Android Mobile Phone Controlled Bluetooth Car Using 8051 Microcontroller

A Course Project Report Submitted to the APJ Abdul Kalam Technological University
in partial fulfilment of requirements for the award of degree
Bachelor of Technology
in
Applied Electronics and Instrumentation Engineering
by

SAVAN J SAJI
(TVE22AE055)

Computer Architecture and Embedded Systems
AET 305

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING
COLLEGE OF ENGINEERING TRIVANDRUM
KERALA
November 2024

DEPT. OF ELECTRONICS & COMMUNICATION ENGINEERING
COLLEGE OF ENGINEERING TRIVANDRUM
2024 - 25

CERTIFICATE

This is to certify that the report entitled **Android Mobile Phone Controlled Bluetooth Car Using 8051 Microcontroller** submitted by : **Savan J Saji** (TVE22AE055) to the APJ Abdul Kalam Technological University in partial fulfilment of the B.Tech. degree in Applied Electronics and Instrumentation Engineering is a bonafide record of the course project work carried out by him under our guidance and supervision. This report in any form has not been submitted to any other University or Institute for any purpose.

Prof. Joaquim Ignatious Monteiro
Assistant Professor
Dept. of ECE
College of Engineering
Trivandrum

Abstract

This paper presents the design and implementation of an Android Mobile Phone Controlled Bluetooth Car Using an 8051 Microcontroller. This project involves the design and implementation of a Bluetooth-controlled car using an 8051 microcontroller. The car is operated remotely via a smartphone, utilising the HC-05 Bluetooth module for wireless communication. The vehicle's movement is controlled by an AT89S52 microcontroller, interfaced with an L293D motor driver to manage the DC motors. Key components include LEDs for status indication, a voltage regulator, and essential passive components to ensure stable operation. The system demonstrates the integration of microcontroller programming with wireless technology, offering a practical application in the field of embedded systems and robotics.

Contents

Chapter 1

Introduction

Nowadays, smartphones are increasingly powerful, with enhanced processors, larger storage, and expanded communication capabilities. Bluetooth, primarily used for data exchange, has transformed how we interact with digital devices, enabling a shift from wired to wireless connections. Created by Ericsson in 1994 [**?**], Bluetooth technology integrates seamlessly with smartphones, allowing communication with up to seven devices simultaneously within an eight-meter range [**?**]. This is especially useful in home environments.

The growing popularity of smartphones, combined with technologies like Bluetooth, has turned them into versatile portable devices. MIT APP Inventor an open-source platform widely used in smartphones [**?**], offers a comprehensive software package that includes an operating system, middleware, and core applications. Unlike iOS, it provides a software development kit (SDK) for easy app development. Utilizing smartphones as the "brains" of robots is an emerging field, offering numerous research opportunities. This paper reviews mobile phone-controlled robots and presents a closed-loop control system using audio channels of mobile devices.

In our study, an Android application, such as MIT App Inventor for Mobile Controlled RC Car using an 8051 microcontroller, is used to control the car's movements.

1.1 Problem Statement

The design of a robotic system has to consider many factors. This project was developed to solve the following problems:

- Accessibility to control systems

A lot of technology requires proprietary control systems making them less accessible to any layman trying to use it. This can be solved via an easily available cross-compatible control system. Phones being omnipresent today make them ideal methods to control robotic systems. Bluetooth connectivity ensures that anyone with a smartphone can easily control this device.

- Intuitive control

Intuitive control systems reduce the steepness of the learning curve. This can be achieved by designing an app tailored to the system. Smartphone control systems are already user-friendly with wide acceptance in the general public. Additionally developing apps with clean interfaces and automated procedural command sequences is becoming increasingly easier effectively letting anyone with the know-how redesign the control system to suit their need.

- Flexibility

General-purpose robotic systems need to be flexible in their functionality to allow for programming in a wide range of procedures as is required by the task. Utilising a programmable chip like the 89S51 allows for reprogramming the command responses of the bot as is required on the fly.

1.2 Project Objectives

1. Designing and implementing a flexible wirelessly controlled bot that is easily re-programmable.
2. Utilization of easy-to-use control systems in the form of mobile apps with clean interfaces for robotic control.
3. Studying UART communication in 89S51 micro-controller system with emphasis on the use of the HC-05 module.
4. Studying assembly language and low-level programming to implement the aforementioned systems.

Chapter 2

Project Design and Testing

The design process for any project involves a series of key steps, each contributing to the overall success of the final product. These steps include:

- Hardware Design: This stage focuses on planning and building the physical circuitry, considering factors like power usage and sensor integration.

- Code Generation: Once the hardware is designed, efficient code is created to control the hardware components.

- Debugging: Identifying and resolving errors in both the hardware and software is crucial to ensure the project's functionality.

- Implementation: This stage involves assembling the physical components, integrating software, and rigorously testing the system to validate its operation.

- Bug Fixes: Addressing any issues that arise throughout the design process is a continuous part of the development cycle.

By completing each stage effectively, the project can be comprehensively developed and refined, leading to a successful final outcome.

2.1 Hardware Design

2.1.1 Components

This project utilizes the following hardware components:

- AT89S52 Microcontroller
- L293D Motor Driver IC
- HC-05 Bluetooth module
- 11.0592MHz crystal oscillator
- Voltage Regulator (7805)
- Capacitors and resistors
- Breadboard
- Jumper Wires
- Lithium Polymer Battery, etc.

2.1.2 Circuit Schematic

2.2 Software Design

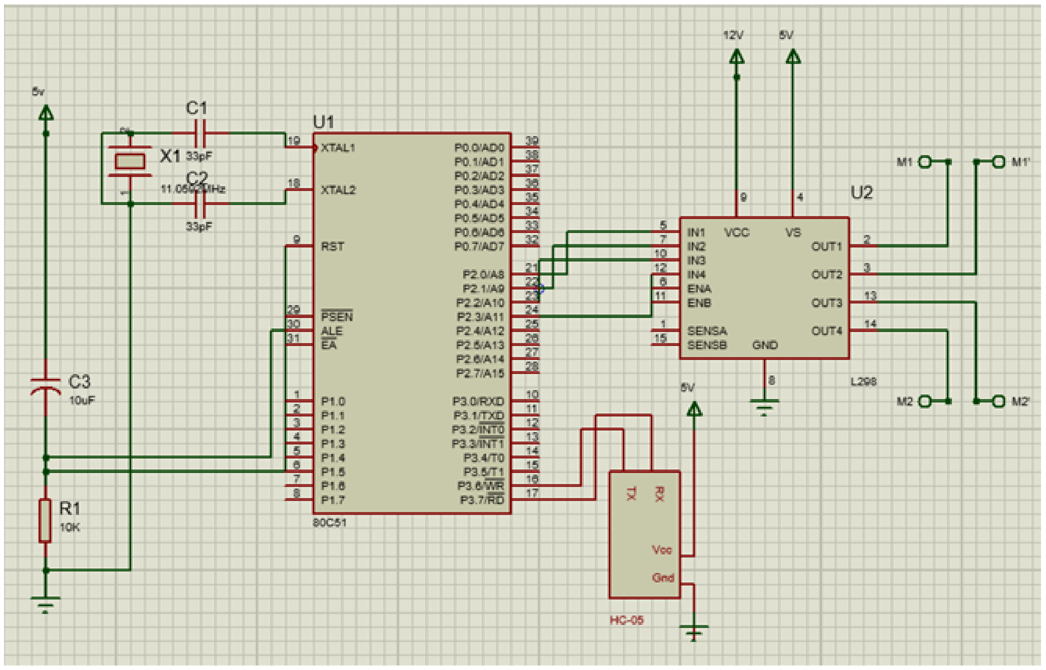
Various software tools, such as Keil C51 IDE, MCU 8051 IDE, and Proteus, are used for tasks such as code development, circuit design, and simulation:

- **Keil C51 IDE**

This tool was used to conceptualise and create the code of this project. Its debugging and desimulation tools were used extensively. It was also used to build the assembly code to create .hex files.

- **MCU8051 IDE**

This tool was used for code creation due to the limited support of Keil C51 IDE in Linux systems. It also has an interface that directly shows the registers in the default state. This tool was later abandoned as Keil was set to run via WINE.



- **Avrdude**

Avrdude was used to burn the code into the 89S51 chipset. It was later automated to a shell script to burn any file copied into a directory with one click. This tool was chosen for the code-burning process due to its CLI-based nature granting it easy scriptability. It is also widely supported on a lot of systems.

- **Proteus**

Proteus was used to generate circuit schematics for this project.

- **Arduino IDE**

Arduino IDE served to program an Arduino to act as an ISP. It was utilised in the code-burning process as an intermediary between the chip set and the PC.

- **Serial Bluetooth Terminal (Android)**

This app was used to send commands from an Android device to the bot. It was chosen due to its simple interface and easy debuggability

Code generation and simulation were done using assembly Keil. Hardware design and circuit schematic generation was done using Proteus. Burning of the created code was done via an Arduino as ISP. Avrdude was scripted to automatically burn .hex files copied to a directory to the connected micro-controller for easier code burning. The Serial Bluetooth Terminal app was used to control the bot via a smartphone.

2.3 Testing

This project underwent thorough testing, moving from simulations to real-world hardware testing on different surfaces. The steps for the procedure are as follows:

- The code was simulated using Keil C51 IDE. The process of character reception and further port simulation was done in this fashion. This helped in clearing problems with the code of the project.
- The base circuit for 89S51 function was created and its functionality was tested with a blink program. This part tested each port to ensure hardware functionality.
- The character reception was attempted with the HC-05 module. This part led to recognising that the 12MHz oscillator would not work for this project. This was corrected and successful character reception and echo transmission was achieved.
- The final bot structure was assembled. All wiring was tested to ensure its correctness.
- The delay function was iteratively adjusted to achieve a satisfactory delay between each action.
- Several different smartphone apps were tested for ease of use, demonstration capability, ease of connection and clean user interfaces.
- Rigorous testing was done to ensure the functionality of the project. Tested on various surfaces including tiles, sand, gravel, asphalt and concrete. The bot's speed, manoeuvrability, ease of control, and collision resistance were tested. Additional reinforcement as insulation tape is provided to secure electronic components from harder strikes.

Chapter 3

Conclusion

3.1 Roadblocks in Implementation

During the implementation of the process, several problems had to be solved. Those faced are as follows:

- **Wrong oscillator chosen**

The crystal oscillator initially bought for the project was 12MHz. This oscillator cannot be made to set the baud rate as 9600 for serial communication. This problem arose from incorrectly assuming it could replace the 11.0592MHz oscillator due to a lack of knowledge in the field. It was corrected by further study.

- **Codeburning**

Streamlining the process of burning the code onto the microcontroller device required a lot of troubleshooting. The need for constantly alt, f i n i s h e r ; b a c k l e f t 71 s e t b p 2 . 1 72 s e t b p 2 . 3ering the circuit to burn along with subsequent software routines made burning the code arduous. The was partially by scripting avrdude to perform one-click code burning along with practice in the connections required.

- **Delays**

The delay for performing each action had to be adjusted manually to account for hardware-specific conditions. Due to the presence of a geared system in the chassis, the wheels could carry momentum, unlike a traditional bot. This required careful adjustment of the delays to correct.

- **Connection Issues**

The bot on receiving a constant influx of instructions gets stuck on the instruction. This was corrected using precise delays for instruction blocking.

3.2 Possible Improvements

The RC car could be enhanced to function as a surveillance rover by incorporating an ESP32 camera.

3.3 Result

The 8051-based Bluetooth-controlled bot project was successfully completed and implemented in hardware. It required an in-depth understanding of UART communication, 8051 programming, and hardware debugging. A detailed understanding of its implementation came from studying its various aspects and a lot of debugging. The final result was a bot that could easily manoeuvre through a crowded classroom. The control was intuitive as it was via an app.

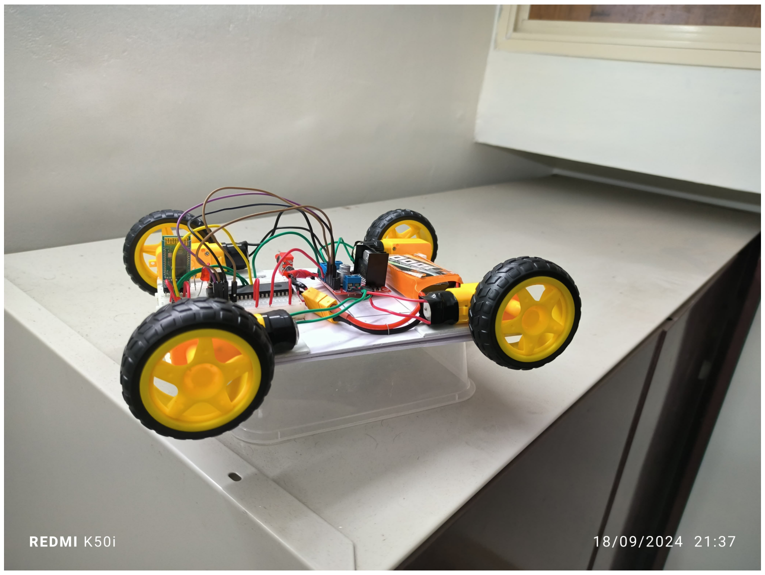


Figure 3.1: RC Bluetooth Controlled Car Using 8051 M.C.

Chapter 4

Bill of Materials

Sl.no	Item	Manufacturer	Price (Rs)	Quantity
1	AT89S52	Atmel	67.50	1
2	Crystal Oscillator	Generic	5.5	1
3	Capacitors	Generic	0.3	2
4	L293D Motor Driver IC	Generic	122	1
5	HC-05 Bluetooth Module	Generic	168	1
6	DC Motors with Wheels	Super Debug	265	4

Table 4.1: Bill of Materials

Chapter 5

Code

```
#include <reg51.h>

// Define pins for motor control
sbit IN1 = P2^5;  // Control pin 1 (connected to motor driver)
sbit IN2 = P2^1;  // Control pin 2 (connected to motor driver)
sbit IN3 = P2^2;  // Control pin 3 (connected to motor driver)
sbit IN4 = P2^3;  // Control pin 4 (connected to motor driver)
sbit LED = P2^4;  // LED pin to check whether the microcontroller is
running
// Function prototypes
void ser_int();
unsigned char rx();
 // UART initialization function
 // Function to receive data via UART
void tx(unsigned char a);
void delay();
// Function to transmit data via UART
 // Delay function for simple timing
void main() {
unsigned char get;
int m;
LED = 0;
// Variable to store received UART data
 // Loop counter
 // Turn off LED initially
// Blink LED 10 times to ensure the system is running
for (m=0; m<10; m++) {
        delay();              // Wait for some time
        LED = LED ^ 1;        // Toggle LED state
    }

    ser_int();                // Initialize UART communication

    while(1) {
        get = rx();           // Receive a character via Bluetooth/serial
communication
        tx(get);              // Transmit the received character back (echo
function)

        // Motor control based on received command
        if (get == 'F') {
            /* Move Forward */
            IN1 = 1;          // Activate motor in forward direction
            IN3 = 1;
            IN2 = 0;
            IN4 = 0;
        } else if (get == 'B') {
            /* Move Backward */
            IN2 = 1;          // Activate motor in reverse direction
            IN4 = 1;
            IN1 = 0;
            IN3 = 0;
        } else if (get == 'R') {
            /* Turn Right */
            IN1 = 1;          // Rotate the motor in a way to turn the robot to th
right
            IN2 = 0;
            IN3 = 0;
            IN4 = 0;
        } else if (get == 'L') {
            /* Turn Left */
            IN3 = 1;          // Rotate the motor in a way to turn the robot to th
left
```

```
            IN2 = 0;
            IN1 = 0;
            IN4 = 0;
        } else if (get == 'S') {
            /* Stop the Motor */
            IN1 = IN2 = IN3 = IN4 = 0;  // Stop all motor control signals
        }
    }
}

// UART Initialization Function
void ser_int() {
SCON = 0x50;    // Set serial mode to 8-bit UART
TMOD = 0x20;    // Set Timer 1 to Mode 2 (auto-reload)
TH1 = TL1 = 0xFD; // Set baud rate to 9600 (FD for 11.0592 MHz
clock)
TR1 = 1;
}
 // Start Timer 1
// Function to receive data via UART
unsigned char rx() {
while (RI == 0);   // Wait until data is received (RI = Receive
Interrupt flag)
RI = 0;
return SBUF;
}
 // Clear the Receive Interrupt flag
// Return the received byte from the serial buffer
// Function to transmit data via UART
void tx(unsigned char a) {
SBUF = a;
 // Load data into the serial buffer (SBUF)
while (TI == 0);   // Wait until the data is transmitted (TI = Transmit
Interrupt flag)
TI = 0;
 // Clear the Transmit Interrupt flag
}
// Simple delay function
```

```
void delay() {
int i, j, k;
for (i = 0; i < 100; i++) {
for (j = 0; j < 100; j++) {
for (k = 0; k < 10; k++);
}
}
}
```

References

- “HC-05 Bluetooth Module Interfacing with 8051 Controller,” ©
- 2018 ElectronicWings.https://www.electronicwings.com/8051/hc-05-bluetoothmodule-interfacing-with-8051 Instructables, “How to Program 8051 Using Arduino!,” Instructables, Nov.
- 2020, [Online]. Available: https://www.instructables.com/How-to-Program 8051-Using-Arduino”
- “UART: A Hardware Communication Protocol Understanding Universal Asynchronous Receiver/Transmitter — Analog Devices,” www.analog.com, [Online]. Available: https://www.analog.com/en/analog-dialogue/articles/uart-a hardware-communication-protocol.htm here
- Mazidi, M.A., et al. The 8051 Microcontroller and Embedded Systems.
- ref1 Ericsson, ”Bluetooth Technology Overview,” 1994.
- ref2 ”Bluetooth Device Communication Capabilities.”
- ref5 ”Android Platform and Development Kit Overview.”
- ”RC Bluetooth Car using 8051 Instructables” https://www.instructables.com/Bluetooth-Controlled-Car-Using-8051/