Lego RC CAR

Introducere

Proiectul constă într-o mașinuță LEGO Technic Porsche 911 RSR controlată la distanță, prin Bluetooth, printr-o aplicație de telefon.

Scopul este ca mașina să se poată deplasa la fel ca una normală, înainte, înapoi, cât și să își schimbe direcția prin rotirea roților. Va dispune și de un set de leduri albe, galbene și roșii, pentru a imita farurile, stop-urile, avariile și semnalizarea unei mașini reale. De asemenea, aceasta va avea un senzor ultrasonic care va funcționa ca un senzor de parcare (va declanșa un buzzer dacă mașina este prea aproape de un obstacol, buzzer-ul va emite din ce în ce mai multe semnale pe masură ce distanța devine mai mică).

Ideea a pornit de la un prieten, care are mai multe astfel de mașinuțe și care îmi tot spunea că a văzut oameni pe net care au motorizat astfel de jucării. Mi-a spus că ar vrea și el să facă asta și am zis de ce nu.

Proiectul este util în primul rând pentru că ar acoperi o varietate de concepte studiate la această materie și în al doilea rând e un Porsche LEGO teleghidat, care ar putea (dacă iese bine) să facă drift-uri.

Descriere generală

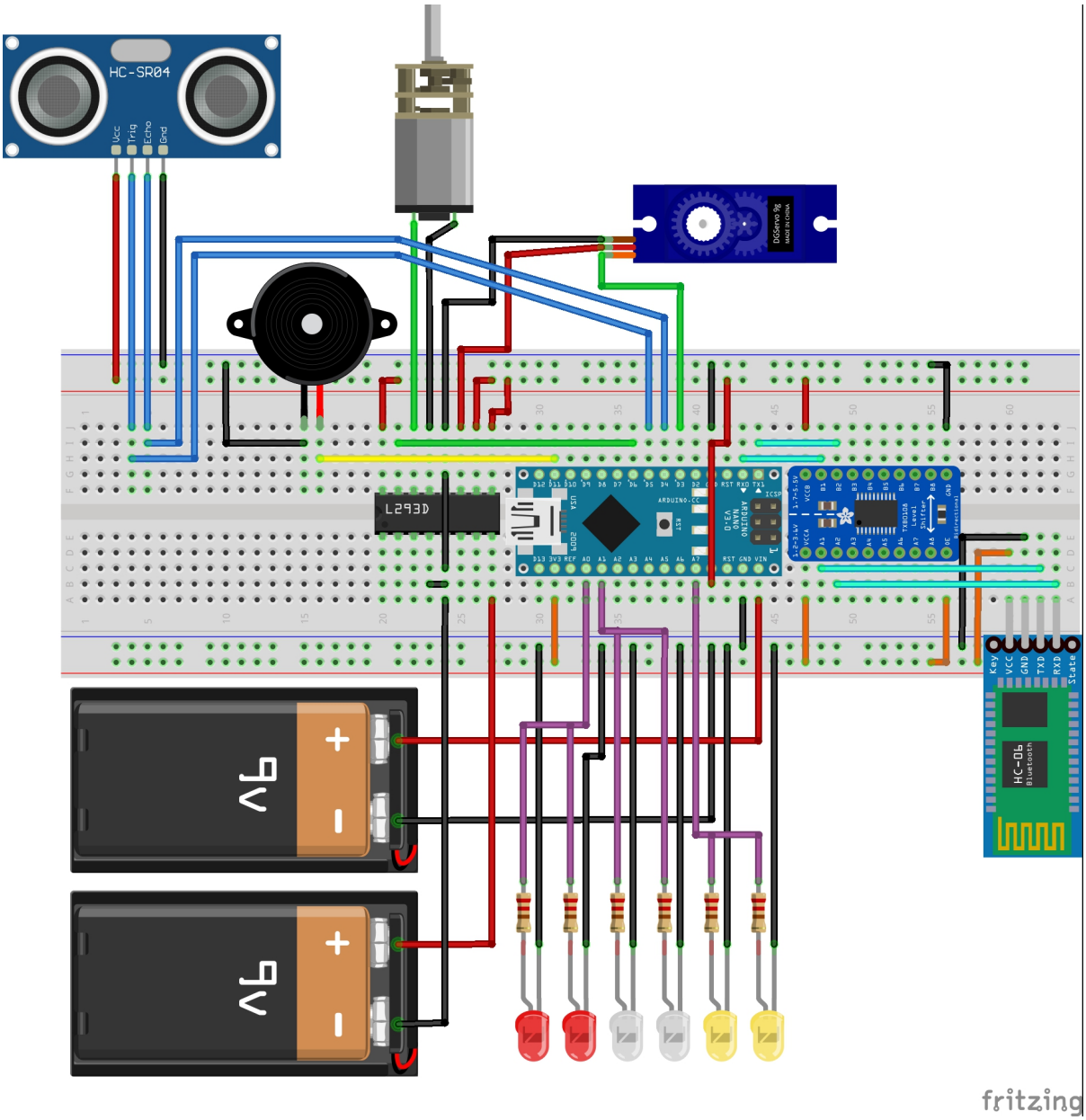
Hardware Design

Listă piese

- Arduino Nano ATmega328P
- Modul Bluetooth JDY-18
- Convertor nivel logic TXS0108E
- Driver motoare L293D
- Motor JGA25-370
- Senzor ultrasonic HC-SR04
- Fotorezistor
- Buzzer

- LED-uri albe, galbene și roșii
- Rezistențe
- 2x Baterii 9V

Schema electrică



Software Design

USART

- Pentru codul mașinii am folosit PlatformIO și avr-libc, repo GitHub
- Pentru codul aplicației de iOS am folosit Xcode, SwiftUI și CoreBluetooth, repo GitHub

Pentru comunicarea Bluetooth am folosit modulul USART0 al microcontrolerului, configurat să declanșeze o întrerupere la finalul fiecărei recepții, astfel încât să pună caracterele primite într-un buffer și să seteze un flag atunci când a detectat finalul unui mesaj (când a detectat newline).

```
void USART0_init(unsigned int ubrr)
{
    /* baud rate registers */
    UBRR0H = (unsigned char)(ubrr>>8);
    UBRR0L = (unsigned char)ubrr;

    /* enable TX and RX */
    UCSR0B = _BV(RXEN0) | _BV(TXEN0);
    // enable receive complete interrupt
    UCSR0B |= _BV(RXCIE0);

    /* frame format: 8 bits, 2 stop, no parity */
    UCSR0C = _BV(USBS0) | (3<<UCSZ00);
}

...

ISR(USART_RX_vect) {
    // read data register into buffer and increase its length
    uint8_t new_data = UDR0;
    usart_buffer[usart_buffer_len++] = new_data;

    // reset buffer length on overflow
    if (usart_buffer_len >= USART_BUFFER_MAX_LEN - 1) {
        usart_buffer_len = 0;
    }

    // set string received flag
    if (new_data == '\n') {
        string_received = true;
    }
}
```

ADC

Pentru a putea detecta lumina din mediul înconjurător cu ajutorul unui fotorezistor am folosit modulul de convertor analogic digital, care să interpreteze practic lumina ca o valoare pe 10 biți (0-1023).

```
void adc_init()
{
```

```
    // channel 7
    ADMUX |= _BV(MUX0) | _BV(MUX1) | _BV(MUX2);
    // AVcc with external capacitor at AREF PIN
    ADMUX |= _BV(REFS0);

    ADCSRA = 0;
    // set 128 prescaler
    ADCSRA |= _BV(ADPS0);
    ADCSRA |= _BV(ADPS1);
    ADCSRA |= _BV(ADPS2);
    // enable ADC
    ADCSRA |= _BV(ADEN);
}

uint16_t adc_get_light_value()
{
        // start conversion
        ADCSRA |= (1 << ADSC);

        // wait until conversion is complete
        while ((ADCSRA & (1 << ADSC)));

        return ADC;
}
```

Timere/PWM

Am folosit toate cele 3 Timere ale ATMega328P, fiecare dintre ele având un scop esențial în funcționarea mașinii.

Timer 0 PWM

Primul timer este configurat în modul Phase Correct PWM și mă folosesc de cei 2 pini PWM ai săi (OC0A și OC0B) pentru a controla motorul mașinii în ambele direcții, în funcție de input-ul utilizatorului.

```
void Timer0_init_phase_correct_pwm()
{
    // set PD6(OC0A)
    // and PD5(OC1A)
    // as output
    DDRD |= _BV(PD6);
    DDRD |= _BV(PD5);

    // set Phase Correct PWM mode
    TCCR0A |= _BV(WGM00);
```

```
    // set OC0A/OC1A when up-counting
    // TCCR0A |= _BV(COM0A0);
    TCCR0A |= _BV(COM0A1);
    // TCCR0A |= _BV(COM0B0);
    TCCR0A |= _BV(COM0B1);

    // set 1024 prescaler
    TCCR0B |= _BV(CS00);
    TCCR0B |= _BV(CS02);

    OCR0A = 0;
    OCR0B = 0;
}
```

Timer 1 Input Capture

Pentru a putea detecta timpul scurs de la emiterea semnalului ultrasonic de către senzorul de distanță, care este de regulă foarte scurt (poate fi de ordinul microsecundelor), am setat contorul timer-ului 1 pe 0 la momentul emiterii semnalului de trigger și am folosit funcția de input capture a timer-ului 1, pentru a putea declanșa o întrerupere la detectarea unui front negativ pe pinul ICP1.

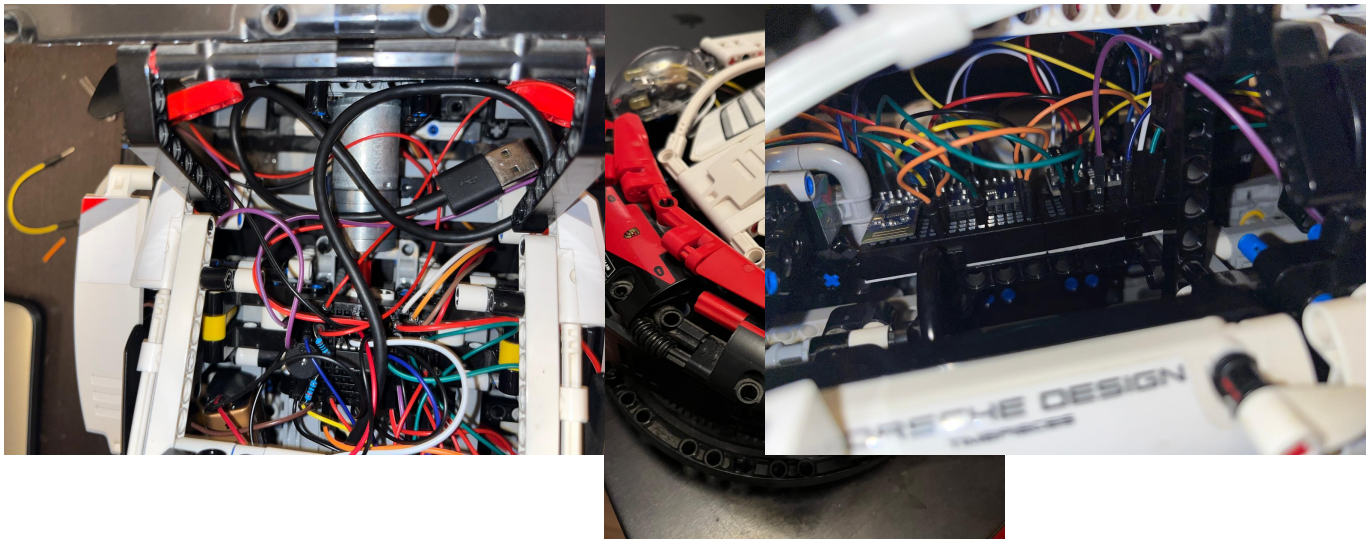
```
void Timer1_init_input_capture()
{
    cli();

    // OC1A/OC0B disconnected, normal mode
    TCCR1A = 0;

    // set input capture noise canceler
    TCCR1B |= _BV(ICNC1);
    // set falling edge trigger
    TCCR1B &= ~_BV(ICES1);

    // set 8 prescaler
    TCCR1B |= _BV(CS11);

    // set input capture interrupt
    TIMSK1 |= _BV(ICIE1);

    sei();
}

ISR(TIMER1_CAPT_vect)
{
    ultrasonic_sensor_micros = ICR1 / 2;
    distance_updated = true;
}
```

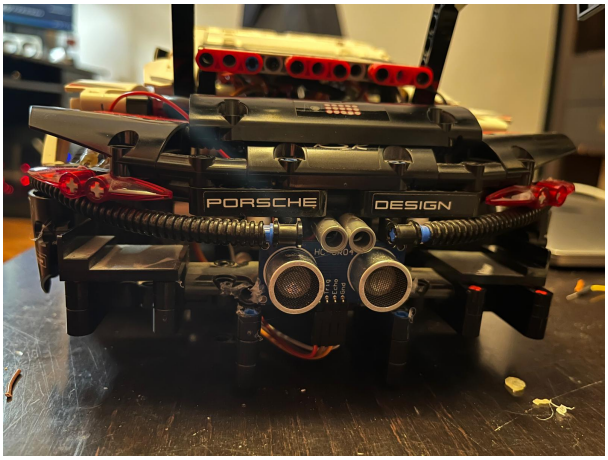
Timer 2 1Khz

Timer-ului 2 rulează în modul CTC pentru a putea oferi o frecvență de 1ms și deservește o bună parte din activitățile mașinii: pulsară regulată a luminilor de semnalizare, ping-urile emise de buzzer în momentul detectării unor obstacole, declanșarea senzorilor pentru a obține valori actualizate de la aceștia de 4 ori pe secundă.

```
void Timer2_init_1Khz()
{
    cli();

    // set CTC mode
    TCCR2A |= _BV(WGM21);

    // set 64 prescaler
    TCCR2B |= _BV(CS22);

    // set compare register for 1000Hz
    OCR2A |= 250;

    // set interrupt
    TIMSK2 |= _BV(OCIE2A);

    sei();
}
```

Rezultate Obţinute





Concluzii

Proiectul a fost mult mai dificil decât mă așteptam, atât pe partea de dezvoltare a codului mașinii și a aplicației, dar mai ales pe partea de hardware și de mecanică, pentru că nu aveam multă experiență cu astfel de lucruri, însă am învățat multe pe parcurs și am reușit să aduc proiectul într-o stare de funcționare bună.

Download

- Arhivă cod mașinuță:rc-lego-car-main.zip
- Arhiva cod aplicație iOS:rc-lego-car-ios-main.zip

Bibliografie/Resurse

Resurse Hardware

- Datasheet ATMega328P atmel-7810-automotive-microcontrollers-atmega328p_datasheet.pdf
- Datasheet TXS108E txs0108e.pdf
- Datasheet JDY-18 jdy-18.pdf
- Datasheet L293D l293d.pdf

Resurse Software

- SwiftUI
- CoreBluetooth

Export to PDF

From:
http://ocw.cs.pub.ro/courses/ - **CS Open CourseWare**

Permanent link:
http://ocw.cs.pub.ro/courses/pm/prj2024/vstoica/alexandru.ciornei

Last update: **2024/05/27 10:24**