Design and Implementation of a Wall-Mapping Autonomous Robot with Bluetooth Mobile Connectivity

Suhani Agarwal 22116090
Department of Electronics and Communication Engineering
Indian Institute of Technology, Roorkee
Email: suhani_a@ece.iitr.ac.in

Prajwal Sastry 22116069
Department of Electronics and Communication Engineering
Indian Institute of Technology, Roorkee
Email: prajwal_s@ece.iitr.ac.in

Aditya Vinay 22123002
Department of Electronics and Communication Engineering
Indian Institute of Technology, Roorkee
Email: aditya_v@ece.iitr.ac.in

***Abstract*—This paper presents the design and implementation of a real-time wall-mapping and control system. The system employs a bot equipped with sensors for distance and obstacle detection, along with an Android application that communicates via an HC-05 Bluetooth module. The app provides both automatic navigation for mapping and manual control through an intuitive interface. Real-time mapping is visualized using MPAndroidChart, integrating embedded systems, Bluetooth communication, and Android development seamlessly.**

***Index Terms*—UART, Android Studio, HC-05 Bluetooth Communication, HC-SR04 Ultrasound Sensor, PID Controller, IR Sensor,Embedded Systems, Tiva C Series TM4C123G, PWM Control**

I. Introduction

The integration of embedded systems and Android applications provides robust solutions for robotics and automation. This project aims to design a wall-mapping and control system using a Tiva TM4C123GH6PM microcontroller-based bot. Equipped with sensors and Bluetooth communication, the bot collects real-time data that is visualized and controlled via an Android application. The system combines the capabilities of precise hardware design with an interactive software interface, enabling autonomous navigation and mapping in real-time. The block-diagram giving an overview of the project is shown in Figure 1

II. System Design

The system comprises two main components: hardware and software.

A. Hardware Modules

The bot hardware includes the following modules:

- **Sensors:**
  - HC-SR04 ultrasound sensor for distance measurement and wall-following.
  - IR sensor for detecting obstacles in front of the bot.
- **Microcontroller:** Tiva TM4C123GH6PM, responsible for processing sensor data, motor control, and Bluetooth communication.

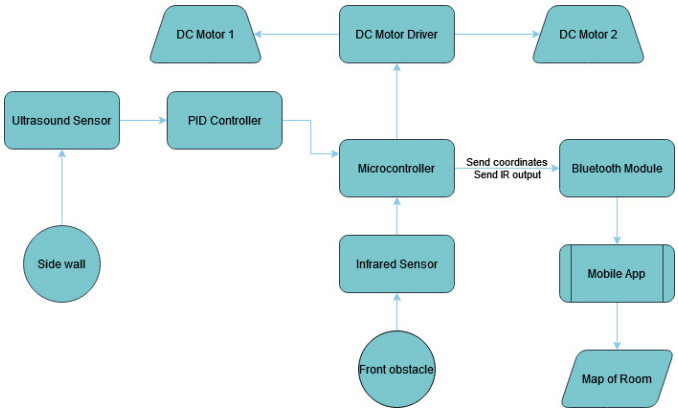

Fig. 1: Block Diagram for the bot

- **Motor Components:** DC gear motors for driving the bot and motor driver L293D IC.
- **Communication:** HC-05 Bluetooth module for real-time communication with the Android app.
- **Power Supply:**
  - 6V 4x AA battery pack to power sensors and the microcontroller.
  - 11V Li-Po battery pack for driving the motors.
- **Prototyping:** Breadboard for rapid assembly and interconnection of components.

B. Software Design

The Android application was developed using Android Studio, and its functionality includes real-time mapping and control.

III. Android App Design

The Android application is developed using Android Studio and provides an intuitive interface for controlling the bot and visualizing real-time wall mapping.

A. App Design

Upon launching the app, the user is first presented with a device selection menu, as shown in Figure 4. In this menu, the user can choose the Bluetooth device they wish to connect to. By selecting the HC-05 Bluetooth module, the app establishes communication with the bot, allowing it to control the bot and monitor its actions.

Once connected, the user can switch between two modes: manual and automatic.

In **manual mode**, the user is given control of the bot through a set of movement buttons that allow for directional commands. The movement buttons are clearly highlighted, indicating their active state. As the bot moves, the app dynamically updates the map, drawing the path based on the bot's movements. The user can navigate the bot manually within the room, and the map is continuously updated to reflect its position and the environment around it. An example of the bot tracing a U path in manual mode is shown in Figure 2

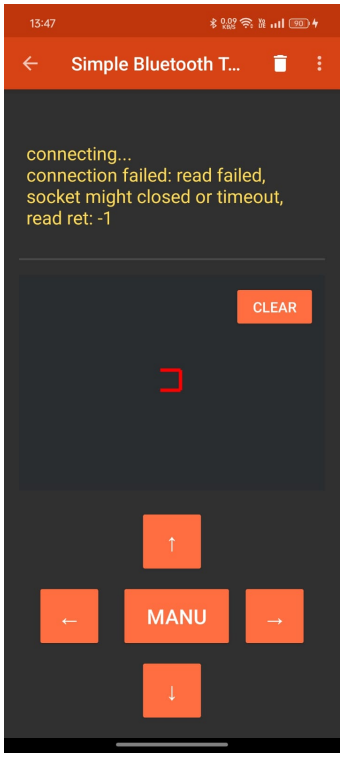

Fig. 2: Manual mode with the bot tracing a U-shape path on the screen. The map updates as the bot moves.

When the user presses the central button on the interface, the app switches to **automatic mode**. In automatic mode, the bot autonomously maintains a constant distance from the wall on its left side while following the wall to navigate. The bot adjusts its path and distance in real time based on sensor inputs from the HC-SR04 ultrasound sensor. The app continuously draws the map of the room according to the bot's path, providing a visual representation of the area it has covered. An example of the bot tracing the corner of a room is shown in Figure 3

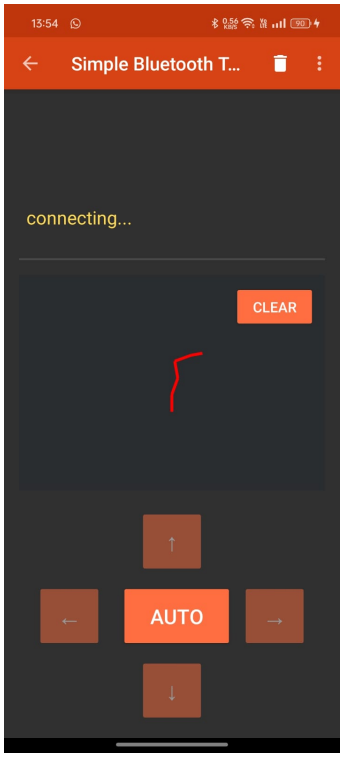

Fig. 3: Automatic mode showing the bot's mapped path in a corner of the room.

The automatic mode also allows the bot to follow the wall, adjusting its path to ensure the left side remains at a consistent distance from the wall. The real-time updates to the map provide a clear visualization of the bot's progress as it explores the environment.

B. Implementation of the App

The app's development was inspired by and builds upon the open-source project available at https://github.com/kai-morich/SimpleBluetoothTerminal.git. This repository provided the foundational Bluetooth communication functionality, which was then customized and expanded to fit the requirements of our bot control and mapping application.

The app implements two modes of operation—manual and automatic—each designed to interact with the bot differently based on the data exchanged via the HC-05 Bluetooth module.

1) Manual Mode: In manual mode, the user directly controls the bot's movement by clicking one of the directional buttons on the interface. Each button corresponds to a specific direction:

- **Up arrow**: Sends the character `'u'` for forward movement.
- **Down arrow**: Sends the character `'d'` for backward movement.
- **Right arrow**: Sends the character `'r'` for turning right.
- **Left arrow**: Sends the character `'l'` for turning left.

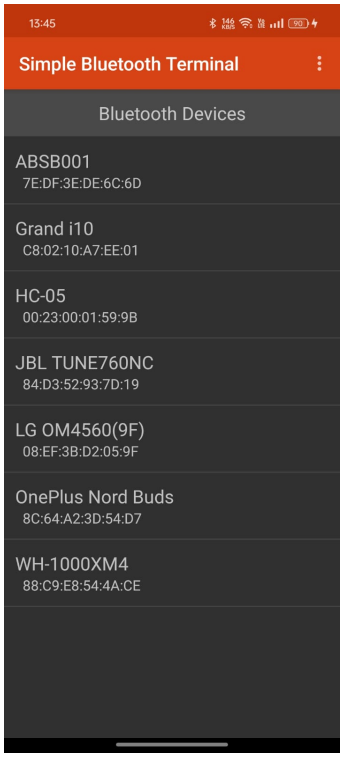

Fig. 4: Device selection menu in the Android app, where the user can choose the HC-05 Bluetooth module to connect to the bot.

These commands are transmitted via the HC-05 module to the bot. After executing the command, the bot responds with a string of the format:

`DISTANCE X TURN Y`

where:

- `X` represents the distance the bot traveled.
- `Y` represents the angle by which the bot turned during the movement.

This string is received by the app, parsed, and used to update the real-time map. The data is visualized using the MPAndroidChart library, dynamically tracing the bot's path on the screen.

2) Automatic Mode: In automatic mode, the bot operates autonomously by utilizing its onboard sensors, including the HC-SR04 ultrasound sensor and an IR sensor, along with a PID controller to maintain a constant distance from the wall on its left. During this mode, the bot continuously navigates the environment, adjusting its path to avoid obstacles and follow walls.

As the bot moves, it sends real-time updates to the app in the same `DISTANCE X TURN Y` format:

- `X` represents the distance traveled since the last update.
- `Y` indicates the correction angle applied to maintain its path.

The app parses these updates and plots them on the map using MPAndroidChart, creating a visual representation of the bot's exploration. This mode showcases the bot's ability to autonomously explore and map its surroundings while maintaining a consistent relationship with nearby walls.

By combining the interactive features of manual mode with the autonomous capabilities of automatic mode, the app provides a comprehensive tool for both controlling and visualizing the bot's behavior in real time.

IV. PID Controller

The **PID controller** (Proportional-Integral-Derivative) is a control loop mechanism used in the bot to maintain a constant distance from the wall during automatic wall mapping. It ensures smooth and precise wall-following behavior by calculating and adjusting the bot's motion based on the difference between the desired distance and the current measured distance.

- **Input Parameters:**
  - K_p: Proportional gain, it is the difference between the desired wall distance and the measured distance.

$$P_{\text{output}} = K_p \cdot e(t)$$

  - K_i: Integral gain, controls the accumulation of error over time.

$$I_{\text{output}} = K_i \cdot \int e(t)\, dt$$

  - K_d: Derivative gain, reduces overshoot by responding to error rate changes.

$$D_{\text{output}} = K_d \cdot \frac{de(t)}{dt}$$

  - `required_distance`: Desired distance from the wall.

  where $e(t)$ is the error difference.

- **Control Output:** Combines all three terms to calculate the correction.

$$\text{Control Output} = P_{\text{output}} + I_{\text{output}} + D_{\text{output}}$$

- **Feedback Mechanism:**
  - Real-time distance measurements are obtained using the ultrasound sensor.
  - Errors are continuously updated based on the difference between the `current_distance` and `required_distance`.
- **Role in the Bot's Functionality:**
  - Ensures the bot adjusts its trajectory to remain parallel to the wall.
  - Dynamically corrects the speed and turning angle based on the control output.

TABLE I: PID Controller Parameters used

K_p	K_i	K_d	**Required Distance**
1.0	0.0	0.5	20.0 cm

V. Ultrasound Sensor

The bot uses the HC-SR04 ultrasonic sensor to measure distances. It plays a critical role in both automatic wall mapping and maintaining a constant distance from the wall. The sensor is attached to the left-side of the bot, providing real time distance measurements.

A. Working Principle

The HC-SR04 sensor works by emitting an ultrasonic pulse from its **Trigger** pin. The pulse travels through the air, and when it encounters an obstacle, it reflects to the sensor. The sensor measures the time taken for the echo to return using its **Echo** pin. The distance is calculated using the formula:

$$\text{Distance (cm)} = \frac{\text{Pulse Duration (microseconds)} \times 0.0343}{2}$$

B. Implementation in the Bot

- **Ports Used:**
  - **Trigger (GPIOB, Pin 7):** Outputs a high pulse to initiate the ultrasonic burst.
  - **Echo (GPIOB, Pin 6):** Reads the reflected signal to calculate distance.
- **SysTick Timer Usage:**
  - The SysTick timer, running on a 50 MHz system clock, is used to measure the high time of the Echo signal.
  - When the Echo signal goes high, the start time is recorded using `SysTick->VAL`.
  - When the Echo signal goes low, the end time is recorded, and the pulse duration is calculated as the difference between the start and end times.

C. Relation with the PID Controller

The distance measured by the `distance_side()` function is directly used as input for the PID controller. This allows the PID controller to compute errors and generate control outputs.

VI. UART Communication for Bluetooth Module

This section details the implementation of UART communication on the TIVA TM4C123GH6PM microcontroller for the HC-05 Bluetooth module, enabling interaction for sending and receiving data. The Universal Asynchronous Receiver-Transmitter (UART) protocol facilitates serial communication, which is essential for transmitting characters and strings.

A. Initialization of UART

The `UART_init()` function configures the UART1 module and prepares the GPIO pins for transmission (TX) and reception (RX). The following steps are executed in the function:

- The system clock for the UART1 module and Port B (GPIO) is enabled.
- Alternate functions for pins PB0 and PB1 are set to correspond to UART1 RX and TX, respectively.
- Digital functionality for PB0 and PB1 is enabled.
- UART1 is temporarily disabled to configure its settings.
- The baud rate is set to 9600 using integer and fractional baud rate divisors.
- The UART line control register is configured for 8-bit data, 1 stop bit, and no parity.
- The system clock is selected as the clock source for UART1.
- Finally, UART1 is enabled for transmission and reception.

B. Receiving Characters via UART

The function `readChar()` reads a single character from the UART1 module. It checks whether the receive FIFO is empty:

- If the FIFO is empty, the function returns a null character (`\0`).
- Otherwise, it returns the character from the data register.

C. Transmitting Characters via UART

The `printChar()` function is used to transmit a single character via UART1. It ensures that the transmit FIFO is not full before writing the character to the data register.

D. Transmitting Strings via UART

The `printString()` function extends the functionality of `printChar()` by enabling the transmission of strings. It iterates through each character in the string and sends it using `printChar()`.

VII. Motor Control

The motor control system enables the bot to move in different directions, forward, backward, left, and right, based on commands from the PID controller (in automatic mode) or Bluetooth commands (in manual mode). The system is driven by two DC motors controlled using the L293D motor driver IC. A schematic of how the motors and the IC were circuited, is shown in Figure 5

A. Working Principle

The L293D is a dual H-bridge motor driver IC that allows independent control of two DC motors. It operates by toggling its input pins to control the direction and speed of each motor.

B. System Components and Connections

- **Motor Driver IC: L293D**
  - **Inputs:**
    - **Input 1, 2 (Right Motor):** Controlled via GPIOA pins 2 and 3.
    - **Input 3, 4 (Left Motor):** Controlled via GPIOA pins 4 and 5.
  - **Outputs:**
    - Connected to the two DC motors to enable forward or reverse motion.

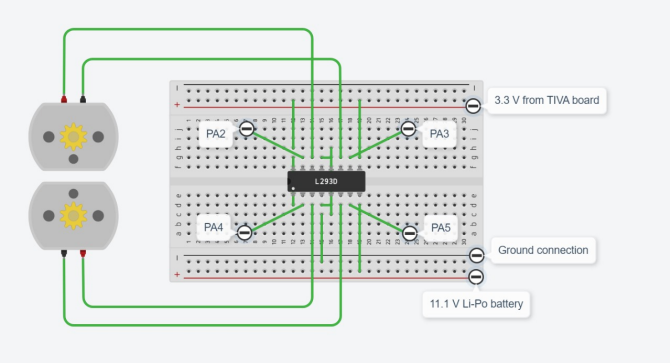

Fig. 5: Schematic of motor circuit

  - **Enable Pins:** Set high for each motor to allow control (not software-configured as always enabled).
- **Motors:**
  - Two DC motors, connected to the outputs of the L293D, drive the bot's wheels.
- **Ports Used:**
  - **GPIOA (Pins 2, 3, 4, 5):** Controls motor directions for both motors.

C. Movement Control Logic

The bot moves in different directions by toggling the GPIO pins controlling the L293D inputs:

- **Forward:** - Right motor input pins: GPIOA (3 = HIGH, 2 = LOW). - Left motor input pins: GPIOA (5 = HIGH, 4 = LOW).
- **Backward:** - Right motor input pins: GPIOA (3 = LOW, 2 = HIGH). - Left motor input pins: GPIOA (5 = LOW, 4 = HIGH).
- **Turn Left:** - Right motor input pins: GPIOA (3 = HIGH, 2 = LOW). - Left motor input pins: GPIOA (5 = LOW, 4 = HIGH) (left motor in reverse).
- **Turn Right:** - Right motor input pins: GPIOA (2 = LOW, 3 = HIGH) (right motor in reverse). - Left motor input pins: GPIOA (4 = HIGH, 5 = LOW).

D. Integration with Control Logic

The `move_boombot()` function implements motor control by setting appropriate GPIO states based on the speed and turn values calculated by the PID controller or commands from the Bluetooth module.

E. Sensor Feedback and Delay

- The bot moves for 300 ms using the `Delay_ms()` function under the overall movement function with a duty cycle in the delay dependent on the speed specified by the function. After each movement operation the next reading from the ultrasonic sensor is taken. This ensures sufficient movement for new readings to reflect a meaningful change in position.

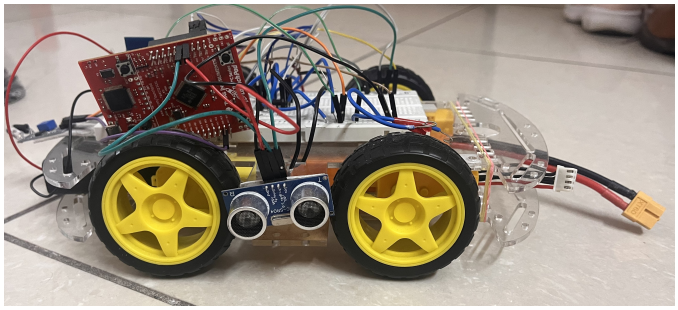

Fig. 6: Final prototype

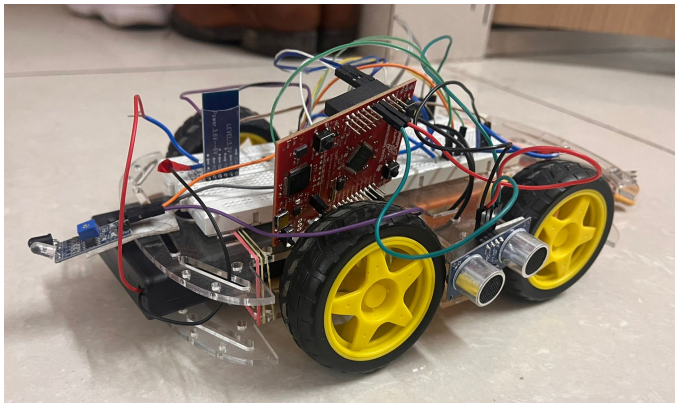

Fig. 7: Front view of the bot

RESULTS AND DISCUSSION

Finally, the bot was constructed by securely attaching the Tiva board, ultrasonic and infrared sensors, Bluetooth module, and a breadboard interconnecting them to a sturdy chassis (Figure 6, Figure 7. Wheels were mounted to the motors for movement, and the entire assembly was powered using an 11V Li-Po battery. The bot was tested in various environments to evaluate mapping accuracy and control precision. The following observations were made:

- The HC-SR04 sensor accurately measured wall distances, and the IR sensor reliably detected obstacles.
- Real-time mapping on the Android app effectively visualised the room's layout.
- Manual mode provided precise control, while automatic mode efficiently mapped the room.

CHALLENGES FACED

Throughout the project, we encountered several technical challenges that required innovative solutions and persistent troubleshooting. One major issue was related to the power supply. The HC-05 Bluetooth module was sensitive to voltage levels, and the Tiva board's 3.3V output was not stable or sufficient to power it consistently. We resolved this by using an external 6V AA battery pack. Additionally, the initial 6V battery supply proved inadequate for powering the motors and other components simultaneously. To overcome this, we upgraded to an 11V Li-Po battery, which provided the necessary stability and power for the entire system.

Another significant challenge was the accuracy of the sensor readings. The HC-SR04 ultrasonic sensor often gave noisy and inconsistent outputs. To address this, we implemented a *moving average filter* in the code to smooth the sensor data. Furthermore, the lack of working PWM and timer modules on the Tiva board posed a major hurdle. Thus, we decided to implement PWM without the modules, and SysTick for measuring time instants. This added some complexity, but ensured the functionality we needed.

Mechanical control of the bot also presented difficulties. The bot's turning mechanism was imprecise, with turns sometimes overshooting or undershooting the desired angle. We iteratively adjusted the motor control logic and incorporated delays to achieve more consistent performance, but this required extensive testing and fine-tuning.

In summary, this project demanded substantial testing and problem-solving to address the technical challenges. Each issue faced pushed us to develop creative solutions. Ultimately, the bot performed reliably without any delays, and performs the desired task of wall-mapping.

Conclusion and Future Work

This project successfully demonstrates a real-time wall-mapping and control system integrating embedded systems and Android development. Future improvements could include:

- Implementing advanced path-planning algorithms.
- Enhancing obstacle detection capabilities.
- Developing a more compact hardware design.
- Integrating Real-Time Operating System (RTOS) for better scheduling of sensors and reducing delays.

Acknowledgments

We thank Prof. Bishnu Prasad Das for their guidance and resources. The code for the microcontroller and Android Studio for this project are available on GitHub: https://github.com/PrajwalSas/Boombot.git.

References

[1] Texas Instruments, "TM4C123GH6PM Microcontroller Datasheet," [Online]. Available: https://www.ti.com/lit/ds/symlink/tm4c123gh6pm.pdf?ts=1732885846776&ref_url=https%253A%252F%252Fwww.ti.com%252Fproduct%252FTM4C123GH6PM.

[2] Texas Instruments, "L293D Quadruple Half-H Driver Datasheet," [Online]. Available: https://www.ti.com/lit/ds/symlink/l293.pdf.

[3] Microcontrollers lab, "HC-05 Blueooth interfacing with TIVA," [Online]. Available: https://microcontrollerslab.com/hc-05-bluetooth-interfacing-tm4c123g-tiva-launchpad-keil-uvision/.

[4] Microcontrollers lab- HC-SR04 Ultrasound interfacing with TIVA Available:https://microcontrollerslab.com/hc-sr04-ultrasonic-sensor-interfacing-with-tm4c123-tiva-c-launchpad/.