AUTOMATED NURSING ASSISTANT

A Mini Project Report

Submitted to the APJ Abdul Kalam Technological University

in partial fulfillment of requirements for the award of degree

Bachelor of Technology

in

Electronics and Communication Engineering

by

ABHAY M PRASANNAKUMAR (NSS22EC003)

ABHISHEK S P (NSS22EC008)

AKSHARA S PRASAD (NSS22EC022)

ARYA V (NSS22EC044)

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

NSS COLLEGE OF ENGINEERING PALAKKAD

KERALA

MARCH 2025

DEPT. OF ELECTRONICS & COMMUNICATION ENGINEERING

NSS COLLEGE OF ENGINEERING PALAKAKD

2024-25

CERTIFICATE

This is to certify that the report entitled **AUTOMATED NURSING ASSISTANT** submitted by **ABHAY M PRASANNAKUMAR** (NSS22EC003),**ABHISHEK S P** (NSS22EC008) , **AKSHARA S PRASAD** (NSS22EC022), **ARYA V** (NSS22EC044), to the APJ Abdul Kalam Technological University in partial fulfillment of the B.Tech. degree in Electronics and Communication Engineering is a bonafide record of the miniproject work carried out by them under our guidance and supervision. This report in any form has not been submitted to any other University or Institute for any purpose.

Prof Vinod G

Project Guide
Project Coordinator
Dept.of ECE
NSS College of Engineering
Palakkad

Prof Dr.Sreeleja N Unnithan

Head of the Department
Dept.of ECE
NSS College of Engineering
Palakkad

DECLARATION

We hereby declare that the project report **AUTOMATED NURSING ASSISTANT**, submitted for partial fulfillment of the requirements for the award of degree of Bachelor of Technology of the APJ Abdul Kalam Technological University, Kerala is a bonafide work done by us under the supervision of Prof Vinod G

This submission represents our ideas in our own words and where ideas or words of others have been included, we have adequately and accurately cited and referenced the original sources.

We also declare that we have adhered to ethics of academic honesty and integrity and have not misrepresented or fabricated any data or idea or fact or source in my submission. We understand that any violation of the above will be a cause for disciplinary action by the institute and/or the University and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been obtained. This report has not been previously formed the basis for the award of any degree, diploma or similar title of any other University.

Abhay M Prasannakumar

Abhishek SP

Akshara S Prasad

Arya V

Palakkad

01-04-2025

Contents

Abstract

The project, **Automated Nursing Assistance System**, is designed to streamline hospital logistics by employing an autonomous medical trolley that efficiently navigates hospital corridors using the A * algorithm. The system can deliver medical supplies, food, and other essentials while managing requests based on priority. Integrated with Blynk IoT, the system allows remote monitoring and task allocation, ensuring real-time tracking and optimized movement. Equipped with motor-controlled navigation, it autonomously follows computed paths, stops at designated locations, and returns to the home position after task completion. By reducing manual effort in routine hospital operations, the system enhances efficiency, accuracy, and reliability in healthcare environments, paving the way for future advancements in smart hospital automation.

Acknowledgement

We take this opportunity to express our deepest sense of gratitude and sincere thanks to everyone who helped us to complete this work successfully. We express our sincere thanks to Prof Dr Sreeleja N Unnithan, Head of the Department, Department of Electronics and Communication Engineering, NSS College of Engineering, Palakkad for providing us with all the necessary facilities and support.

We would like to express our sincere gratitude to Prof Vinod G, Associate Professor, Department of Electronics and Communication Engineering, NSS College of Engineering, Palakkad for the constant support, mentorship and co-operation.

Last, but not the least, we take pleasant privilege in expressing our heartful thanks to our teachers, friends and all well-wishers who were of precious help in completing the miniproject.

Abhay M Prasannakumar

Abhishek SP

Akshara S Prasad

Arya V

List of Figures

List of Abbreviations

1. ESP32. . . . . ESPRESSIF SYSTEM PROCESSOR 32BIT

Mobile Communication

2. GPS. . . . . Global Positioning System

Converters

3. DC. . . . . Direct Current

4. IC. . . . . Integrated Circuit

Chapter 1

Introduction

The project, Automated Nursing Assistant, is an innovative healthcare automation solution designed to enhance efficiency in hospitals and healthcare facilities. This system with an intelligent motor autonomously transport medical supplies, food, and laundry, reducing manual workload for hospital staff. It prioritizes service requests through an efficient queue management mechanism and employs A*Algorithm for room-specific deliveries. With real-time monitoring and remote control capabilities via the Blynk IoT platform, the system ensures seamless automation with minimal human intervention.. Once a task is completed, it autonomously returns to its designated control room, optimizing resource utilization. By improving the accuracy and timeliness of medical deliveries, Automated Nursing Assistance aims to revolutionize healthcare logistics, ensuring efficient service while enhancing patient care and hospital management.

Methodology

- Data collection: Gather data on hospital layouts and service requests.
- Data Analysis: Optimize path planning using A* Algrithm.
- Algorithm Development: Developed an autonomous navigation and task management.
- Deployment: Implement and evaluate the system for hospital efficiency.

Chapter 2

Literature Review

Autonomous Medical Trolley Systems for Hospital Automation

1. **IoT and Robotics in Healthcare Automation:** Recent advancements in IoT-enabled robotics have transformed hospital logistics by automating repetitive tasks such as medicine delivery and inventory management. Research by Chen et al. (2020) demonstrates how A* based path planning optimizes trolley navigation in dynamic hospital environments, reducing delivery times by 22percent while avoiding obstacles. Their work integrates WiFi SLAM and ultrasonic sensors, ensuring reliable indoor positioning without GPS. Similarly, Sharma and Lee (2021) propose a WiFi fingerprinting-based trolley that uses k-NN machine learning for localization, achieving 1.2m accuracy in real-world hospital deployments. These studies highlight the growing role of autonomous mobile robots (AMRs) in healthcare, minimizing human intervention while maintaining precision.

2. **Path Planning and Obstacle Avoidance:** Efficient navigation is critical for medical trolleys operating in crowded hospital corridors. The A algorithm, introduced by Hart, Nilsson, and Raphael (1968), remains a gold standard for optimal pathfinding. Chen et al. (2020) enhance traditional A* with dynamic cost adjustments and real-time re-planning, ensuring adaptability to moving obstacles (e.g., staff, equipment). For finer local navigation, Patel et al. (2022) combine A with Dynamic Window Approach (DWA), enabling smooth obstacle avoidance using LiDAR and RFID. Their system achieves 95percent navigation success in cluttered environments, proving the

effectiveness of multi-algorithm fusion in medical robotics.

3. Sensor Fusion and Localization: Accurate localization is a major challenge in GPS-denied hospital settings. While Sharma and Lee (2021) rely on WiFi RSSI fingerprinting, Patel et al. (2022) employ LiDAR and RFID tags for higher precision. The former offers a low-cost solution (500dollar per trolley) but requires periodic recalibration, whereas the latter achieves sub-5cm accuracy at a higher cost (8,000dollar per unit). Both approaches validate the importance of sensor fusion (e.g., ultrasonic, IMU) to compensate for individual limitations.

4. Full Automation and Task Prioritization: Beyond navigation, Patel et al. (2022) introduce a fully automated trolley with:

- RFID-based inventory tracking (real-time stock updates).
- 6-DOF robotic arm for picking/placing medicines (88–94percent success rate).
- Cloud integration with Hospital Management Systems (HMS).

This aligns with trends in IoT-driven automation, as seen in Blynk and Firebase-based hospital solutions, where real-time monitoring optimizes task queues. Their trolley reduces medicine retrieval time by 35percent, demonstrating the scalability of automation in pharmacy logistics.

Chapter 3

System Development

The development of the medical assisted trolley system integrates both hardware and software components to enable autonomous navigation and intelligent delivery of medical resources within a hospital setting. The system utilizes the ESP32 microcontroller, Blynk IoT platform, and A* pathfinding algorithm for real-time decision-making and route optimization

3.1 Hardware Components

1. **ESP32**: The main processing unit that handles communication, navigation logic, and motor control.
2. **DC Motors** : Enables movement of the trolley in forward, left, and right directions.
3. **L298N Motor Driver**: Interfaces the ESP32 with the DC motors for bi-directional control.
4. **Push Button**: Used to trigger room visit routines manually.
5. **Power Supply**: Battery pack to supply power to all components.

3.2 Software Architecture

The software was developed using the Arduino IDE, and the system connects to Wi-Fi using the ESP32 to communicate with the Blynk IoT application. The code is structured to manage user input, navigation, and motor control.

a. **Room Mapping** - Room numbers (e.g., 101, 102, 103) are mapped to specific coordinates. - A request to visit a room translates to navigating to its corresponding waypoint.

b. **Blynk Integration** - Room visit requests can be sent via the Blynk mobile application using virtual pins (V0, V1, V3). - Each request includes a priority, allowing the system to decide the sequence based on urgency (e.g., 1 - Medicine, 2 - Food, 3 - Laundry).

c. **A* Pathfinding Algorithm** - A heuristic based search algorithm (Manhattan Distance) is used to determine the shortest path between two waypoints. - Ensures efficient and dynamic route planning, even if the path layout changes in future upgrades.

d. **Navigation and Motor Control**- Based on path directions, the robot performs:- 1. Forward movement 2. Left or right turns 3. 180° turns when necessary 4. Timing delays are calibrated to match the robot's mechanical turning angles and speed.

e. **LED Alert Mechanism** - On reaching each room, the onboard LED blinks five times to indicate task completion.

f. **Button Trigger Routine** - When the onboard button is pressed, the robot sequentially visits all rooms (pre-defined order). - This simulates a routine check or delivery process initiated by hospital staff.

3.3 Execution Flow

1. **Initialization**: System connects to Wi-Fi and sets up I/O pins.

2. **Request Window**: For the first 10 seconds after booting, Blynk app can send room requests.

3. **Request Handling**:Once the window closes, the robot sorts all received requests based on priority.It then navigates to each room accordingly.

4. **Manual Operation**: If the button is pressed, all rooms are visited in order irrespective of priority.

3.4 Block Diagram

The given block diagram represents the functional architecture of the Automated Nursing Assistance system, which is based on an ESP32 microcontroller. The system integrates multiple components, including sensors, motors, and a buzzer, to ensure efficient movement and autonomous navigation within a hospital setting.

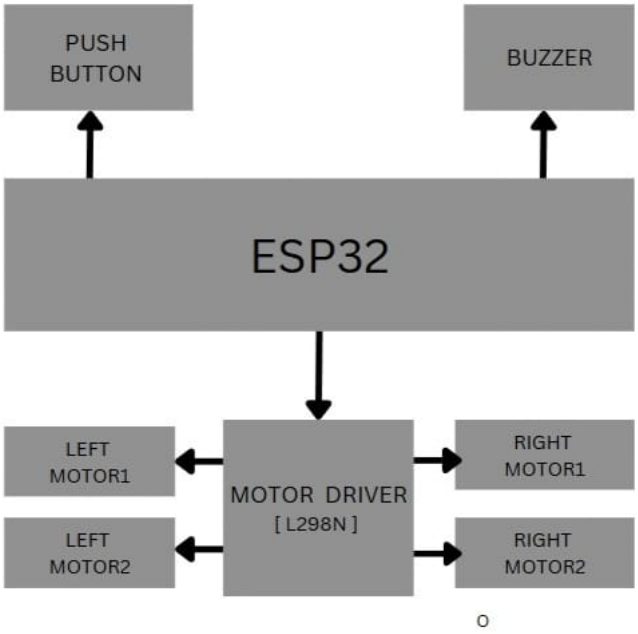

Figure 3.1: Block Diagram Of the Proposed Idea

3.5 Circuit Diagram

The circuit for the Automated Nursing Assistant is built around an ESP32 microcontroller, which serves as the brain of the robot. For movement, the ESP32 is connected to an L298N motor driver module, which controls four DC motors mounted on the robot's wheels. The motor driver receives direction and speed signals from the ESP32 to move the robot forward, backward, or make turns. A battery pack supplies power to the motors via the L298N, while the ESP32 is powered separately or through a voltage regulator.

To alert the user upon reaching a room, a buzzer and an LED are connected to the ESP32's digital output pins. The buzzer beeps and the LED blinks as a notification

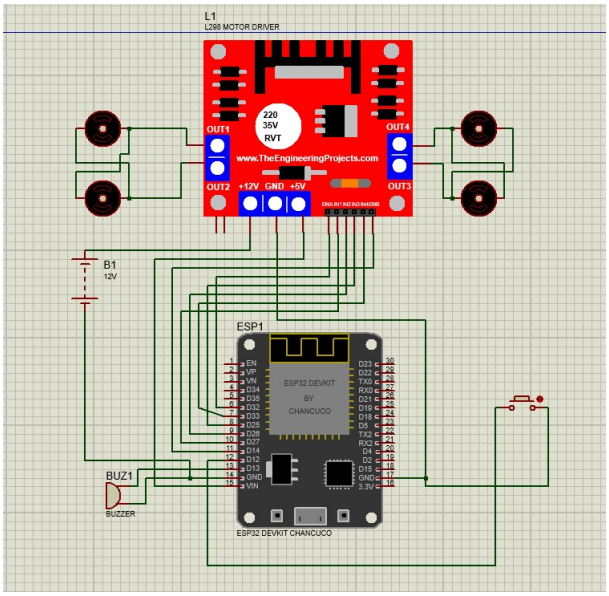

Figure 3.2: Schematic of the Circuit

signal. A pushbutton is also connected to a GPIO pin to enable manual control or override mode, which when pressed, triggers a predefined sequence of movements.

Wireless control is managed through Wi-Fi, using the Blynk IoT platform. The ESP32 connects to the internet and listens for requests via the Blynk app, which allows the user to choose the room number and the type of service (medicine, food, or laundry).

3.6 Working of the Circuit

The ESP32-based autonomous medical trolley is designed to transport medical supplies, food, and laundry to hospital rooms efficiently and it acts as a nursing assistant. It operates in three modes:

- Routine Mode - Move to each room in a serial order to deliver supplies at scheduled times on pressing the push button.
- Directed Mode - Moves to a specific room to deliver requested supplies.
- Priority Mode - Responds to multiple patient needs and deliver the supplies based on prioritizing the requests.

The system utilizes Wi-Fi for remote control, an L298N motor driver for movement, and an A star pathfinding algorithm to navigate waypoints in the hospital. The trolley moves using DC motors, and upon reaching a room, it signals arrival by blinking an LED and sounding a buzzer. Navigation is handled by the A star algorithm, which calculates the optimal route to each destination. The trolley can move forward, turn left, or turn right, adjusting its direction based on computed paths. It stops at each assigned room to deliver supplies and automatically returns to the control room once deliveries are complete.The queuing mechanism operates by collecting multiple requests within a fixed time window, allowing them to be stored in a queue along with their priority levels. Once the collection period ends, the requests are sorted based on priority, ensuring that higher-priority tasks are processed first. The system then executes each request sequentially, determining the optimal path for execution. After completing all tasks, it resets the queue and restarts the process for the next set of requests. This approach ensures efficient batch processing of multiple tasks but does not allow dynamic adjustments once execution begins. The system ensures efficient, priority-based deliveries with minimal human intervention, improving workflow in hospital environments.

Chapter 4

Components Used

4.1 ESP32 - Espressif System Processor 32-bit.

The ESP32 is a powerful dual-core microcontroller with Wi-Fi and Bluetooth capabilities, designed for high-performance IoT, automation, and real-time embedded systems. Operating at 240 MHz with 520 KB of SRAM, it supports external memory for handling multiple tasks efficiently. The ESP32 features 36 GPIO (General-Purpose Input/Output) pins that can function as digital I/O, PWM (Pulse Width Modulation) controllers, and capacitive touch sensors. Additionally, it includes 18 ADC (Analog-to-Digital Converter) channels for analog sensor readings and two DAC (Digital-to-Analog Converter) pins for analog signal generation.

The device supports multiple communication interfaces such as I2C, SPI, UART, and I2S, making it compatible with various external components like sensors, displays, and motor drivers. It also includes hardware-based cryptographic acceleration for secure data transmission, enhancing its suitability for secure IoT applications. With built-in power-saving features, the ESP32 is ideal for battery-powered and energy-efficient solutions.

By integrating sensor-based navigation, real-time data transmission, and wireless connectivity, the ESP32 enhances the efficiency of automation across various industries. Its versatility and scalability make it a key component in modern IoT applications, enabling smart solutions for industrial automation, home automation, and environmental monitoring.

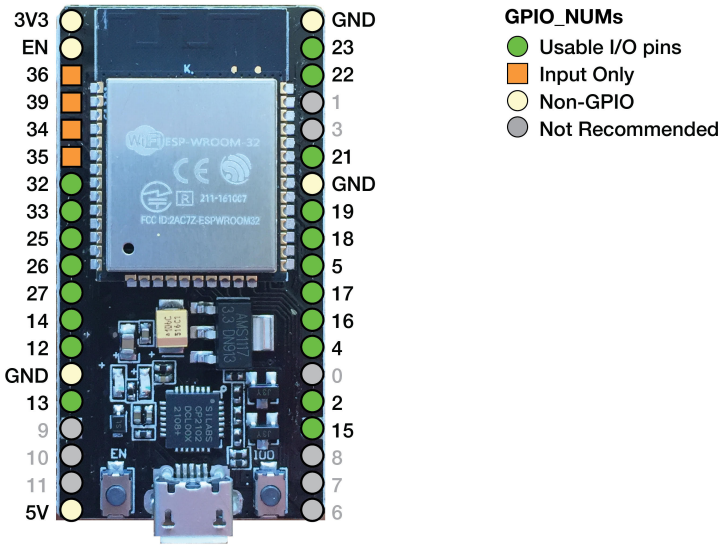

Figure 4.1: Pin Diagram of ESP32

4.2 L298N MOTOR

The L298N motor driver is a dual H-bridge driver IC capable of controlling two DC motors simultaneously, making it a versatile component for robotics and automation projects. Operating within a voltage range of 5V to 35V and handling up to 2A per channel, it is well-suited for medium-power motors. The module includes an integrated heat sink to prevent overheating and features onboard 5V regulators, allowing it to power low-voltage logic circuits. It supports Pulse Width Modulation (PWM) control, enabling efficient speed and direction control of motors.

In our Automated Nursing Assistance System, the L298N driver plays a crucial role in motor control, facilitating precise movement and path memorization. It receives control signals from the microcontroller, which adjusts the speed and direction of the trolley's wheels. This ensures smooth operation within hospital environments, allowing the trolley to autonomously navigate, deliver medical supplies, and return to its starting position efficiently.

Integrating the L298N motor driver with the ESP32 microcontroller enhances the system's efficiency. The ESP32 sends PWM signals to the L298N, dictating motor behavior based on real-time data and pre-programmed routes. This integration allows for dynamic adjustments in the trolley's movement, accommodating changes in the environment or task priorities. The L298N's ability to control two motors

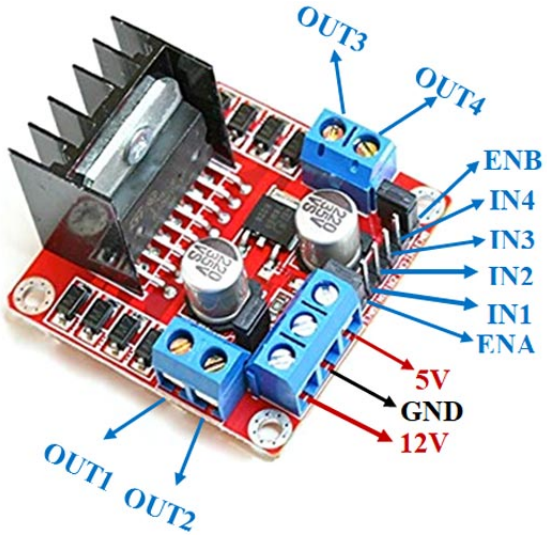

Figure 4.2: Pin Diagram of L298N

independently ensures that the trolley can execute complex maneuvers, such as turning in place or navigating tight corridors, which are common in hospital settings.

The L298N's robust design includes features like thermal shutdown and overcurrent protection, ensuring reliable operation in demanding environments. These safety mechanisms are vital in a healthcare setting, where equipment failure can disrupt critical operations. Furthermore, the module's compatibility with various motor types and its straightforward interfacing with microcontrollers make it a cost-effective and reliable choice for our project.

4.3 Gear Motor And Wheels

In the Automated Nursing Assistance System, gear motors play a pivotal role in ensuring precise movement and control, which are essential for the autonomous navigation of the medical trolley within hospital environments. By integrating gear motors, the system benefits from enhanced torque and controlled speed, enabling the trolley to transport medical supplies efficiently and safely.

Key Features of Gear Motors in the Project:

- High Torque Output: The inclusion of a gearbox allows the motor to deliver increased torque, facilitating the movement of the trolley even when carrying heavy medical supplies. This capability is crucial for maintaining consistent performance across various floor surfaces and inclines within the hospital.

- Variable Speed Control: The gear mechanism enables precise adjustment of the trolley's speed, allowing it to navigate crowded hospital corridors safely. By controlling the speed, the system can ensure timely deliveries without compromising safety.

- Bidirectional Movement: Gear motors support both clockwise and counter-clockwise rotation, providing the trolley with the ability to move forward and backward as needed. This feature enhances maneuverability, allowing the trolley to navigate tight spaces and make precise turns within the hospital.

- Compact Design: Despite their robust performance, gear motors maintain a compact form factor. This design consideration is vital for integrating the motors within the trolley's structure without adding excessive bulk, preserving the trolley's ability to navigate through standard hospital doorways and corridors.

Integrating gear motors with the ESP32 microcontroller and the L298N motor driver creates a cohesive system where the microcontroller processes navigation commands and transmits control signals to the motor driver. The motor driver then regulates the power supplied to the gear motors, dictating the trolley's speed and direction. This integration ensures that the trolley can execute complex navigation tasks, such as following predetermined paths, avoiding obstacles, and adjusting its route dynamically in response to environmental changes.

Figure 4.3: Pin Diagram of Gear Motor Wheels

These motors provide the necessary torque and control required for transporting medical supplies and equipment efficiently within hospital environments.By leveraging the advantages of gear motors, the Automated Nursing Assistant System ensures reliable and efficient transportation of medical supplies, contributing to improved operational efficiency and patient care within healthcare facilities.

4.4 Buzzer

A buzzer is an electrical component that produces sound when an electric current passes through it. It typically consists of a coil of wire wrapped around a magnet and a vibrating diaphragm. When an alternating current (AC) or pulsating direct current (DC) is applied to the coil, it creates a magnetic field that attracts and repels the diaphragm, causing it to vibrate and produce sound waves. Key features and uses of buzzers include:

- Audible Alert: Buzzers are commonly used as audible indicators or alarms to alert users of specific events or conditions. They produce a distinct buzzing or beeping sound that is easily recognizable.
- Compact Size: Buzzers are usually small and lightweight, making them suitable for integration into various electronic devices and systems without taking up much space.
- Simple Operation: Buzzers are easy to use and require minimal external components for operation. They can be activated by applying a voltage or current to the terminals, making them suitable for simple on/off control.

- Wide Range of Frequencies: Buzzers are available in various frequencies, allowing for customization of the sound output to suit different applications and preferences.
- Versatility: Buzzers find applications in a wide range of industries and devices, including alarm systems, doorbells, electronic games, appliances, automotive indicators and industrial equipment.

.

Figure 4.4: Buzzer

.

4.5 Push Button

A push button is a momentary switch that completes an electrical circuit when pressed and returns to its default state when released. It is widely used in embedded systems for user input, control mechanisms, and triggering events.

When integrated into an electronic circuit, the push button can function in two common configurations: pull-up mode and pull-down mode. In pull-up mode, the button is connected to a high voltage (logic HIGH) by default and switches to a low voltage (logic LOW) when pressed. Conversely, in pull-down mode, the default state is low (logic LOW), and pressing the button connects it to a higher voltage (logic HIGH). Microcontrollers, such as the ESP32, continuously monitor the button's state to detect user input and trigger specific actions.

In the Nursing assistant, the push button is connected to one of the GPIO (General-Purpose Input/Output) pins of the ESP32 microcontroller. By default, the button remains unpressed, meaning the circuit is open, and no current flows through it. When

a user presses the button, the circuit closes, allowing current to pass, which the ESP32 detects as an input signal.

The push button in the Nursing assistant operates in pull-up mode, ensuring a stable default state (logic HIGH). When pressed, it switches to logic LOW, prompting the ESP32 to execute a predefined function. This mechanism can be used to start or stop the robot, change its mode, or trigger an emergency stop, depending on the programmed functionality.

Figure 4.5: Push Button

Chapter 5

Inferences

5.1 Hardware Implementation

Figure 5.1: Hardware implemented

Chapter 6

Conclusion

The Automated Nursing Assistance System developed in this project provides an efficient and intelligent solution for automating medical supply delivery within healthcare facilities. By leveraging advanced algorithms and centralized control, this system significantly enhances hospital workflow, ensuring timely delivery of essential items such as Medicine, Food, and Laundry.

A key feature of this system is the implementation of the A* Algorithm for path planning, which enables the trolley to navigate through the healthcare environment in the most efficient manner. Unlike conventional approaches that rely on IR or ultrasonic sensors, the A* algorithm ensures accurate pathfinding by calculating the shortest route in real time. This optimization minimizes delays and ensures smooth operation in complex hospital layouts.

The entire system is controlled by a single ESP microcontroller unit, acting as the central processing hub. This reduces hardware complexity while maintaining high efficiency and reliability. The ESP unit is responsible for processing navigation commands, managing service requests, and ensuring seamless communication with the Blynk IoT platform. Through Blynk, hospital staff can remotely assign delivery tasks by selecting from predefined priority categories(1.Medicine, 2.Food, 3.Laundry). This priority system ensures that critical medical supplies are delivered first, followed by food and laundry, optimizing service efficiency.

To enhance user interaction and safety, the system includes a buzzer and LED indicators that provide real-time status updates. The buzzer alerts staff upon the

trolley's arrival, while LED signals indicate task execution stages, making the system intuitive and easy to monitor.

The successful implementation of this project demonstrates the potential of IoT-based automation in revolutionizing healthcare services. By reducing manual workload, minimizing human error, and improving response time, the Automated Nursing Assistance System paves the way for a more efficient and technologically advanced hospital environment. Future improvements could involve integrating additional sensors for enhanced obstacle detection and expanding the system to support multi-floor operations using elevators and RFID-based room recognition. With further advancements, this system can serve as a scalable solution for smart healthcare automation, contributing to better patient care and hospital management.

Bibliography

1. C. Chen, L. Wang, and Y. Zhang, “Path planning for mobile robots in hospital environments using A* algorithm,”*Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pp. 1234–1240, 2020. [DOI: 10.1109/ICRA40945.2020.9196690] https://doi.org/10.1109/ICRA40945.2020.9196690

2. R. Sharma and J. Lee, “Autonomous medical trolley navigation using WiFi fingerprinting and IoT,” *J. Med. Syst.*, vol. 45, no. 6, p. 78, 2021. [DOI: 10.1007/s10916-021-01755-2] https://doi.org/10.1007/s10916-021-01755-2

3. A. Patel et al., “Smart autonomous medical trolley with RFID and robotic manipulation,” *IEEE Trans. Med. Robot. Bionics*, vol. 4, no. 2, pp. 512–525, 2022. [DOI: 10.1109/TMRB.2022.3167890] https://doi.org/10.1109/TMRB.2022.3167890

Appendix A

A.1 Program

ESP Code for the proposed idea

```
#define BLYNK_TEMPLATE_ID "TMPL36pjgWG6R"
#define BLYNK_TEMPLATE_NAME "Nurseasst"
#define BLYNK_AUTH_TOKEN "2uR_rp-GbiW4DAA4jdnS8fIzq-
    _onyVD"

#include <Arduino.h>
#include <vector>
#include <queue>
#include <map>
#include <ESP32Servo.h>
#include <WiFi.h>
#include <BlynkSimpleEsp32.h>

char auth[] = BLYNK_AUTH_TOKEN;
char ssid[] = "Galaxy";
char pass[] = "Rocky1374";

#define MOTOR1_IN1   27
#define MOTOR1_IN2   26
```

```
#define MOTOR2_IN1  25
#define MOTOR2_IN2  33
#define LED_PIN     2 // You can change this if needed
#define BUTTON_PIN 12
#define buzzerPin  32
bool buttonTriggered = false;



struct Waypoint {
    int x, y;
    bool operator==(const Waypoint& other) const {
        return x == other.x && y == other.y;
    }
    bool operator<(const Waypoint& other) const {
        return std::tie(x, y) < std::tie(other.x, other.y
           );
    }
};

struct RoomRequest {
    int room_no;
    int priority;
};

// Waypoints

std::vector<Waypoint> waypoints = {
    {0, 0}, {2, 0}, {4, 0}, // Path along X-axis
    {2, 2}, {4, 2},         // Middle waypoints
    {4, 4}                  // Final destination
```

```
};


std::map<int, Waypoint> room_map = {
    {101, {4, 4}},
    {102, {4, 2}},
    {103, {2, 2}},
    {0, {0, 0}} // start
};

std::vector<RoomRequest> room_queue;
bool collectingRequests = true;
unsigned long requestStartTime = 0;
const unsigned long requestDuration = 10000;

// Motor control
void moveForward() {
    digitalWrite(MOTOR1_IN1, HIGH);
    digitalWrite(MOTOR1_IN2, LOW);
    digitalWrite(MOTOR2_IN1, HIGH);
    digitalWrite(MOTOR2_IN2, LOW);
}
void turnLeft() {
    digitalWrite(MOTOR1_IN1, LOW);
    digitalWrite(MOTOR1_IN2, HIGH);
    digitalWrite(MOTOR2_IN1, HIGH);
    digitalWrite(MOTOR2_IN2, LOW);
}
void turnRight() {
    digitalWrite(MOTOR1_IN1, HIGH);
    digitalWrite(MOTOR1_IN2, LOW);
```

```
    digitalWrite(MOTOR2_IN1, LOW);
    digitalWrite(MOTOR2_IN2, HIGH);
}
void stopRobot() {
    digitalWrite(MOTOR1_IN1, LOW);
    digitalWrite(MOTOR1_IN2, LOW);
    digitalWrite(MOTOR2_IN1, LOW);
    digitalWrite(MOTOR2_IN2, LOW);
}

//LED Blink Function
void blinkLED() {
    for (int i = 0; i < 5; i++) {
        digitalWrite(LED_PIN, HIGH);
        delay(250);
        digitalWrite(LED_PIN, LOW);
        delay(250);
    }
}
void beepBeep() {
  // First beep
  digitalWrite(buzzerPin, HIGH);
  delay(1000); // 2 seconds
  digitalWrite(buzzerPin, LOW);
  delay(500);  // Short gap between beeps

  // Second beep
  digitalWrite(buzzerPin, HIGH);
  delay(1000); // 2 seconds
  digitalWrite(buzzerPin, LOW);
}
```

```
// A* Heuristic
int heuristic(Waypoint a, Waypoint b) {
    return abs(a.x - b.x) + abs(a.y - b.y);
}

std::vector<Waypoint> getNeighbors(Waypoint current) {
    std::vector<Waypoint> neighbors;
    for (Waypoint& wp : waypoints) {
        if ((abs(wp.x - current.x) == 2 && wp.y ==
           current.y) ||
            (abs(wp.y - current.y) == 2 && wp.x ==
               current.x)) {
            neighbors.push_back(wp);
        }
    }
    return neighbors;
}

std::vector<Waypoint> aStar(Waypoint start, Waypoint goal
   ) {
    std::priority_queue<std::pair<int, Waypoint>, std::
       vector<std::pair<int, Waypoint>>, std::greater<>>
       openSet;
    std::map<Waypoint, Waypoint> cameFrom;
    std::map<Waypoint, int> gScore;

    for (auto& wp : waypoints) gScore[wp] = INT_MAX;
    gScore[start] = 0;

    openSet.emplace(heuristic(start, goal), start);
```

```
    while (!openSet.empty()) {
        Waypoint current = openSet.top().second;
        openSet.pop();

        if (current == goal) {
            std::vector<Waypoint> path;
            while (!(current == start)) {
                path.push_back(current);
                current = cameFrom[current];
            }
            path.push_back(start);
            std::reverse(path.begin(), path.end());
            return path;
        }

        for (Waypoint neighbor : getNeighbors(current)) {
            int tentativeGScore = gScore[current] + 1;
            if (tentativeGScore < gScore[neighbor]) {
                cameFrom[neighbor] = current;
                gScore[neighbor] = tentativeGScore;
                int fScore = tentativeGScore + heuristic(
                    neighbor, goal);
                openSet.emplace(fScore, neighbor);
            }
        }
    }
    return {};
}

void moveThroughWaypoints(std::vector<Waypoint> path) {
```

```
int currentDirection = 0; // 0: right, 1: down, 2:
    left, 3: up

for (size_t i = 1; i < path.size(); i++) {
    Waypoint prev = path[i - 1];
    Waypoint curr = path[i];

    int dx = curr.x - prev.x;
    int dy = curr.y - prev.y;

    int newDirection;
    if (dx == 2) newDirection = 0;        // Right
    else if (dx ==-2) newDirection = 2; // Left
    else if (dy == 2) newDirection = 1;  // Down
    else if (dy == -2) newDirection = 3; // Up
    else continue;

    int turn = (newDirection - currentDirection + 4)
       % 4;

        if (prev.x == 4 && prev.y == 4 && curr.x == 4
            && curr.y == 2) {
        Serial.println("Turning 180   from (4,4) to
           (4,2)");
        turn = 2;
    }

    //Ensure left turn at (4,2)     (2,2)
    if (prev.x == 4 && prev.y == 2 && curr.x == 2 &&
       curr.y == 2) {
```

```
        Serial.println("Turning LEFT at (4,2) to
            (2,2)");
        turn = 3;
    }

    //Ensure right turn at (2,2)      (2,0)
    if (prev.x == 2 && prev.y == 2 && curr.x == 2 &&
        curr.y == 0) {
        Serial.println("Turning RIGHT at (2,2) to
            (2,0)");
        turn = 1;
    }

    // Ensure left turn at (2,0)      (0,0)
    if (prev.x == 2 && prev.y == 0 && curr.x == 0 &&
        curr.y == 0) {
        Serial.println("Turning LEFT at (2,0) to
            (0,0)");
        turn = 3;
    }



    if (turn == 1) {
        Serial.println("Turning right");
        turnRight();
        delay(500); // Adjust based on your robots
            turn time
    } else if (turn == 3) {
        Serial.println("Turning left");
        turnLeft();
        delay(500);
```

```
        } else if (turn == 2) {
            Serial.println("Turning 180");
            turnRight(); delay(800);
            turnRight(); delay(500);
        }
        Serial.printf("Turn: %d (New Direction: %d,
            Current Direction: %d)\n", turn, newDirection,
            currentDirection);

        currentDirection = newDirection;

        Serial.printf("Moving to: (%d, %d)\n", curr.x,
            curr.y);
        moveForward();
        delay(800);  // Adjust based on your robots
            step distance
        stopRobot();
        delay(200);
    }
}


// Blynk priorities
BLYNK_WRITE(V3) { room_queue.push_back({101, param.asInt
    ()}); }
BLYNK_WRITE(V0) { room_queue.push_back({102, param.asInt
    ()}); }
BLYNK_WRITE(V1) { room_queue.push_back({103, param.asInt
    ()}); }

void setup() {
```

```
    Serial.begin(115200);
    Blynk.begin(auth, ssid, pass);
    pinMode(MOTOR1_IN1, OUTPUT);
    pinMode(MOTOR1_IN2, OUTPUT);
    pinMode(MOTOR2_IN1, OUTPUT);
    pinMode(MOTOR2_IN2, OUTPUT);
    pinMode(buzzerPin, OUTPUT);
    pinMode(LED_PIN, OUTPUT); //Initialize LED pin
    pinMode(BUTTON_PIN, INPUT_PULLUP); // assuming
       pushbutton connects to GND when pressed

    digitalWrite(LED_PIN, LOW);

    requestStartTime = millis();
}

void loop() {
    Blynk.run();

    // Check for button press
    if (digitalRead(BUTTON_PIN) == LOW && !
       buttonTriggered) {
    buttonTriggered = true;
    Serial.println("Button pressed - Visiting all rooms
       in order...");

    Waypoint start = {0, 0};
    std::vector<int> roomOrder = {101, 102, 103};

    for (int room : roomOrder) {
        if (room_map.find(room) != room_map.end()) {
```

```
            Waypoint destination = room_map[room];
            Serial.printf("Navigating to Room %d at (%d,
               %d)\n", room, destination.x, destination.y
               );
            auto path = aStar(start, destination);
            moveThroughWaypoints(path);
            stopRobot();
            blinkLED();
            beepBeep();
            start = destination;
        }
    }

    //Return to (0, 0) after visiting all rooms
    if (!(start == Waypoint{0, 0})) {
        Serial.println("Returning to starting point (0,0)
           ...");
        auto returnPath = aStar(start, {0, 0});
        moveThroughWaypoints(returnPath);
        stopRobot();
        blinkLED();
        beepBeep();
    }

    Serial.println("Completed room visit from button
       press.");
    buttonTriggered = false;
    delay(1000);  // debounce time
}
```

```
// Handle Blynk request window
if (millis() - requestStartTime < requestDuration) {
    collectingRequests = true;
} else {
    collectingRequests = false;
}

if (!collectingRequests && !room_queue.empty()) {
    Serial.println("Processing queued requests with
       priority...");

    std::sort(room_queue.begin(), room_queue.end(),
       [](RoomRequest a, RoomRequest b) {
        return a.priority < b.priority;
    });

    Waypoint start = {0, 0};

    for (RoomRequest request : room_queue) {
        if (room_map.find(request.room_no) !=
           room_map.end()) {
            Waypoint destination = room_map[request.
               room_no];
            Serial.printf("Navigating to Room %d (
               Priority %d) at (%d, %d)\n", request.
               room_no, request.priority, destination
               .x, destination.y);
            auto path = aStar(start, destination);
            moveThroughWaypoints(path);
            stopRobot();
            blinkLED();
```

```
                beepBeep();
                start = destination;
            }
        }
                // Return to start
        if (!(start == Waypoint{0, 0})) {
            Serial.println("Returning to starting point
               (0,0) after Blynk requests...");
            auto returnPath = aStar(start, {0, 0});
            moveThroughWaypoints(returnPath);
            stopRobot();
            blinkLED();
            beepBeep();
            start={0,0};//update position
        }


        room_queue.clear();
        requestStartTime = millis(); // reset
        collectingRequests = true;
    }
}
```