Ultrasonic-Aware Touch Control Car

Aditya Makwana
Tirth Patel
Huaishu Huang

Instructor : Dr. Tamer R Omar

o Table of contents

1. Abstract

This project presents the design and implementation of a touch-controlled robotic car that can operate in two distinct modes: Driving Mode and Ultrasonic Display Mode. The system integrates multiple microcontrollers, including two ESP32 boards and an STM32F401RE, to achieve wireless low-latency control, sensor fusion, and real-time motor actuation. Touch input is captured using a resistive touchpad and TSC2046 controller, transmitted via ESP-NOW, and processed on the STM32 to generate motion commands. Ultrasonic sensors provide distance measurements that enable obstacle detection and automated safety stopping. The project demonstrates effective multi-device communication, stable wireless control, and reliable sensor-based decision-making.

2. Introduction

The objective of this project was to develop a robotic car capable of being controlled wirelessly through a touch-based interface while also incorporating an autonomous mode using ultrasonic sensing. The system required seamless integration of several embedded technologies: a resistive touchpad for input, ESP-NOW wireless communication for high responsiveness communication between car and the remote, I2C communication for microcontroller coordination, and PWM-driven motor control for smooth movement.

The controller portion uses an ESP32 paired with a TSC2046 touch controller to collect X/Y coordinates from the touchpad. These coordinates are then transmitted wirelessly to a second ESP32 located on the robotic car. The receiving ESP32 forwards the data to an STM32F401RE microcontroller, which interprets the touch input and converts it into motion commands such as forward, backward, left, right, or stop.

In addition to manual control, the STM32 can switch the car to Ultrasonic Display Mode through a hardware button. In this mode, three HC-SR04 sensors continuously measure distance to detect obstacles and prevent collisions by automatically halting the car when something is too close.

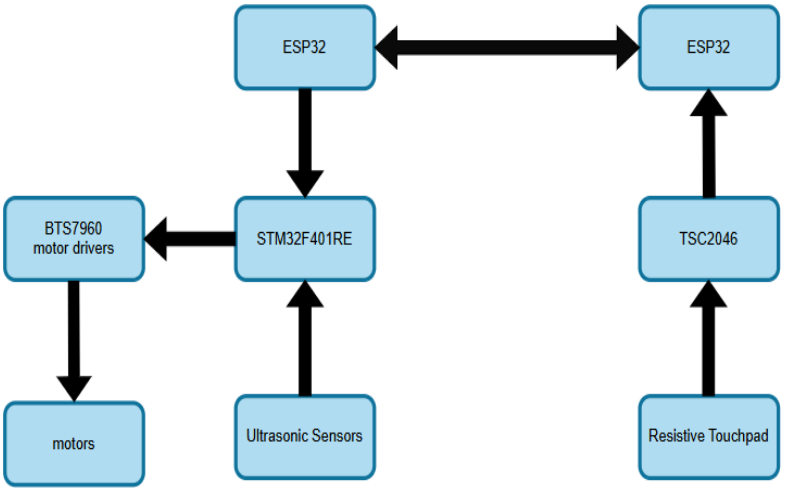

Figure 1. Project block diagram

3. Experimental methodology

1. System Architecture

The system is built using a two-module architecture that separates the user interface hardware from the vehicle's onboard electronics. This design allows for easier debugging, improves reliability, and ensures that each module has a clear and dedicated function. The first module, known as the Controller Module, is responsible for capturing user input through a resistive touchpad. This module consists of an ESP32 transmitter, a TSC2046 touch controller, and the touchpad itself. The ESP32 reads the digital X and Y coordinate data coming from the TSC2046, processes it in real time, and sends it wirelessly to the car using the ESP-NOW protocol. The touch controller setup, including the ESP32 and TSC2046 wired on a breadboard, can be inserted here as an image for reference.

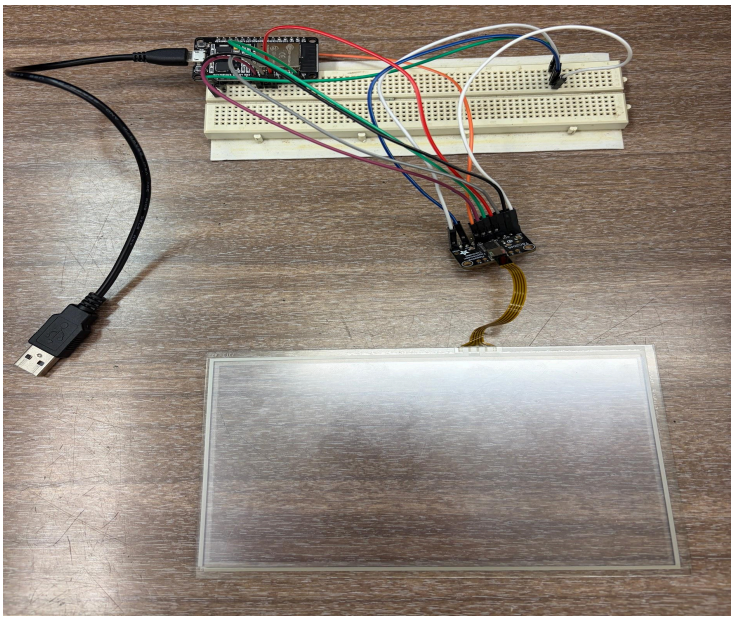

Figure 2.Trackpad connection layout

The second part of the architecture is the Car Module, which receives the touch data and translates it into physical motor movement. It contains an ESP32 receiver that accepts ESP-NOW packets and forwards the coordinate information to an STM32F401RE microcontroller over an I2C connection. The STM32 acts as the main control unit, performing tasks such as interpreting touch regions, generating PWM signals, reading ultrasonic sensor values, and switching between driving and ultrasonic modes. The car uses two IBT-2 motor drivers to operate four DC motors, allowing for smooth and responsive movement. Additionally, three HC-SR04

ultrasonic sensors are integrated on the front, left, and right sides of the car to provide real-time obstacle detection.

A dedicated system-level connection diagram can be placed here to show how all the electronic components interface with each other. This diagram visually explains the wiring connections between the ESP32 boards, the STM32 microcontroller, the motor drivers, the ultrasonic sensors, and the motors. It highlights how the sensor signals, motor control lines, and communication lines flow throughout the system.

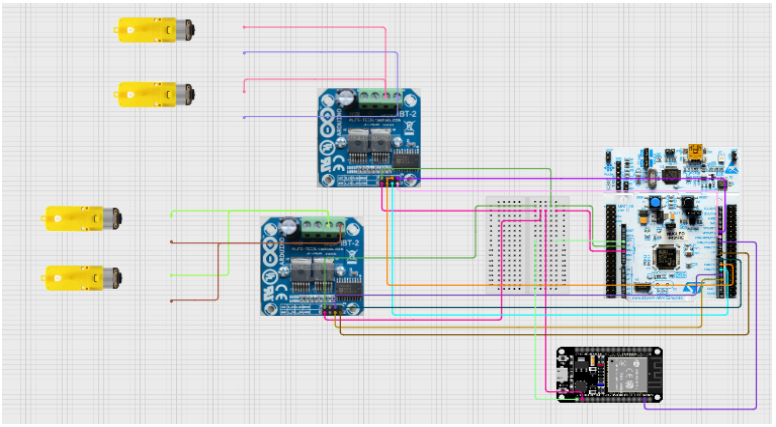

Figure 3: System-Level Connection Diagram

In addition to the signal-level wiring, the project also includes a separate power supply and control line diagram. This diagram presents the power distribution network for the entire system, showing how the battery power is routed to the IBT-2 motor drivers, how regulated voltages are supplied to the ESP32 and STM32, and how common grounding is maintained to ensure stable and noise-free operation. It also shows the UART communication wiring and the high-current paths used to drive the motors. Including this diagram helps distinguish the low-power logic wiring from the high-current motor wiring, making troubleshooting and understanding the system much easier.

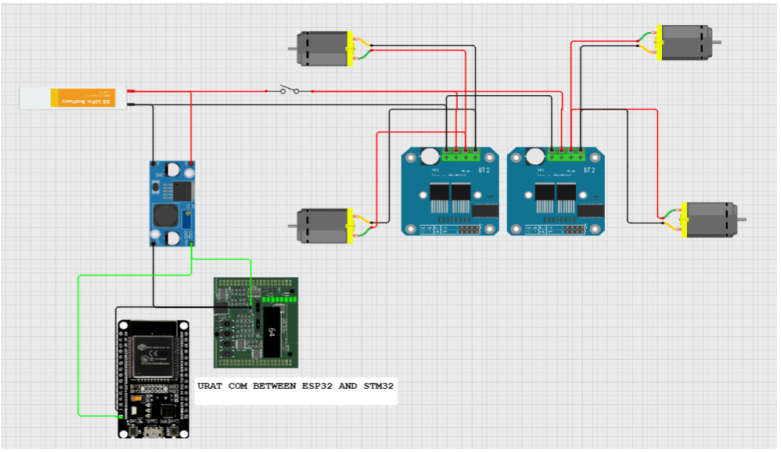

Figure 4: Motor Power Supply

Wireless communication between the Controller Module and the Car Module is handled entirely using ESP-NOW. This protocol allows for extremely low-latency, peer-to-peer transmission of touch coordinates without requiring a Wi-Fi network. In practice, ESP-NOW enables the car to respond instantly to touch movements, resulting in smooth and real-time control. Once both ESP32 devices are configured on the same channel, packet loss is minimal, and the system maintains stable communication even during rapid directional changes. This fast and reliable wireless link forms the core of the interaction between the user and the robotic car.

2. Touch Input Acquisition

The touch input acquisition process begins at the resistive touchpad, where user interactions generate analog voltage variations. These signals are digitized through the TSC2046 touch controller, which communicates with the ESP32 using the SPI interface. This allows the ESP32 to read precise X and Y coordinate values that correspond to the user's touch location on the pad. After obtaining these coordinates, the ESP32 processes and packages them into a compact 6-byte data frame that is optimized for fast and reliable ESP-NOW wireless transmission. When the STM32 receives these coordinates, it interprets them based on predefined directional regions each mapped to a specific vehicle action such as forward, backward, left, right, or stop. This mapping ensures that every touch input on the pad results in a predictable and controlled motion of the car.

3. Wireless Transmission (ESP-NOW)

After determining the user's touch location, the ESP32 transmitter sends the coordinate data to the ESP32 receiver located on the robotic car using the ESP-NOW protocol. ESP-NOW provides extremely low communication latency, ensuring the car responds quickly to changes in user input. Once the receiving ESP32 obtains the data packet, it extracts the X and Y values and forwards them to the STM32F401RE microcontroller using an I2C connection. This design cleanly separates wireless communication from system control, allowing the STM32 to focus solely on processing commands and managing the car's behavior.

STM32 Pin Configuration Overview

The pinout diagram of the STM32F401RE microcontroller provides a clear representation of the functional assignment of each pin in the LQFP64 package. This diagram is important for understanding how different peripherals and signals are mapped to the microcontroller during the development of the robotic car. It shows the

locations of GPIO pins, UART lines, I2C channels, timer channels, analog inputs, and power pins, all of which are essential to the operation of the system. Critical connections used in this project, such as I2C communication lines between the ESP32 and the STM32, PWM outputs for driving the IBT-2 motor drivers, and trigger and echo inputs from the ultrasonic sensors, can be identified directly from the pinout. The diagram also highlights the USART pins used for UART communication as well as timer channels (such as TIM1 and TIM3) used for generating precise PWM signals. By referencing this pin configuration during system wiring, the correct mapping of each signal ensures stable performance, prevents incorrect wiring, and makes the debugging process significantly easier.

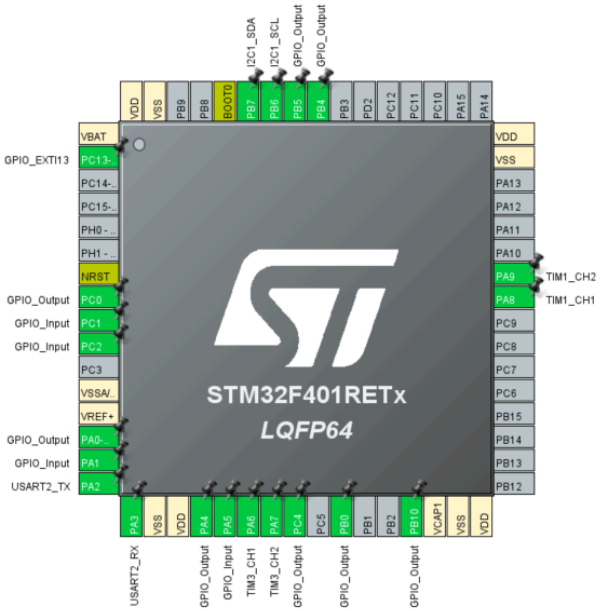

Figure 5 : Stm32 Pin configuration

4. STM32 Drive Mode Algorithm

The STM32F401RE plays a central role in interpreting the incoming touch coordinates and converting them into specific movement commands. It determines which direction the car should move by checking which of the predefined touch regions the coordinates fall into. Once the intended direction is identified, the STM32 generates PWM signals with predefined duty cycles to control the dual IBT-2 motor drivers. Through PWM modulation, the STM32 adjusts motor speed and direction to perform actions such as forward motion, backward motion, left turns, right turns, or complete stop. This algorithm ensures smooth transitions between directions and stable motor behavior, making the car highly responsive to user control.

5. Ultrasonic Mode Operation

In addition to manual touch-based driving, the system includes an Ultrasonic Mode that enhances safety by enabling automatic obstacle detection. This mode is activated using the B1 push button on the STM32 development board. When triggered, the STM32 begins sequentially reading measurements from three HC-SR04 ultrasonic sensors positioned at the front, left, and right sides of the car. If any sensor detects an obstacle closer than 30 centimeters, the STM32 immediately halts all motor activity to prevent collisions.

4. Experimental results

1. Touch Controller Performance

Testing confirmed that the touch controller accurately detected finger position and produced stable X/Y coordinates. Wireless transmission showed no perceptible lag, and the car responded correctly to each direction command.

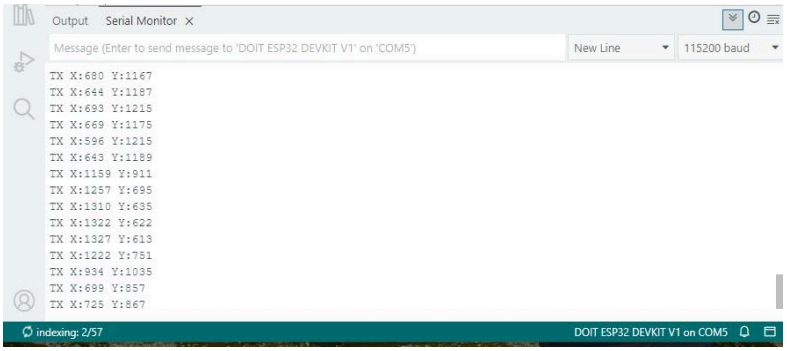

Figure 6: touchpad serial output

2. Motor Control Behavior

The PWM-based motor control implemented on the STM32 provided smooth and predictable motion across all operating conditions. Forward and backward movement remained stable even at varying speeds, and turning maneuvers were performed cleanly through differential motor control. The system also handled idle conditions effectively; whenever the touch input fell outside the defined control zones, the STM32 correctly issued a stop command, ensuring that the car halted safely without unintended motion.

3. Ultrasonic Obstacle Detection

The ultrasonic subsystem performed reliably during obstacle detection tests. All three HC-SR04 sensors produced accurate and consistent distance measurements, enabling the car to detect nearby objects in real time. When any obstacle was detected within a 30-centimeter threshold, the STM32 immediately halted motor operation to prevent collisions. Once the obstacle was removed and the path was clear, the car resumed movement without delay. UART outputs confirmed stable timing and distance readings, demonstrating the effectiveness of the sensing algorithm.

```
Front=7 cm  Right=69 cm  Left=218 cm
Front=4 cm  Right=70 cm  Left=218 cm
Front=4 cm  Right=70 cm  Left=218 cm
Front=4 cm  Right=70 cm  Left=218 cm
Front=6 cm  Right=70 cm  Left=218 cm
Front=6 cm  Right=70 cm  Left=218 cm
Front=7 cm  Right=70 cm  Left=218 cm
Front=9 cm  Right=70 cm  Left=218 cm
```

Figure 7: Ultrasonic output

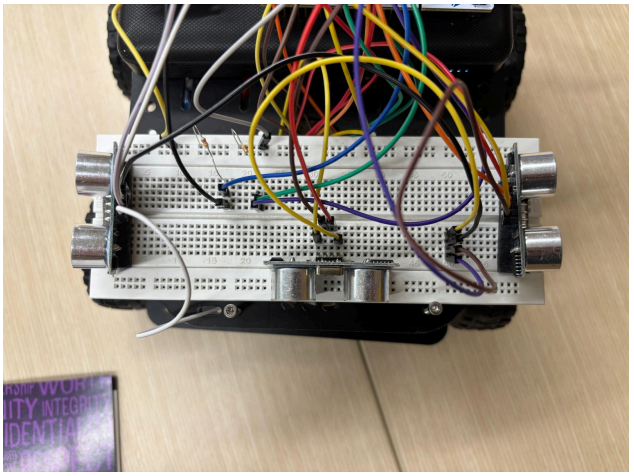

Figure 8: Ultrasonic placement

4. System Integration Results

Overall system integration proved to be robust and stable. The two ESP32 modules communicated dependably using ESP-NOW, maintaining continuous wireless connectivity throughout testing. The STM32 microcontroller successfully processed both touch input data and ultrasonic sensor readings simultaneously, coordinating them without interrupting system performance. Additionally, all hardware components—including motors, motor drivers, sensors, and wiring—functioned reliably under continuous operation, validating the soundness of the electrical and firmware design.

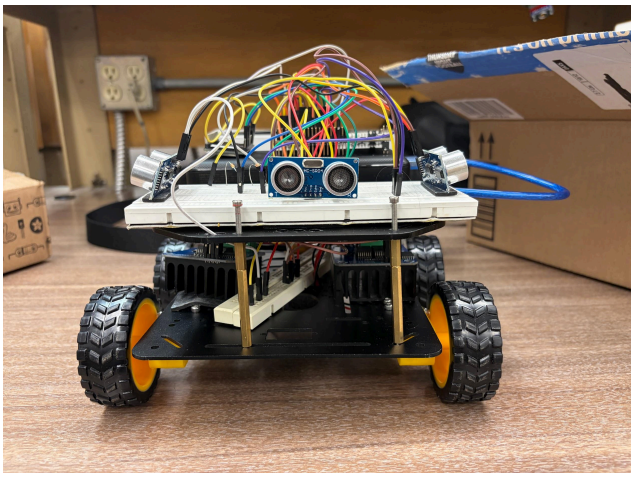

Figure 9 : component placement

5. Summary of challenges and how they were solved

1. Unstable Touchpad Readings

Challenge:
The raw touchpad values were noisy and fluctuated constantly, even when the finger was still. This caused wrong direction detection and jerky motor movement.

Solution:
We implemented a form of touch filtering that checks whether the touch input is stable and recent before accepting it.
We also added hysteresis, meaning the system sticks to the previous direction if the finger has not clearly moved to a different zone. This stabilized the input and removed jitter.

2. Difficulty Mapping Touch Input to Motor Directions

Challenge:
The touchpad produced raw X-Y coordinates that needed to be converted into meaningful motor commands like forward, backward, left, or right.

Solution:
We calibrated the touchpad by defining four reference zones.
Then we used nearest-zone selection to find which zone the finger was closest to.
This made direction control accurate and intuitive.

3. ESP-NOW Wireless Packet Reliability

Challenge:
Touch data sent from the ESP32 touchpad side sometimes did not reach the ESP32 receiver reliably.

Solution :
We improved communication stability by structuring the data properly and ensuring peer-to-peer synchronization.
The system now sends small, consistent data packets and acknowledges them, resulting in reliable wireless control.

4. I²C Data Consistency Between ESP32 and STM32

Challenge:

When the STM32 requested touch data from the ESP32 receiver, the data was sometimes incomplete or invalid.

Solution :

We added data validation and sanity checks.

Only properly formatted and complete touch packets are processed.

This ensured the STM32 always operates on valid touch information.

5. Ultrasonic Sensor Inaccuracy

Challenge:

Ultrasonic sensors sometimes returned fluctuating or incorrect distances due to angle, surface, or environmental noise.

Solution :

We added timeout handling, multiple echo checks, and conversion smoothing to ensure that only stable distance readings are used.

This made obstacle detection accurate and dependable.

6. Conclusions

The project successfully demonstrates a dual-mode embedded control system built around the STM32 microcontroller, ESP32 wireless modules, IBT-2 motor drivers, ultrasonic sensors, and a capacitive touchpad. The system allows a user to manually control the motors through a touch interface while also providing an ultrasonic distance-display mode for obstacle monitoring.

In Drive Mode, the touchpad sends X-Y coordinates wirelessly through ESP-NOW, and the STM32 interprets these positions to drive the motors forward, backward, left, or right. The implementation of touch filtering, hysteresis, debouncing, and calibrated control zones results in smooth and stable manual operation. In Ultrasonic Display Mode, the system measures distances from three ultrasonic sensors front, left, and right and displays the readings through UART. A safety stop mechanism prevents movement when an obstacle is too close, ensuring safer operation. This mode is intended purely for distance inspection and monitoring.

Multiple challenges such as noisy touch data, inconsistent sensor readings, communication reliability, and motor mismatch were solved through classical embedded techniques like filtering, data validation, hysteresis, PWM trimming, and timing management. These solutions ensured accurate sensing, stable communication, and balanced motor performance.

Overall, the project achieves its goal of creating a responsive, user-controlled motor system with integrated safety feedback. It demonstrates strong proficiency in embedded programming, real-time interfacing, wireless communication, and multi-sensor integration. The system provides a solid foundation that can be extended with additional features in the future, but remains strictly manual and sensor-assisted in its current form.

7. References

[1] STMicroelectronics, ***"STM32F4 Series Reference Manual: RM0090 – 32-bit ARM Cortex-M4 MCUs,"*** **2022.**
Available: https://www.st.com

[2] STMicroelectronics, ***"STM32 HAL Library User Manual,"*** **2023.**
Available: https://www.st.com

[3] Espressif Systems, ***"ESP32 Technical Reference Manual,"*** **2022.**
Available: https://www.espressif.com

[4] Espressif Systems, ***"ESP-NOW User Guide,"*** **2021.**
Available: https://docs.espressif.com

[5] Espressif Systems, ***"ESP32 Wi-Fi API Reference,"*** **2022.**
Available: https://docs.espressif.com/projects/esp-idf/en/latest/api-guides/wifi.html

[6] Texas Instruments, ***"I2C Bus Specifications and User Guide,"*** **2019.**
Available: https://www.ti.com

[7] Arduino, ***"Wire (I2C) Library Documentation,"*** **2023.**
Available: https://www.arduino.cc

[8] Infineon Technologies, ***"BTS7960 High-Current Half-Bridge Motor Driver Datasheet,"*** **2020.**
Available: https://www.infineon.com

[9] MaxBotix Inc., ***"Ultrasonic Sensor Theory of Operation,"*** **2020.**
Available: https://www.maxbotix.com

[10] ARM Ltd., ***"Cortex-M4 Devices Generic User Guide,"*** **2021.**
Available: https://developer.arm.com