Wireless Joystick Controlled Zumo Bot

Elizabeth Heider, Adrian Ruiz

Goal:

- Control DC Motor Robot Wirelessly with ESP8266 Via WIFI
- Enable Robot to move directions forward, reverse, left and right
- Controlled wirelessly with phone application “Joystick”

Deliverables:

This project is intended to transmit control functions of the DC Motor Robot wirelessly through Wifi, although programmed in AVR. The DC motor robot will respond to controls from a mobile application “blink”. Using Pulse Width Modulation (PWM) we are able to control the speed of the robot as well as the direction (left, right, forwards, backwards). Using USART we are able to transmit controls via Wifi connecting to the NodeMCU.

I. Components

A. Atmega328P

The Atmega328P is an 8 bit microcontroller that we programmed in C (although can be programmed in AVR Assembly). The Atmega 328P contains 6 PWMs, 3 Timer/counters, and an internal 8Mhz clock. Only one USART model included.

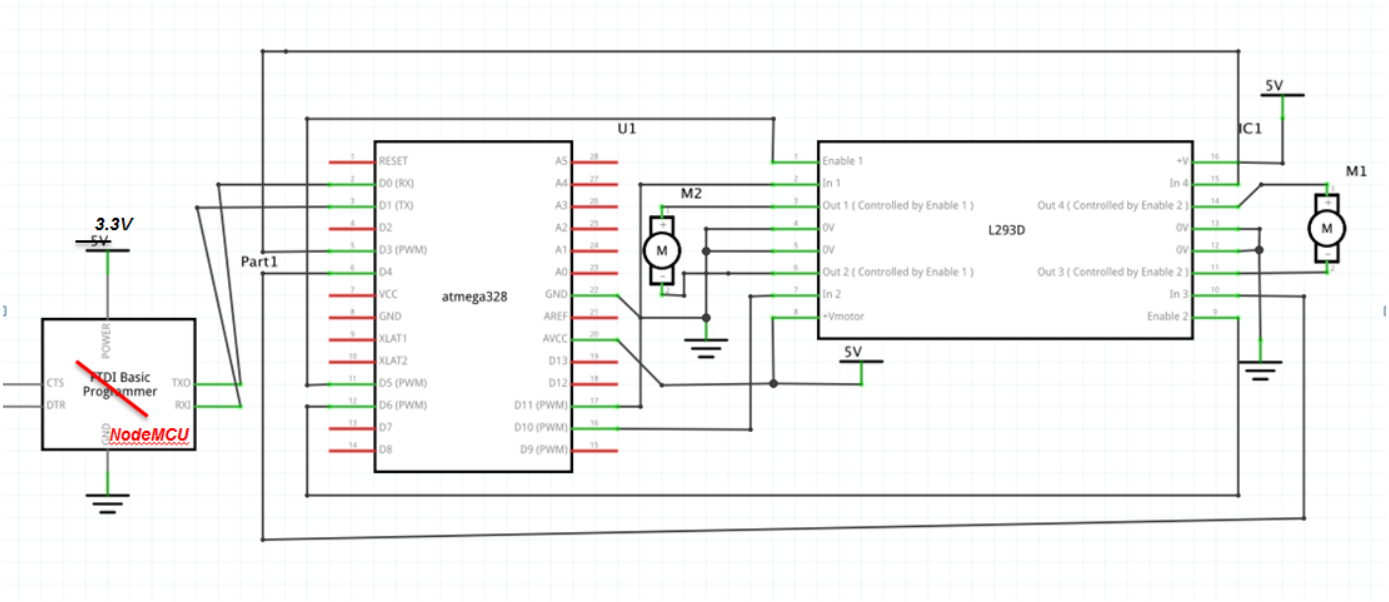
B. Node MCU

The NodeMCU is an ESP8266 wifi board.

C. L293D

The L293D IC is a motor driver. Internal of the L293D are two H-bridge motors.

II. SCHEMATICS



Implementation:

- We were given the body of a Zumo DC Motor Robot with two DC motors

- To drive the motor, four PWM outputs were used from Timer 0 and Timer 2 which connect to the H-Bridge driver in the L293D chip.
- USART function was needed to connect from the NodeMCU to the Blink App. The Node MCU was programmed with the Arduino IDE to connect to a set Wifi.
- The functions of Motor Driver and USART configuration were made into header files and implemented as function calls into the main code

-

III. Code

MOTOR_CONTROL_CODE

```
#ifndef DRIVE_MOTOR_H
#define DRIVE_MOTOR_H

#include <avr/io.h>

#define PMW0A PORTD6
#define PMW0B PORTD5
#define PMW2A PORTB3
#define PMW2B PORTD3
//! Initialize motor pins
void init_motor(){
  DDRB |= (1<<PMW2A);
  DDRD |= (1<<PMW2B);
  DDRD |= (1<<PMW0B)|(1<<PMW0A);

  // top is 255
  TCCR0A = (1<<COM0A1)|(1<<COM0B1)|(1<<WGM00)|(1<<WGM01);   // NON INVERTING MODE
  TCCR0B = (1<<CS00);                    // no Pre-scalar

  TCCR2A = (1<<COM2A1)|(1<<COM2B1)|(1<<WGM20)|(1<<WGM21);   // NON INVERTING MODE
  TCCR2B = (1<<CS20);                    // no Pre-scalar

}

//! Allows left & right wheel movements (forwards, backwards)
void drive_motor(int left_pct, int right_pct){
  int tempL = (int)(left_pct/100.0 * 255);
  int tempR = (int)(right_pct/100.0 * 255);

  if (tempL>=0 && tempR>=0){
    OCR0A = 0;                      // no movement in other direction
    OCR0B = (tempL)<255?tempL:255;          // change duty cycle
    OCR2A = 0;                      // no movement in other direction
    OCR2B = (tempR)<255?tempR:255;          // change duty cycle

  }
  else if (tempL>=0 && tempR<0){
    OCR0A = 0;                      // no movement in other direction
    OCR0B = (tempL)<255?tempL:255;          // change duty cycle
    OCR2A = (-tempR)<255?(-tempR):255;        // no movement in other direction
    OCR2B = 0;                      // change duty cycle

  }
  else if (tempL<0 && tempR>=0){
    OCR0A = (-tempL)<255?(-tempL):255;        // no movement in other direction
    OCR0B = 0;                      // change duty cycle
    OCR2A = 0;                      // no movement in other direction
    OCR2B = (tempR)<255?tempR:255;          // change duty cycle

  }
  else{
    OCR0A = (-tempL)<255?-tempL:255;        // no movement in other direction
    OCR0B = 0;                      // change duty cycle
    OCR2A = (-tempR)<255?-tempR:255;        // no movement in other direction
```

```
    OCR2B = 0;                    // change duty cycle
  }
}
/**@}*/
#endif
```

USART INITIALIZATION CODE

```
#include <avr/io.h>
#include <util/delay.h>


Initialize UART in the Atmega328p
volatile signed char receivedChar;
char charBuff[BUFF_SIZE];
volatile unsigned char charread;
int l, r;
char turnt = 0, done = 0;


void initUART(){
  unsigned int baudrate;

  // Set baud rate:  UBRR = [F_CPU/(16*BAUD)] -1
  baudrate = ((F_CPU/16)/BAUD) - 1;
  UBRR0H = (unsigned char) (baudrate >> 8);
  UBRR0L = (unsigned char) baudrate;

  UCSR0B |= (1 << RXEN0) | (1 << TXEN0);   // Enable receiver & transmitter
  UCSR0C |= (1 << UCSZ01) | (1 << UCSZ00); // Set data frame: 8 data bits, 1 stop bit, no parity
}

//! Transmit/write one character to the output
void writeChar(unsigned char c) {
  UDR0 = c;     // Display character on serial (i.e., PuTTY) terminal
  _delay_ms(10);    // delay for 10 ms
}

//! Transmit/write a NULL-terminated string to the output
void writestring(char *c){
  unsigned int i = 0;
  while(c[i] != 0)
  writeChar(c[i++]);
}

void readString(){
  done = 0;
  while (done == 0){
    if(UCSR0A & (1 << RXC0)){

      receivedChar = UDR0;       // Read the data from the RX buffer
      charBuff[charread] = receivedChar; // load char into buffer

      if(receivedChar == '<')      // esp32 has stopped sending gibberish
      return;

      else if(receivedChar == 'l')   // left value input ready
      turnt = 1;

      else if(receivedChar == 'r')   // right value input ready
```

```
      turnt = 0;

      else if(turnt){          // set right
        r = (int)receivedChar;
        done = 1;
      }
      else{              // set left
        l = (int)receivedChar;
        done = 0;
      }
    }

  }
}

/**@}*/


#endif
```

MAIN FUNCTION

```
#define F_CPU 8000000UL  //clock speed of Atmega328p - 8MHz

#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>
#include <stdlib.h>
#include "drive_motor.h"
#include "uart.h"


int main(void){
  // initialize WIFI
  initUART();
  init_motor();


}
```

ARDUINO FUNCTION

```
#define BLYNK_PRINT Serial


#include <WiFi.h>


#include <WiFiClient.h>


#include <BlynkSimpleEsp32.h>


// You should get Auth Token in the Blynk App.


// Go to the Project Settings (nut icon).


char auth[] = "52f3c4e2349b42c68b94579fd313bf1e";
```

```
// Your WiFi credentials.


// Set password to "" for open networks.
char ssid[] = "";
char pass[] = "";

BLYNK_WRITE(V1){

// variable declarations

int x = param[0].asInt();          // x direction value
int y = param[1].asInt();          // y direction value

// turning data into percentages

x /= 100;
y /= 100;

// left & right wheel percentage calculations


x = -x;


int v = (100-abs(x)) * y/100 + y;
int w = (100-abs(y)) * x/100 + x;

int r = (v+w) / 2;
int l = (v-w) / 2;

// sending joystick data onto Atmega328p

Serial.write(l);

Serial.write('l');

Serial.write(r);

Serial.write('r');

}

}
```

```
}

void setup(){

Serial.begin(9600);                        // setup UART

Blynk.begin(auth, ssid, pass); // setup Wi-Fi authentication

Serial.write('<'); // send '<' for read ready

void loop(){

Blynk.run();    // run Blynk
```

IV. Links

Youtube Video / Working Robot: https://www.youtube.com/watch?v=IHMFdks4CGg
Youtube Video / Presentation: https://www.youtube.com/watch?v=iI3Y15tvnIY&feature=youtu.be

V. CONCLUSION

We successfully created a wireless system to control the DC Zumo Robot. Using the NodeMCU we were able to establish a connection via wifi from a mobile app to controls of the microcontroller. Our biggest trouble shooting issue was interfacing the joystick on the blink app to control the direction of the motors. After much trial and error we were able to overcome this issue. For further implementation of this project, to enhance the level of creativity and create a new challenge to this bot, we can add sensors to the bot such as an ultrasonic sensor to enable the bot to "see". We could also try using PWM on servo motors to have better accuracy.