Final Lab Report

Bluetooth Motor Control System

Logan Barber, Ian Nail

May 4, 2022

Ian Nail

Logan Barber

ME-4370 - Stephen Canfield

1 Introduction

In this final lab, the objective was to use two microcontrollers connected through Bluetooth to control two "fans". The fans are two motors that are speed controlled with potentiameters. Additionally, two sensors are used to turn the motors off at any moment given object detection. This system demonstrates the remote control of motors and the use of object detection using proximity and motions sensors for a safety systems.

Several analog to digital conversions take place to operate this system. The potentiameters are analog inputs converted into a digital number between 0 and 250. Where 0 is the motor stopped and 250 is the motor at max speed. A total of two potentiameters, each controls one of the two motors.

The sensors are also analog inputs converted into a digital signals. Two sensors were used. The SHARP 2Y0A02F18 IR distance sensor and the HC-SR501 IR motion sensor. The distance sensor was used to turn off motor1 if it detected an object within 20 cm. The motion sensor was used to turn off motor2 if motion was detected in the sensors field of view.

Two HM-10 bluetooth modules were used to communicate data from system to system. The bluetooth modules were configured to pair with one another and with the serial UART TX and RX connection with the microcontroller.

Demonstration of this system can be viewed on YouTube [1].

This Lab report should contain the following:

1. Cover page (1 page) Include title, date, authors, signature, recipient 2. Introduction 3. Procedure for design 4. Procedure for development / construction 5. Assessment 6. Conclusions 7. References 8. Appendix 1: Circuit diagram 9. Appendix 2: Program Flowchart 10. Appendix 3: Program listing

2 Design

The electrical components used in the system are as follows:

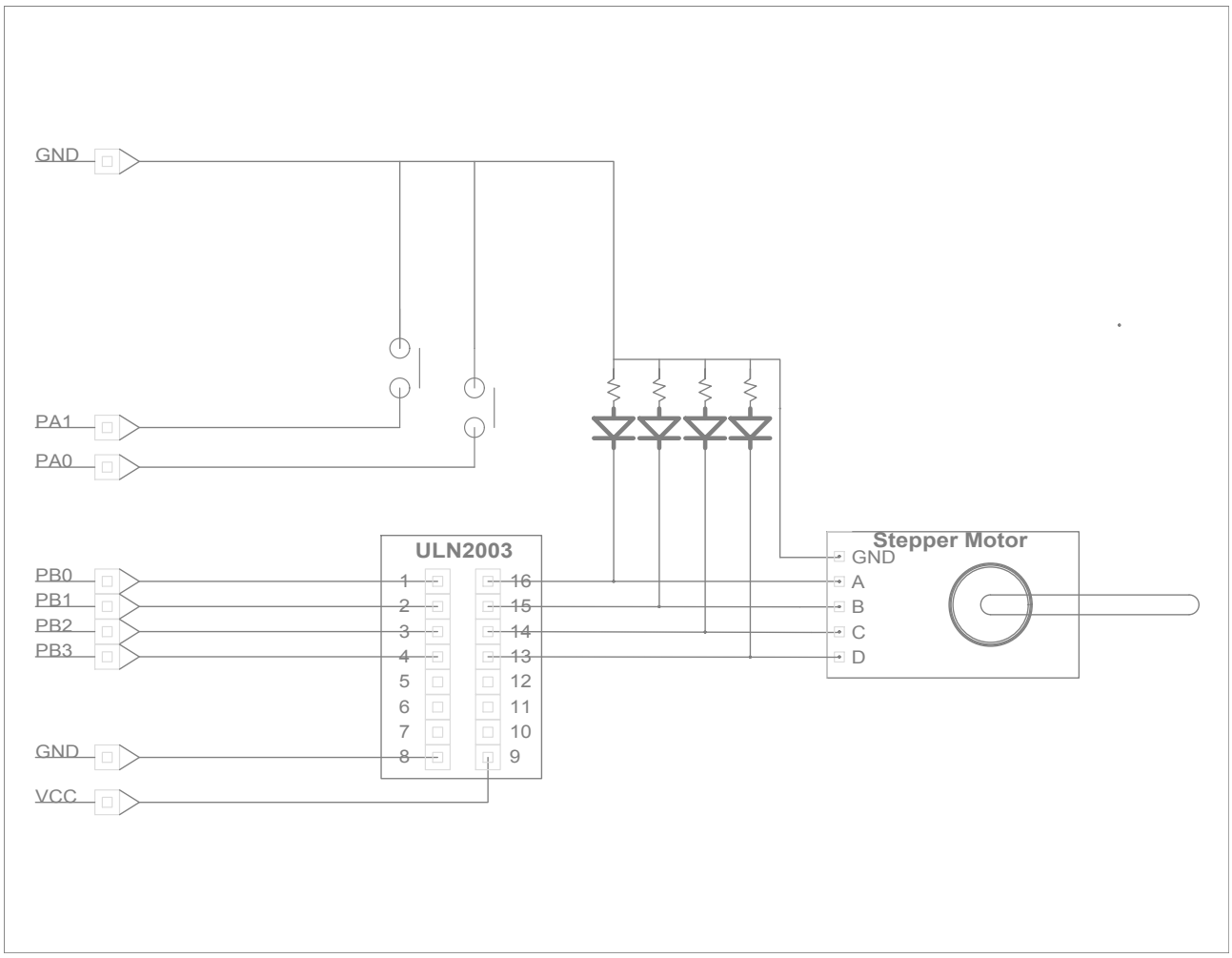

Figure 1: Circuit Diagram

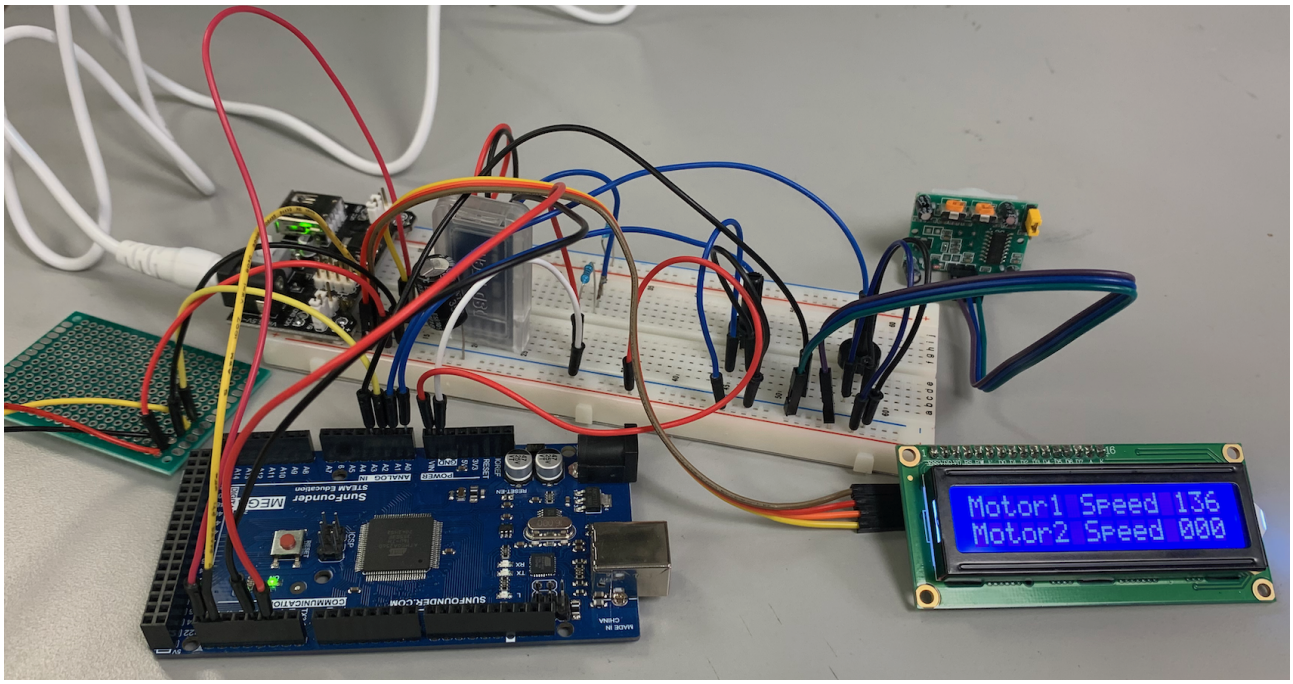

Figure 2: Breadboard and Connections

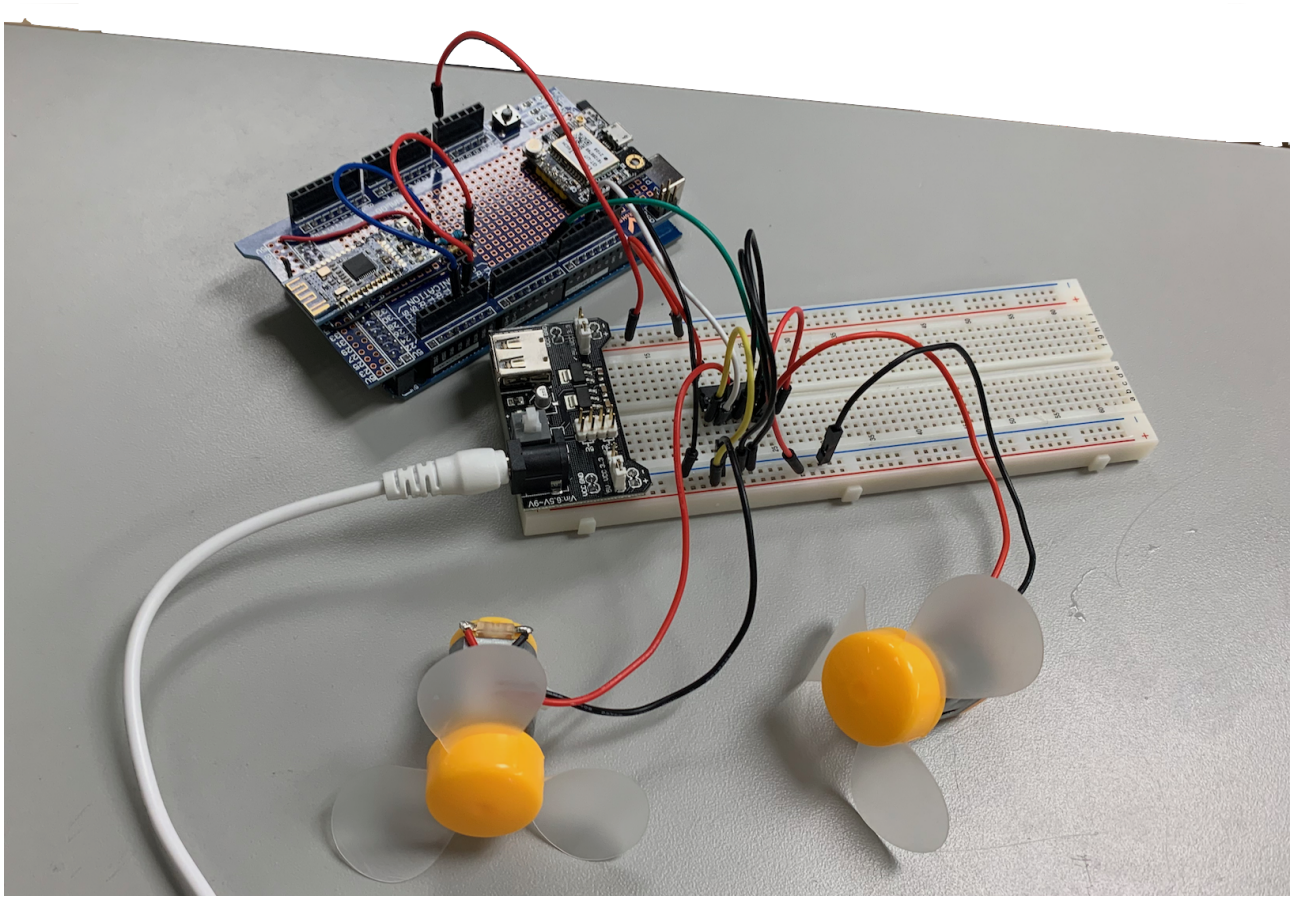

Figure 3: Breadboard and Connections

Quantity	Model	Description
2	Arduino Atmega 2560	Microcontoller
2	HM-10	Low Energy Bluetooth modules
2	10k Generic Potentiometer	Variable Resistor
2	Gikfun DC Motor	3-6 Volt DC Motor

3 Development

4 Assessment

[1] `https://youtu.be/fZUjRV97KJc`

9 Appendix iii: Source Code - Remote Control

This section contains the source code from the microcontroller reading potentiometers and the sensors to control the speed of the motors, and then sending the speeds over bluetooth to the motor control microcontroller.

```
/*

 */


/* I N C L U D E S */

#include <LiquidCrystal_I2C.h>

/* D E C L A R A T I O N S */
void u8_get_motor1_speed(void);
void u8_get_motor2_speed(void);
void average_distance(void);
void zero_pad_int_str( int n, char s[]);
void reverse_string(char s[]);
void integer_to_ascii(int n, char s[]);
void motion_detection(void);


/* D E F I N E S */
// DEFINE PIN NUMBERS
#define ADDRESS 0x27
#define COLS 16
#define ROWS 2
#define baud 9600

/* G L O B A L S */

uint8_t u8_start_byte = 255;
uint8_t u8_stop_byte = 254;
uint8_t u8_motor1_speed=50;
uint8_t u8_motor2_speed=50;
uint8_t u8_data = 192;
uint8_t i_index = 0;
uint8_t u8_command_state = 3;
uint8_t u8_motor1_state = 0;
uint8_t u8_motor2_state = 0;
uint32_t u32_average_distance = 0;
bool bool_motion_detection = false;

char message0[] = "Motor1 Speed";
char message1[] = "Motor2 Speed";
char message2[] = "000";
char message3[] = "000";

/* S E T U P */

// CREATE LiquidCrystal OBJECT
LiquidCrystal_I2C lcd(ADDRESS, COLS, ROWS);

void setup() //run once
```

```
{
52        // setup timer 1
  TCCR1A = 0b00000000; // see notes, normal mode
54  TCCR1B = 0b00000100; // normal mode divide clock by 1
  TIMSK1 = 0b00000000; // no interrupts
56
  // Start the serial connection with the HM-10 bluetooth module
58  Serial2.begin(baud,SERIAL_8N1);
  Serial.begin(baud,SERIAL_8N1);
60
     // INITIALIZE THE LCD SCREEN
62   lcd.begin();

64   // PRINT MESSAGE
     lcd.print(message0);
66   lcd.setCursor(0, 1);
     lcd.print(message1);
68
   // DISABLE DIGITAL BUFFER FOR ALL ANALOG PINS
70   DIDR0 = 0xFF;
   DIDR2 = 0xFF;
72
   // SET UP ANALOG TO DIGITIAL CONVERSION IN FREE RUNNING MODE
74   ADCSRB = 0x00; // 0b00000000

76



78
}
80

82 /* M A I N */
void loop() // run over and over
84 {

86   u8_get_motor1_speed();
   u8_get_motor2_speed();
88   average_distance();
   motion_detection();
90   if( u32_average_distance > 400){
     u8_motor1_speed = 0;
92   }

94   if( bool_motion_detection == true){
     u8_motor2_speed = 0;
96   }


98


100 //  SENDING MOTOR1 AND MOTOR2 SPEED TO BLUETOOTH
   if( TIFR1 &= 0b00000001 ) {
102  TIFR1 |= 0b000000001; // clear the TOV flag by writing a 1 to it

104  //Write to motor control
   Serial2.write(u8_start_byte);
106  // MOTOR 1
   Serial2.write(u8_motor1_speed);
108  lcd.setCursor(13, 0);
   integer_to_ascii(u8_motor1_speed,message2);
110  zero_pad_int_str(3,message2);
```

```
    lcd.print(message2);

        // MOTOR 2
    Serial2.write(u8_motor2_speed);
   lcd.setCursor(13, 1);
   integer_to_ascii(u8_motor2_speed,message3);
    zero_pad_int_str(3,message3);
    lcd.print(message3);

    Serial.println(u8_motor1_speed);
    Serial.println(u8_motor2_speed);
    Serial.println(u32_average_distance);

    }
}



/* F U N C T I O N S */

//  ANALOG TO DIGITIAL CONVERSION FUNCTION FOR CAPTURING MOTOR1 SPEED FROM
   POTENTIMETER
void u8_get_motor1_speed(void){
    uint16_t valueADC0 = 0;
  // READ THE INPUT ON ANALOG CHANNEL 0
   ADMUX = 0x40; // 0b01000000;

    // ENABLE THE ADC, RESET ADIF, AND SET SAMPLING RATE TO 125 kHz
    ADCSRA = 0x97; // 0b10010111
    // START THE CONVERSION
    ADCSRA = 0xD7; // 0b11010111

    // WAIT HERE DURING THE CONVERSION
    while((ADCSRA & 0x10) == 0x00)
    {
        asm("NOP");
    }

    // IF THE CONVERSION IS COMPLETE
    if((ADCSRA & 0x10) == 0x10)
    {
        // DISABLE THE ADC, RESET ADIF, AND SET SAMPLING RATE TO 125 kHz
        ADCSRA = 0x17; // 0b00010111
        // READ THE ADC VALUE
        valueADC0 = ADCL;
        valueADC0 += (ADCH << 8);
    }

    if(valueADC0 <= 50){
      u8_motor1_speed = 0;
    }
    else if(valueADC0 > 50){
      u8_motor1_speed = (valueADC0)/4+20;
    }

}

//  ANALOG TO DIGITIAL CONVERSION FUNCTION FOR CAPTURING MOTOR2 SPEED FROM
   POTENTIMETER
void u8_get_motor2_speed(void){
```

```
    uint16_t valueADC1 = 0;
    // READ THE INPUT ON ANALOG CHANNEL 1
    ADMUX = 0x41; // 0b01000001;
    // ENABLE THE ADC, RESET ADIF, AND SET SAMPLING RATE TO 125 kHz
    ADCSRA = 0x97; // 0b10010111
    // START THE CONVERSION
    ADCSRA = 0xD7; // 0b11010111
    // WAIT HERE DURING THE CONVERSION
    while((ADCSRA & 0x10) == 0x00)
    {
        asm("NOP");
    }

    // READ THE CONVERSION VALUE
    if((ADCSRA & 0x10) == 0x10)
    {
        // DISABLE THE ADC, AND RESET ADIF, AND SET SAMPLING RATE TO 125 kHz
        ADCSRA = 0x17; // 0b00010111
        // GET THE HIGH BITS
        valueADC1 = ADCL;
        valueADC1 += (ADCH << 8);
    }

    if(valueADC1 <= 50){
      u8_motor2_speed = 0;
    }
    else if(valueADC1 > 50){
      u8_motor2_speed = (valueADC1)/4 + 20;
    }

}

//  ANALOG TO DIGITIAL CONVERSION FUNCTION FOR CAPTURING DISTANCE RF SENSOR
void average_distance(void){
    uint16_t u16_index;
    uint16_t valueADC2 = 0;
    // READ THE INPUT ON ANALOG CHANNEL 2
    ADMUX = 0x42; // 0b01000010;
    // ENABLE THE ADC, RESET ADIF, AND SET SAMPLING RATE TO 125 kHz
    ADCSRA = 0x97; // 0b10010111

for(u16_index = 0; u16_index < 20; u16_index++) {

    // START THE CONVERSION
    ADCSRA = 0xD7; // 0b11010111
    // WAIT HERE DURING THE CONVERSION
    while((ADCSRA & 0x10) == 0x00)
    {
        asm("NOP");
    }
    // READ THE CONVERSION VALUE
    if((ADCSRA & 0x10) == 0x10)
    {
        // DISABLE THE ADC, AND RESET ADIF, AND SET SAMPLING RATE TO 125 kHz
        ADCSRA = 0x17; // 0b00010111
        // GET THE HIGH BITS
        valueADC2 = ADCL;
        valueADC2 += (ADCH << 8);
    }
    u32_average_distance = u32_average_distance + valueADC2;
```

```
}
u32_average_distance = u32_average_distance/20;
}


void motion_detection(void){

    uint16_t valueADC3 = 0;
    // READ THE INPUT ON ANALOG CHANNEL 2
    ADMUX = 0x43; // 0b01000011;
    // ENABLE THE ADC, RESET ADIF, AND SET SAMPLING RATE TO 125 kHz
    ADCSRA = 0x97; // 0b10010111

    // START THE CONVERSION
    ADCSRA = 0xD7; // 0b11010111
    // WAIT HERE DURING THE CONVERSION
    while((ADCSRA & 0x10) == 0x00)
    {
        asm("NOP");
    }
    // READ THE CONVERSION VALUE
    if((ADCSRA & 0x10) == 0x10)
    {
        // DISABLE THE ADC, AND RESET ADIF, AND SET SAMPLING RATE TO 125 kHz
        ADCSRA = 0x17; // 0b00010111
        // GET THE HIGH BITS
        valueADC3 = ADCL;
        valueADC3 += (ADCH << 8);
    }
   if(valueADC3 > 500){
    bool_motion_detection = true;
   }
   else {
    bool_motion_detection = false;
   }
}



/* integer_to_ascii:  convert n to characters in s */
void integer_to_ascii(int n, char s[])
{
  int i, sign;

  if ((sign = n) < 0) /* record sign */
    n = -n;   /* make n positive */
  i = 0;
  do {    /* generate digits in reverse order */
    s[i++] = n % 10 + '0'; /* get next digit */
  } while ((n /= 10) > 0);  /* delete it */
  if (sign < 0)
    s[i++] = '-';
  s[i] = '\0';
  reverse_string(s); // needs to be reversed
}

/* reverse_string:  reverse string s in place */
void reverse_string(char s[])
{
  int i, j;
```

```
  char c;

  for (i = 0, j = strlen(s)-1; i<j; i++, j--) {
    c = s[i];
    s[i] = s[j];
    s[j] = c;
  }
}
/************* Zero pad a string by n amount. If myString = "5" and
   zero_pad_int_str(4,myString) then myString  == "0005" *************/
void zero_pad_int_str( int n, char s[])
    {

      int i;
        i = strlen(s);
        reverse_string(s);
        while(i < n){
          s[i] = '0';
          i++;
        }
        reverse_string(s);

    }
```

../Remote_control.c

10 Appendix iv: Source Code - Motor Control

This section contains the source code from the microcontroller controlling the two motors.

```
/*
 * The circuit:
 * RX is digital pin rxPin (connect to TX of other device)
 * TX is digital pin txPin (connect to RX of other device)
 *
 */

 /* I N C L U D E S */

/* D E F I N E S */

#define baud 9600
#define DELAY 500


/* G L O B A L S */
uint8_t u8_command_state = 0;
uint8_t u8_motor1_speed = 0;
uint8_t u8_motor2_speed = 0;
uint8_t u8_motor1_state = 0;
uint8_t u8_motor2_state = 0;
uint8_t u8_start_byte = 193;
uint8_t u8_stop_byte = 194;

/* D E C L A R A T I O N S */
```

```
/* S E T U P */
void setup() //run once
{
    // setup timer 1
    TCCR1B = 0b00000100; // normal mode divide clock by 1
    TIMSK1 = 0b00000000; // no interrupts

    // Start the connection to the bluetooth module
    Serial1.begin(baud,SERIAL_8N1);

    // SET UP PINS FOR OUTPUT
    DDRB |= 1<<PB7;
    DDRH |= 1<<PH6;

    // USE TCNT1
    TCCR0A = 0b10000011; // non-inverted mode, 8 bit fast PWM
    TCCR0B = 0b00001100; // (I/O clk)/256 (From prescaler)

    // USE TCNT2
    TCCR2A = 0b10000011; // non-inverted mode, 8 bit fast PWM
    TCCR2B = 0b00001100; // (I/O clk)/256 (From prescaler)

    // PWM COUNTER
    OCR0A = 0;
    OCR2B = 0;


}

/* M A I N */
void loop() // run over and over
{

    if(Serial1.available()){

      if(u8_command_state=1){
        u8_motor1_speed =  Serial.read();
      }
      if(u8_command_state=2){
        u8_motor2_speed =  Serial.read();
      }

    }
    OCR0A = u8_motor1_speed;
    OCR2B = u8_motor2_speed;




    if( TIFR1 &= 0b00000001 ) {
    TIFR1 |= 0b000000001; // clear the TOV flag by writing a 1 to it
    Serial1.write(u8_start_byte); // Start signature
    Serial1.write(u8_motor1_state);
    Serial1.write(u8_motor2_state);

    }

}
```

../Motors.c