

Voice Controlled DC MOTOR CAR

Monib SAAdi

hassan Abdullah

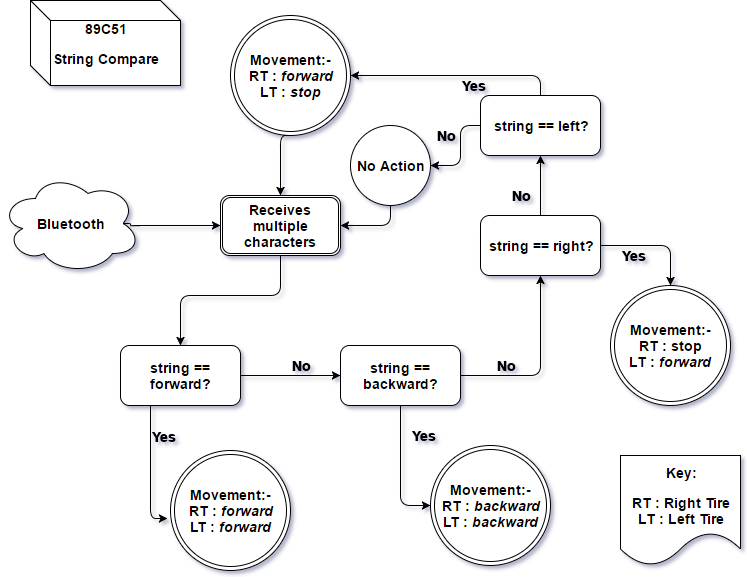
Muhammad Usama

BEE 7B | Microprocessor and Systems | June 2, 2017.

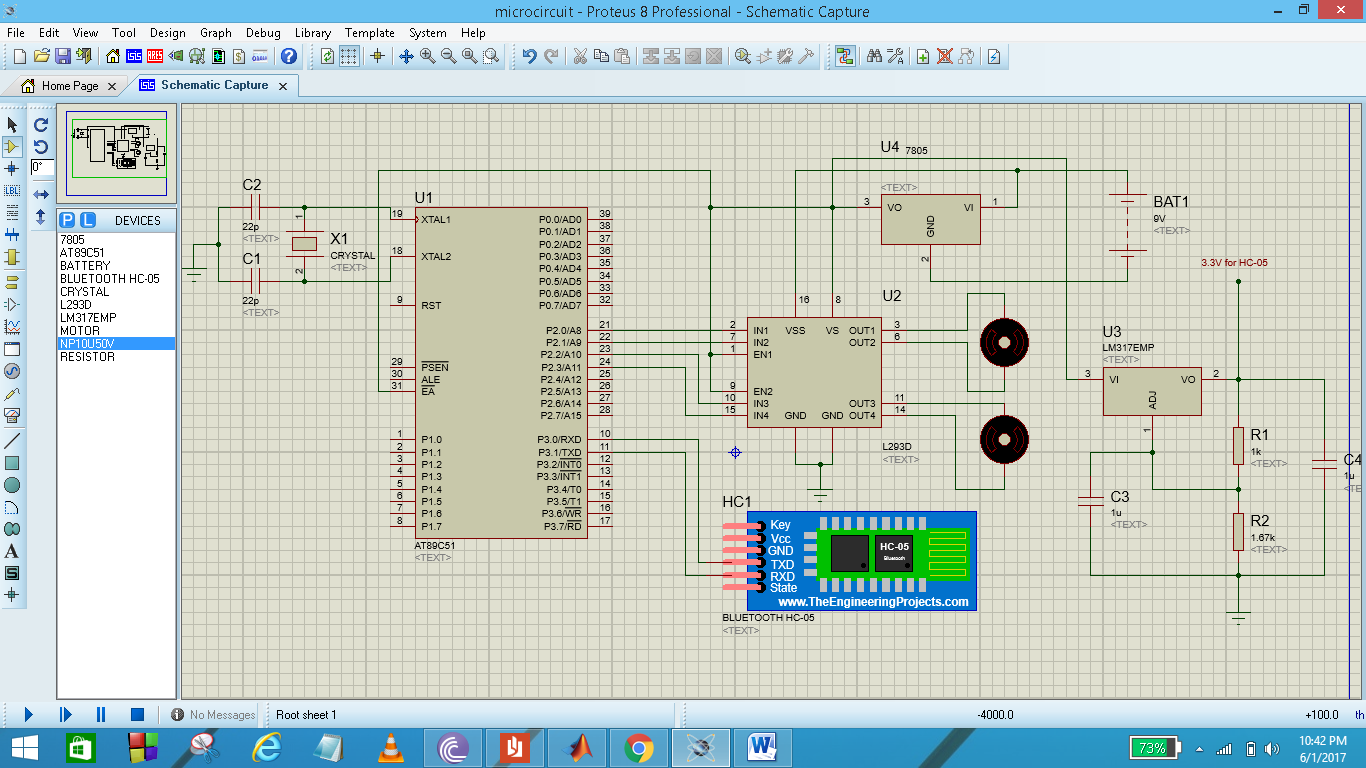
* ABSTRACT

The purpose of this project was to learn the use of a microcontroller (89c51) in practical application. This will be demonstrated by controlling the movement of an electrical car through voice commands. These commands will be sent in the form of strings through an android app, specifically made for this project, using Bluetooth communication. The commands will then be received by the Bluetooth module HC-05 which will then be transmitted to the microprocessor through serial communication. The processor, using an algorithm written in C, will then make decisions based on the data it receives, and will set its output ports to either high or low based on which direction the motor is to be rotated. This in turn will drive the car.

* BLOCK DIAGRAM



* CIRCUIT DIAGRAM



* code

#include <reg51.h>

sbit motRf = P2^0;

sbit motRb = P2^1; //Two ports are given to each motor, so we can control direction

sbit motLf = P2^2;

sbit motLb = P2^3;

sbit motRfb = P2^4;

sbit motRbb = P2^5;

sbit motLfb = P2^6; //CODE IS FOR 4 MOTORS We may only use 2 in real-time for effeciency.

sbit motLbb = P2^7;

void main (void)

{

TMOD = 0x20; //Using Timer1, in 8-bit auto reload mode

TH1 = 0xFD; // To set Baud Rate = 9600, So we can communicate with HC-05

SCON = 0x50; //SETTING MOD 1 for DATA TRANSFER, 8 BIT DATA with start and stop bits

TR1 = 1; //STARTS TIMER

IE = 0x90; //ENABLES SERIAL INTERRUPT

while(1); //NEVER ALLOWS THE MAIN TO END

}

void ser\_intr(void)interrupt 4

{

char d;

IE = 0x00;

while(RI == 0);

d = SBUF;

RI = 0;

if(d == 'f'){ //forward

while(RI == 0);

d = SBUF;

RI = 0;

if(d == 'o'){

while(RI == 0);

d = SBUF;

RI = 0;

if(d == 'r'){

while(RI == 0);

d = SBUF;

RI = 0;

if(d == 'w'){

motRf = 1;

motRb = 0;

motLf = 1;

motLb = 0;

motRfb = 1;

motRbb = 0;

motLfb = 1;

motLbb = 0;

}

}

}

} //forward

else if(d == 'b'){ //backward

while(RI == 0);

d = SBUF;

RI = 0;

if(d == 'a'){

while(RI == 0);

d = SBUF;

RI = 0;

if(d == 'c'){

while(RI == 0);

d = SBUF;

RI = 0;

if(d == 'k'){

motRf = 0;

motRb = 1;

motLf = 0;

motLb = 1;

motRfb = 0;

motRbb = 1;

motLfb = 0;

motLbb = 1;

}

}

}

}//backward

else if(d == 'r'){ //right

while(RI == 0);

d = SBUF;

RI = 0;

if(d == 'i'){

while(RI == 0);

d = SBUF;

RI = 0;

if(d == 'g'){

while(RI == 0);

d = SBUF;

RI = 0;

if(d == 'h'){

motRf = 0;

motRb = 0;

motLf = 1;

motLb = 0;

motRfb = 0;

motRbb = 0;

motLfb = 1;

motLbb = 0;

}

}

}

} //right

else if(d == 'w'){ //right

while(RI == 0);

d = SBUF;

RI = 0;

if(d == 'r'){

while(RI == 0);

d = SBUF;

RI = 0;

if(d == 'i'){

while(RI == 0);

d = SBUF;

RI = 0;

if(d == 't'){

motRf = 0;

motRb = 0;

motLf = 1;

motLb = 0;

motRfb = 0;

motRbb = 0;

motLfb = 1;

motLbb = 0;

}

}

}

} //right

else if(d == 'l'){ //left

while(RI == 0);

d = SBUF;

RI = 0;

if(d == 'e'){

while(RI == 0);

d = SBUF;

RI = 0;

if(d == 'f'){

while(RI == 0);

d = SBUF;

RI = 0;

if(d == 't'){

motRf = 1;

motRb = 0;

motLf = 0;

motLb = 0;

motRfb = 1;

motRbb = 0;

motLfb = 0;

motLbb = 0;

}

}

}

} //left

else if(d == 's'){ //stop

while(RI == 0);

d = SBUF;

RI = 0;

if(d == 't'){

while(RI == 0);

d = SBUF;

RI = 0;

if(d == 'o'){

while(RI == 0);

d = SBUF;

RI = 0;

if(d == 'p'){

motRf = 0;

motRb = 0;

motLf = 0;

motLb = 0;

motRfb = 0;

motRbb = 0;

motLfb = 0;

motLbb = 0;

}

}

}

} //stop

RI = 0;

IE = 0x90;

}

* DESCRIPTION

At first let us explain the **Voice Command Application**, we have used MIT App Inventor, to make the app. We’ve used the Google voice API, in our app. Our App recognizes the words and send them to the Bluetooth module HC-05.

These words are sent character by character, i.e. serially, by the **Hc-05** to the Microcontroller. This is done through the TX RX Pins.

**On the software end**, we have basically used Interrupts to handle Serial Communication.

Our C-Code which we compiled on Keil uVision, accepts characters which are serially transferred to the microcontroller and then performs a character by character comparison on it. Since there is **no built-in function for string comparison**, we perform this task using brute force approach and compare each 8-bit character one at a time. After a successful or unsuccessful comparison, the microprocessors readies itself for the next command, since we are using an infinite loop for the purpose.

The commands which can be accepted are “forward”, “backward”, “left”, “right”, “stop”. Each of these commands works perfectly. Furthermore any sentence incorporating any of these words also do the same function.

Now on the hardware front, the main problem is the current division. Since in such a project we can cannot put a DC Power Supply on its top thus we had to devise a plan to give appropriate amount of current to each peripheral.

We have used 2 DC Batteries which are connected in series. Collectively they provide 8.4V and individually they are capable of providing 1.2mAh. They are connected in parallel with motor drivers (L293D) and 7805. 7805 limits the amount of current and gives output voltage of 5V. This 5V is connected in parallel to 89C51 and to LM317 which gives output voltage of 3.3V. 7805 can only give 5V voltage but LM317 can have a variable output depending upon resistances attached with it, so it is also called adjustable regulator. And most importantly the circuitry is required to have a common ground.

* CONCLUSION

This project helped us to get an insight of 8051’s features such as serial communication and motor control. First and foremost challenge was to understand how serial communication works and how the internal registers such as timers and interrupts can be used to get data from serial buffer. Next stage was to interface HC-05 Bluetooth module with AT89c51. Setting its baud rate and receiving data from it. The Android App was developed using MIT App Inventor. Passing strings to 8051 was quite challenging too. We had to pass the whole string such as ‘forward’ and make 8051 analyze and decode it.

Providing 3.3 volts for HC-05 was also a challenge and it was tackled using LM-317 voltage regulator and accompanying circuitry. Interfacing the crystal oscillator with the microcontroller was an unresolved issue due to hardware incompatibility. Ultimately we had to use developer board for the clock circuitry.

This project is just a stepping stone towards a more advance project. The next phase can include mind control, using sensors for detecting brain waves and transmitting it to PIC or ARDUINO for further processing. To enhance the computation time two microntrollers can be used one for analyzing the signal and other for controlling the actuators.