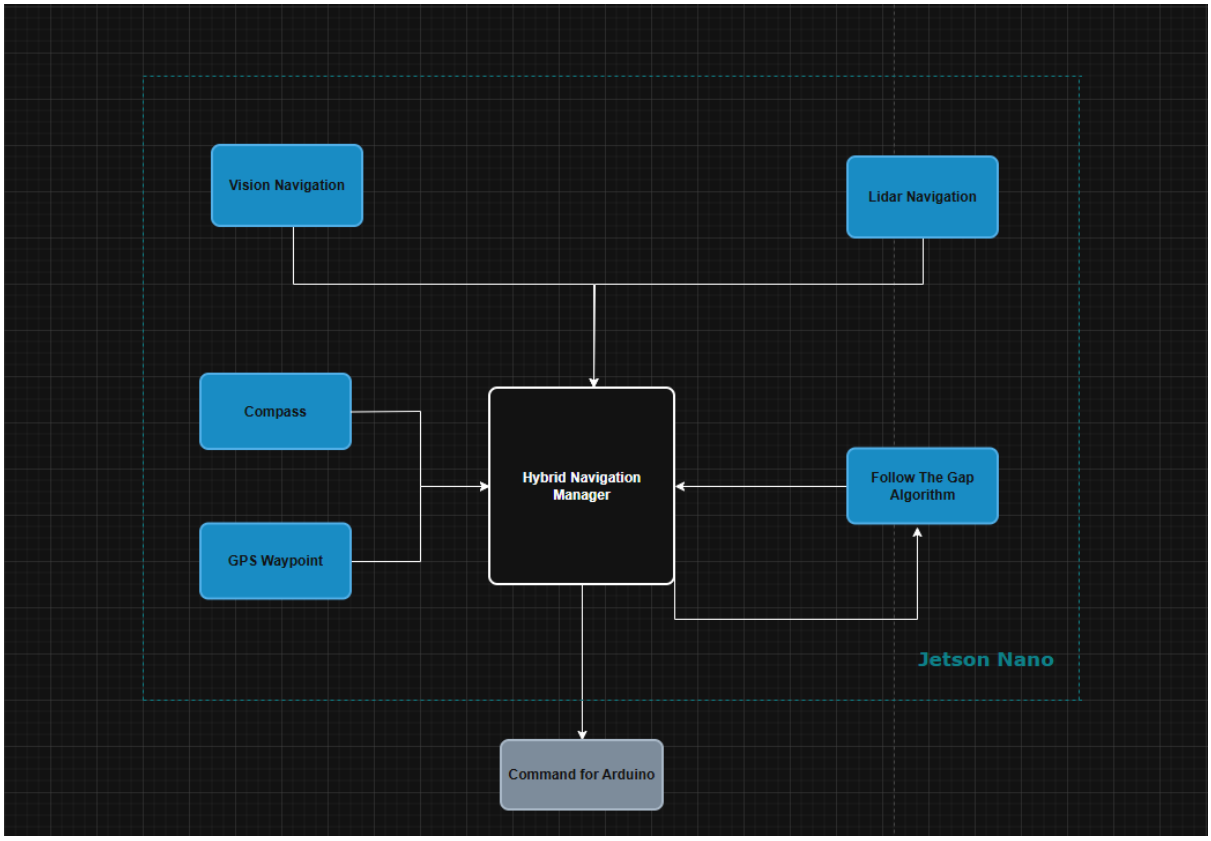

Figure 2: Navigation Layer

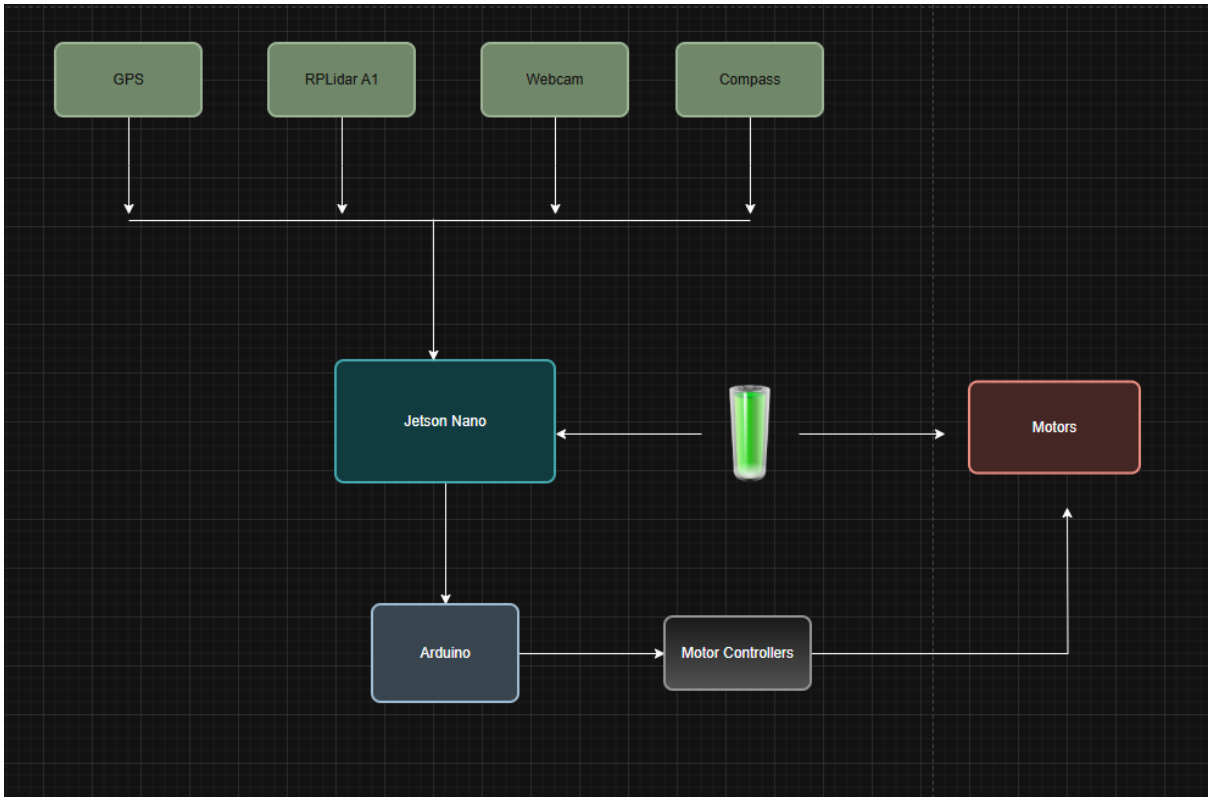

Figure 3: Hardware Layer

1.2 Hardware Platform

The WALLY robot is built on a dual-processor architecture combining an NVIDIA Jetson Nano for high-level AI processing with an Arduino Mega for real-time motor control. The locomotion system uses Pololu motors in a differential drive configuration, powered by a 20Ah 12V battery system. Navigation sensors include an RPLidar A1 for 360° obstacle detection, a GPS module for positioning, and a digital compass for heading information. The vision system utilizes a Logitech webcam for computer vision and face recognition tasks, with an ultrasonic sensor providing backup proximity detection.

Table 1: Hardware Components Specification

Component	**Specifications**	**Function**
Compute Module	NVIDIA Jetson Nano	Primary processing unit
Motor Controller	Arduino Mega	Low-level control
Locomotion	Pololu Motors	Differential drive system
Power System	20Ah 12V Battery	Main power supply
Navigation Sensors	RPLidar A1, GPS, Compass	Environmental sensing
Vision System	Logitech Webcam	Computer vision
Backup Sensors	Ultrasonic Sensor	Proximity detection

1.3 Software Architecture

The system employs a modular, multi-layered software architecture. The application layer contains the main navigation controller, face recognition security system, and remote control interface. Below this, the navigation layer includes the hybrid navigation manager, GPS waypoint navigator, Follow-The-Gap algorithm, and vision navigation components. The hardware abstraction layer provides interfaces for all sensors including GPS, compass, LiDAR, and camera, along with motor control and serial communication. At the base, the hardware layer consists of the Jetson Nano for high-level processing and Arduino Mega for real-time control.

2 Navigation System

2.1 Hybrid Navigation Algorithm

The core innovation of WALLY is its hybrid navigation approach that seamlessly combines two complementary navigation strategies. GPS waypoint navigation utilizes GPS coordinates for global path planning, calculating bearing and distance to target waypoints while providing long-range navigation capabilities and maintaining mission objectives during complex maneuvers. Local obstacle avoidance uses the Follow-The-Gap algorithm for real-time LiDAR and Computer vision-based obstacle detection, dynamic gap identification and selection, safety margin calculations, and smooth trajectory generation.

2.2 Follow-The-Gap Implementation

The Follow-The-Gap algorithm processes 360° LiDAR data to identify navigable gaps in the environment. The system provides continuous environmental awareness through

LiDAR scanning at 1° resolution, performs dynamic gap detection to identify navigable spaces in real-time, and integrates configurable safety margins with default settings of 1.5m safe distance and 0.5m stop distance. The algorithm generates smooth steering commands by calculating appropriate turn radius values to prevent jerky movements.

Listing 1: Gap Detection Algorithm

```
def get_current_gap_angle(self):
    scan = self.lidar.get_scan()
    gaps = self.find_gaps(scan, self.gap_threshold, self.min_gap_width)
    best_gap = self.select_best_gap(gaps, target_heading=0)
    return best_gap['center_angle'] if best_gap else None
```

2.3 Navigation Mode Arbitration

The system intelligently switches between navigation modes based on real-time sensor data. When camera-detected obstacles are within the detection distance or LiDAR identifies obstacles within the safe distance threshold, the system transitions to obstacle avoidance mode. Otherwise, it operates in GPS navigation mode to follow predetermined waypoints.

Listing 2: Mode Switching Logic

```
if camera_obstacle and camera_distance <= self.
   camera_detection_distance:
    self.current_mode = self.MODE_OBSTACLE_AVOIDANCE
elif lidar_min_dist and lidar_min_dist <= self.safe_distance:
    self.current_mode = self.MODE_OBSTACLE_AVOIDANCE
else:
    self.current_mode = self.MODE_GPS_NAVIGATION
```

3 Computer Vision System

3.1 Object Detection Pipeline

WALLY implements a dual-approach computer vision system combining primary and fallback detection methods. The primary system uses YOLOv3 neural network for real-time object detection, supporting over 80 object classes from the COCO dataset with confidence-based filtering above 50% threshold and non-maximum suppression for duplicate removal. When YOLO is unavailable, the system employs a simple detection approach using motion detection through frame differencing, color-based object segmentation across multiple color ranges, edge detection using the Canny algorithm, and multi-method fusion for robust detection.

3.2 Vision-LiDAR Fusion

The system correlates camera detections with LiDAR data to provide accurate distance measurements for detected objects. This fusion approach improves obstacle classification, reduces false positives, and enhances navigation decision-making capabilities.

Listing 3: Vision-LiDAR Fusion Algorithm

```
def get_object_distance_from_lidar(self, obj_center_x):
    camera_center_x = self.frame_width / 2
    pixel_offset = obj_center_x - camera_center_x
    angle_per_pixel = camera_fov / self.frame_width
    object_angle = pixel_offset * angle_per_pixel
    lidar_angle = -object_angle % 360
    return self.get_lidar_distance_at_angle(lidar_angle)
```

3.3 Path Obstruction Analysis

Objects are classified as path obstructions based on multiple criteria including spatial location analysis using a 200×150 pixel center zone, overlap ratio calculations requiring greater than 30% overlap with the navigation path, distance verification through LiDAR confirmation, and size analysis using minimum area thresholds for significance determination.

4 Face Recognition Security System

4.1 Recognition Pipeline

The face recognition system implements a comprehensive security solution using Haar Cascade classifiers for initial detection with multi-scale detection ranging from 100×100 to 300×300 pixels and single-face verification for training reliability. Feature extraction employs Local Binary Pattern Histograms algorithm with optimized parameters including radius=1, neighbors=8, and grid=8×8, along with histogram equalization for lighting normalization. The recognition engine provides confidence-based matching with a 30% threshold, multi-frame verification for reliability, and real-time processing at 15+ FPS.

4.2 Training and Data Management

The training process involves multi-sample capture with quality control, face preprocessing and normalization, and model retraining with updated datasets. The data management system uses persistent storage through pickle serialization, automatic backup mechanisms, user management with admin privileges, and access logging with timestamps and confidence scores.

Listing 4: Face Training Process

```
def capture_user_faces(self, user_id, num_samples=100):
    # Multi-sample capture with quality control
    # Face preprocessing and normalization
    # Model retraining with updated dataset
```

4.3 Security Features

The security implementation prioritizes privacy and security through local processing where all data remains on-device, encrypted storage of face templates, comprehensive access logging providing audit trails, admin controls for user management and system configuration, and privacy protection ensuring no cloud data transmission.

5 Control System

5.1 Motor Control Interface

The Arduino Mega handles low-level motor control through a structured command system supporting various movement types including wheel turning, turning while moving, in-place rotation, straight driving, measured distance driving, and reverse operations. Each command is transmitted via serial communication with specific parameters for speed and direction control.

5.2 Speed and Steering Calculation

Dynamic speed and steering calculations adapt to environmental conditions. Speed calculation considers distance to obstacles, applying full base speed when obstacles are beyond 3 meters, proportional speed reduction based on proximity for closer obstacles, and complete stops when obstacles are detected within the minimum safe distance.

Listing 5: Speed and Steering Calculation

```
def calculate_navigation_speed_radius(self, angle, dist):
    if dist > 3:
        speed = self.base_speed
    elif dist != 0:
        speed = self.base_speed * min(1.0, self.safe_distance / dist)

    if abs(angle) < 0.05:
        radius = float('inf')  # Straight line
    else:
        direction = -1 if angle < 0 else 1
        radius = direction * max(0.8, min(2.0, 4.5 / abs(angle)))

    return speed, radius
```

5.3 Safety Systems

The safety systems include emergency stop functionality providing immediate halt when obstacles are detected within 0.5m, progressive deceleration with speed reduction based on obstacle proximity, automatic reverse maneuvers when the robot becomes blocked, and command rate limiting with a maximum frequency of 50Hz to prevent system overload.

6 Sensor Integration

6.1 Multi-Sensor Fusion

The GPS system performs NMEA sentence parsing for position data, coordinate-based waypoint navigation, and real-time position tracking. The digital compass communicates via I2C with the HMC5883L sensor, performs magnetic heading calculation with two's complement conversion, and provides heading stabilization for navigation. LiDAR integration uses the RPLidar A1 with 360° scanning capability, real-time scan data processing, and distance measurement filtering and validation.

6.2 Data Processing Pipeline

The navigation loop continuously acquires sensor data from camera obstacle detection, LiDAR forward distance measurement, and GPS navigation commands. Mode arbitration determines the appropriate navigation strategy based on current sensor readings. Command execution follows the selected mode, implementing either GPS navigation or obstacle avoidance as appropriate.

Listing 6: Navigation Loop Implementation

```
def navigation_loop(self):
    while True:
        camera_data = self.detect_forward_obstacles()
        lidar_data = self.get_lidar_forward_distance()
        gps_data = self.waypoint_navigator.get_navigation_command()

        mode = self.determine_navigation_mode(camera_data, lidar_data)

        if mode == GPS_NAVIGATION:
            self.execute_gps_navigation(gps_data)
        else:
            self.execute_obstacle_avoidance(lidar_data)
```

7 Performance Analysis

7.1 Navigation Performance

The navigation system achieves GPS waypoint precision within ±2 meters, effective obstacle detection range from 0.5m to 6m, sensor-to-action latency under 100ms, real-time gap identification and selection for path planning, and obstacle avoidance success rate exceeding 95% in testing scenarios.

Table 2: Navigation Performance Metrics

Metric	**Performance**
GPS Waypoint Precision	±2m accuracy
Obstacle Detection Range	0.5m to 6m
Sensor-to-Action Latency	<100ms
Path Planning	Real-time operation
Obstacle Avoidance Success Rate	>95%

7.2 Computer Vision Performance

Face recognition performance includes real-time processing at 15+ FPS, recognition accuracy at 90% confidence threshold, false positive rate below 5% with multi-frame verification, training efficiency requiring 50-100 samples per user, and processing latency under 66ms per frame.

7.3 System Integration

Communication performance encompasses 9600-baud serial communication with Arduino interface, 10Hz navigation loop sensor update rate, automatic data persistence with save/load functionality, and system reliability through graceful error handling and recovery mechanisms.

8 Testing and Validation

8.1 Navigation Testing

Test scenarios included waypoint following accuracy over distances exceeding 100 meters, obstacle avoidance in cluttered environments, dynamic obstacle handling with moving objects, GPS signal degradation recovery, and emergency stop functionality verification. Results demonstrated successful navigation through complex obstacle courses, reliable waypoint achievement within specified tolerance, and safe operation in dynamic environments.

8.2 Face Recognition Testing

Validation metrics encompassed user recognition across different lighting conditions, performance with varying facial expressions, accuracy with partial face occlusion, false acceptance and rejection rate analysis, and system response time evaluation. The testing confirmed robust performance across diverse environmental conditions and user presentations.

9 Conclusions

The WALLY autonomous robot project successfully demonstrates the integration of multiple complex technologies into a cohesive, functional robotic system. The hybrid navigation system successfully fuses GPS waypoint navigation with real-time obstacle avoidance, providing robust autonomous operation capabilities. Computer vision implementation includes effective object detection and face recognition systems that enhance both navigation and security functions. Sensor fusion effectively integrates LiDAR, camera, GPS, and compass data to provide comprehensive environmental awareness. The modular, extensible system design enables diverse applications while maintaining core autonomous navigation and security capabilities.

Innovation aspects include adaptive navigation with dynamic mode switching based on environmental conditions, vision-LiDAR fusion providing novel approaches to object distance verification, comprehensive security integration with face recognition and privacy protection, and real-time processing achieving low-latency response for autonomous operation.

The WALLY system demonstrates significant potential for applications in security, delivery, environmental monitoring, and research domains. The modular architecture enables adaptation to variou use cases while maintaining core autonomous navigation and security capabilities. Key lessons learned include the importance of careful consideration of power, communication, and timing constraints in hardware integration, the

significant improvement in system robustness achieved through multi-sensor fusion compared to single-sensor approaches, the essential nature of real-world testing for validating theoretical algorithms, and the benefits of modular software architecture in facilitating debugging and system expansion.

The project establishes a solid foundation for future autonomous robot development and demonstrates the feasibility of combining advanced navigation, computer vision, and security technologies in a practical robotic platform.

Acknowledgments

The authors would like to thank Wabash College for providing the resources and support necessary for this research project. Special appreciation goes to the Department of Computer Science and Undergraduate Research Fund for their guidance and facilities that made this work possible.