



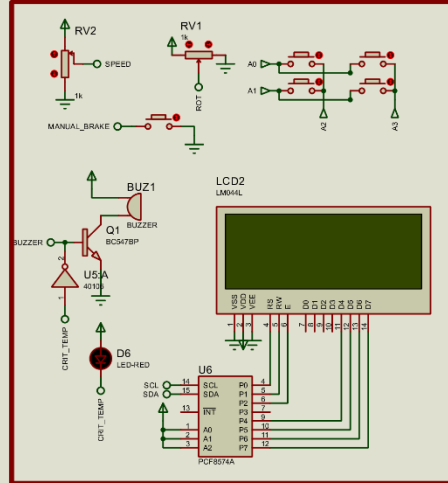
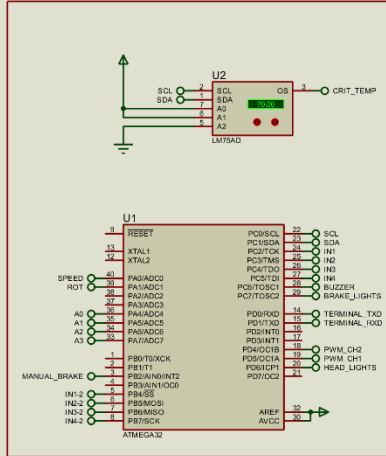
گزارش کار پروژه پایانی آزمایشگاه دیجیتال ۲

سیستم کنترل ماشین بر پایه ATMEGA32

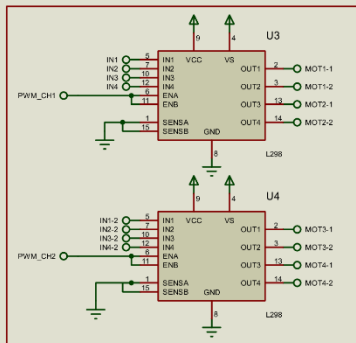
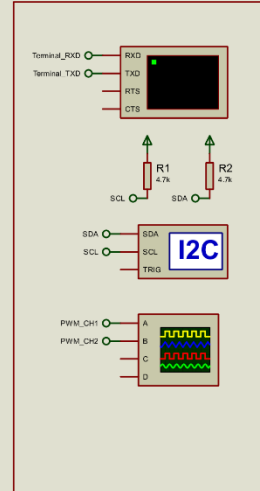
مهرزاد گلابی ۴۰۰۲۴۹۰۴۹

۱. شماتیک پروژه

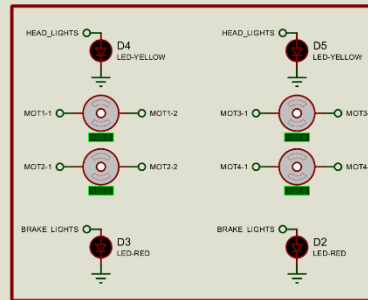
DISPLAY & CONTROL



DEBUGGING



DRIVERS



MOTORS

۲. اجزای شبیه سازی در Proteus

سیستم از ۴ بخش اصلی تشکیل شده است و هر بخش، شامل زیر اجزایی هستند که عملکرد سیستم را مهیا می سازند.

کنترلر

۱. ATMEGA32

۲. دماسنج LM75AD با امکان اتصال I2C یا TWI

درایور

۲ عدد L298 (DUAL H-BRIDGE DC MOTOR DRIVER)

موتورها

۲ عدد MOTOR-DC که تنظیمات اندوکتانس و وزن آن اصلاح شده است.

نمایش و کنترل

۱. پتانسیومتر کنترل سرعت

۲. پتانسیومتر کنترل جهت (فرمان)

۳. کیپد 2X2

۴. بازر و هشدار

۵. LED هشدار دما

۶. LCD 20X4 متصل شده به PCF8574A برای امکان اتصال I2C یا TWI

۷. ترمز دستی با استفاده از LATCH نرم افزاری

DEBUGGING

۱. VIRTUAL TERMINAL

۲. I2C SCANNER

۳. OSCILOSCOPE

۳. عملکرد کلی

در شروع شبیه سازی، پیام ابتدایی در LCD به نمایش در خواهد آمد. سپس به ترتیب مراحل زیر در کد انجام خواهد شد:

۱. مقدار های ADC ها خوانده شده و مقادیر مورد نیاز از آن ها حساب میشوند
۲. با استفاده از USART برای DEBUG کردن، متغیر های کلیدی در ترمینال مجازی به نمایش در خواهد آمد.
۳. کیپد که به صورت نرم افزاری DEBOUNCE شده است، خوانده شده و بر اساس مقدار کلید فشرده شده عملکرد مورد نیاز را انجام خواهد داد. (CASE SWITCH)
۴. ترمز دستی به صورت وقفه تعبیه شده است، در این مرحله از کد در صورتی که مقدار متغیر turn_brake_light صحیح باشد، چراغ ترمز روشن خواهد شد.
۵. تابع updateMotorControl اجرا میشود. این تابع وظیفه تمام کنترل سرعت و جهت هر ۴ موتور را دارد.
۶. در صورت حرکت به عقب، بازر بوق خواهد زد.
۷. تابع lcd_display که وظیفه نمایش تمام محتوا LCD را دارد در این بخش در هر ۱ ثانیه یک بار به روز رسانی میشود.
۸. برای ۱۰۰ میلی ثانیه صبر میکند و while دوباره اجرا میشود.

۴. تنظیمات اولیه در CodeVisionAVR

ATMEGA32 در فرکانس کاری 8MHz تنظیم شده است.

وضعیت ورودی و خروجی ها بر اساس شماتیک کشیده در پروتئوس تنظیم شده است.

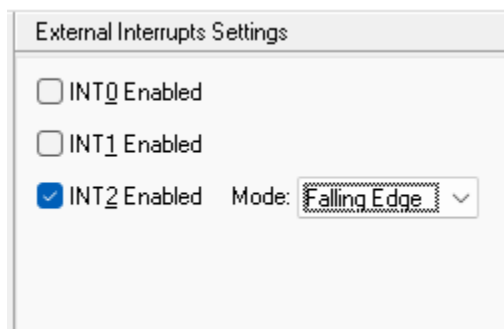
```
// Input/Output Ports initialization
{
// Port A initialization
// Function: Bit7=In Bit6=In Bit5=out Bit4=out Bit3=In Bit2=In Bit1=In Bit0=In
DDRA=(0<<DDA7) | (0<<DDA6) | (1<<DDA5) | (1<<DDA4) | (0<<DDA3) | (0<<DDA2) | (0<<DDA1) | (0<<DDA0);
// State: Bit7=T Bit6=T Bit5=T Bit4=T Bit3=T Bit2=T Bit1=T Bit0=T
PORTA=(0<<PORTA7) | (0<<PORTA6) | (1<<PORTA5) | (1<<PORTA4) | (0<<PORTA3) | (0<<PORTA2) | (0<<PORTA1) | (0<<PORTA0);

// Port B initialization
// Function: Bit7=Out Bit6=Out Bit5=Out Bit4=Out Bit3=Out Bit2=Out Bit1=In Bit0=In
DDRB=(1<<ddb7) | (1<<ddb6) | (1<<ddb5) | (1<<ddb4) | (1<<ddb3) | (1<<ddb2) | (0<<ddb1) | (0<<ddb0);
// State: Bit7=0 Bit6=0 Bit5=0 Bit4=0 Bit3=1 Bit2=1 Bit1=T Bit0=T
PORTB=(0<<PORTB7) | (0<<PORTB6) | (0<<PORTB5) | (0<<PORTB4) | (1<<PORTB3) | (1<<PORTB2) | (0<<PORTB1) | (0<<PORTB0);

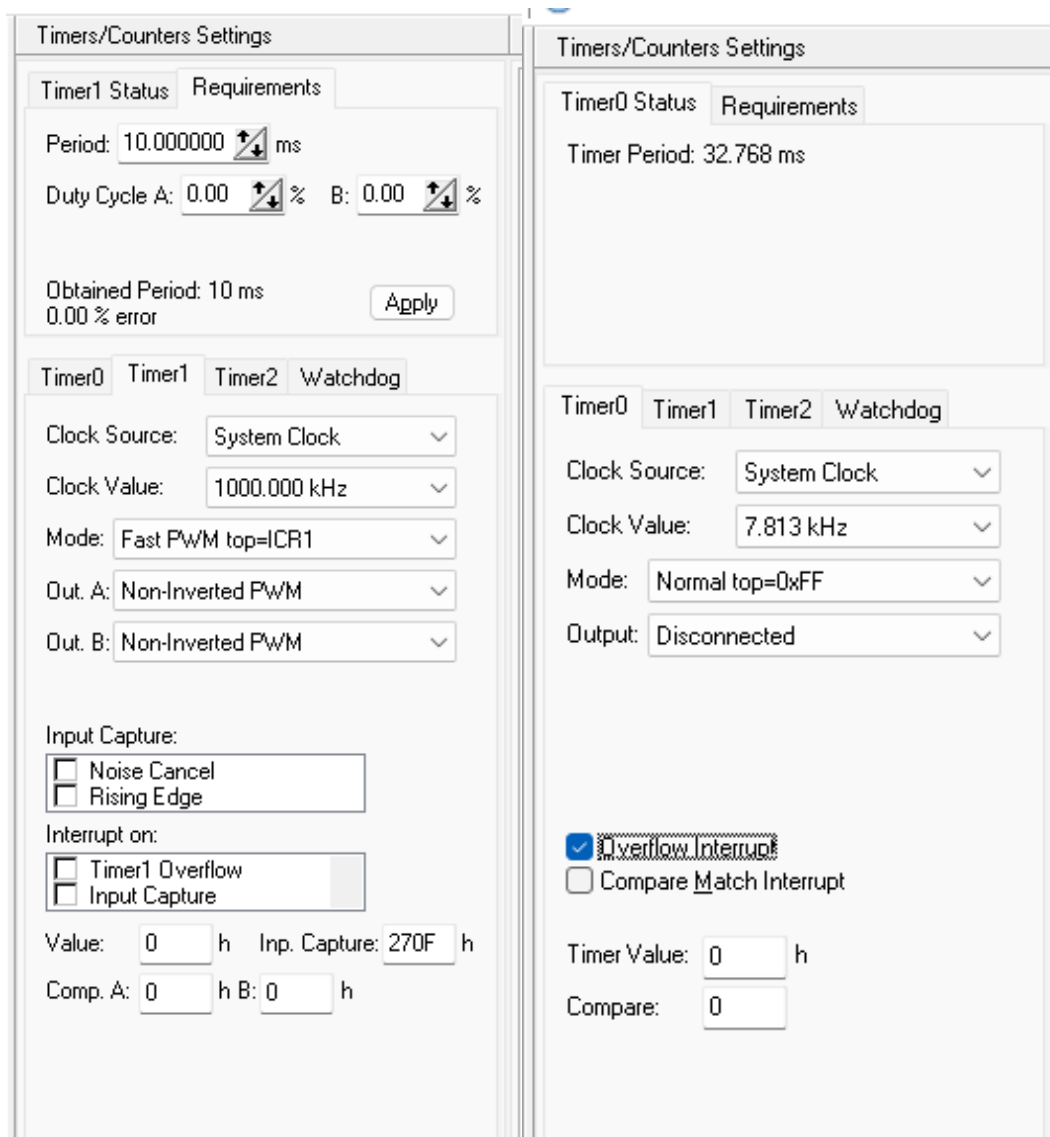
// Port C initialization
// Function: Bit7=Out Bit6=Out Bit5=Out Bit4=Out Bit3=Out Bit2=Out Bit1=In Bit0=In
DDRC=(1<<ddc7) | (1<<ddc6) | (1<<ddc5) | (1<<ddc4) | (1<<ddc3) | (1<<ddc2) | (0<<ddc1) | (0<<ddc0);
// State: Bit7=0 Bit6=0 Bit5=0 Bit4=0 Bit3=0 Bit2=0 Bit1=T Bit0=T
PORTC=(0<<PORTC7) | (0<<PORTC6) | (0<<PORTC5) | (0<<PORTC4) | (0<<PORTC3) | (0<<PORTC2) | (0<<PORTC1) | (0<<PORTC0);

// Port D initialization
// Function: Bit7=In Bit6=Out Bit5=Out Bit4=Out Bit3=In Bit2=In Bit1=In Bit0=In
DDRD=(0<<ddd7) | (1<<ddd6) | (1<<ddd5) | (1<<ddd4) | (0<<ddd3) | (0<<ddd2) | (0<<ddd1) | (0<<ddd0);
// State: Bit7=T Bit6=0 Bit5=0 Bit4=0 Bit3=T Bit2=T Bit1=T Bit0=T
PORTD=(0<<PORTD7) | (0<<PORTD6) | (0<<PORTD5) | (0<<PORTD4) | (0<<PORTD3) | (0<<PORTD2) | (0<<PORTD1) | (0<<PORTD0);
}
```

از interrupt 2 در حالت rising edge برای ترمز دستی استفاده شده است.



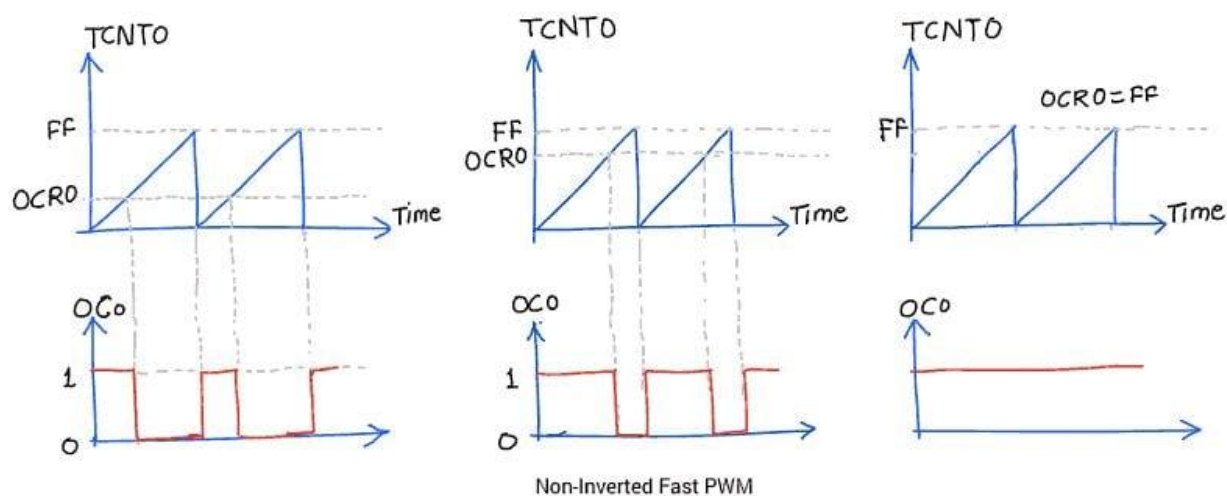
از ۲ تایمر ۰ و ۱ استفاده است. تایمر ۰ برای ساعت و به صورت کلی به دست آوردن مقدار زمان سپری شده ۱ ثانیه برای سایر بخش های کد استفاده شده است و از تایمر ۱ برای PWM در حالت FAST PWM استفاده کرده ایم:



از تایمر صفر و مد نرمال برای نشان دادن زمان کارکرد شبیه سازی استفاده شده است. وقفه ارسالی تایمر روی ۳۲.۷ میلی ثانیه تنظیم شده که با تقریب خوبی معادل هر ۳۳ بار یک ثانیه در نظر گرفت.

$$(top - bottom) \times \frac{N}{f} = 32.768ms$$

PWM در حالت Fast Pwm top=ICR1 هست که به این معنا خواهد بود که مدل pwm به صورت زیر خواهد بود:

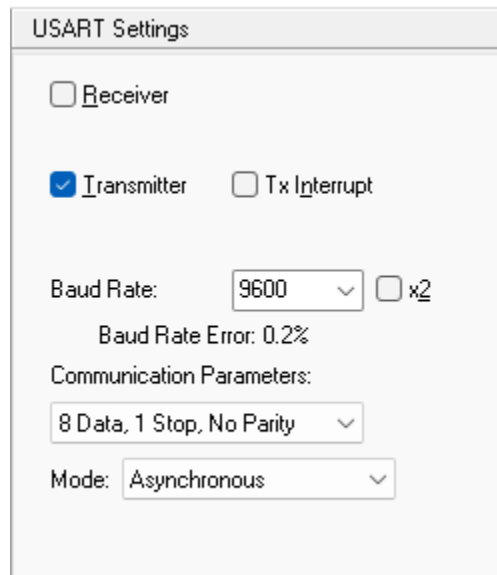


مقدار top را ICR1 مشخص میکند و مقدار OCRA و OCRB مشخص کننده DUTY CYCLE خواهند بود.

مقدار ICR1=10000 به دست می آید که در کد به صورت DEFINE تعریف شده و در بخش حساب کردن DUTY CYCLE مقدارهای مورد نیاز بر آن تقسیم می شوند.

```
// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: 1000.000 kHz
// Mode: Fast PWM top=ICR1
// OCL1A output: Non-Inverted PWM
// OCL1B output: Non-Inverted PWM
// Noise Canceler: Off
// Input Capture on Falling Edge
// Timer Period: 10 ms
// Output Pulse(s):
// OCL1A Period: 10 ms Width: 0 us
// OCL1B Period: 10 ms Width: 0 us
// Timer1 Overflow Interrupt: Off
// Input Capture Interrupt: Off
// Compare A Match Interrupt: Off
// Compare B Match Interrupt: Off
TCCR1A=(1<<COM1A1) | (0<<COM1A0) | (1<<COM1B1) | (0<<COM1B0) | (1<<WGM11) | (0<<WGM10);
TCCR1B=(0<<ICNC1) | (0<<ICES1) | (1<<WGM13) | (1<<WGM12) | (0<<CS12) | (1<<CS11) | (0<<CS10);
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x27;
ICR1L=0x10;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
OCR1BL=0x00;
```

از USART با تنظیمات پایه زیر برای نمایش مقادیر و DEBUG کردن در ترمینال مجازی استفاده شده است:



USART Settings

☐ Receiver

☒ Transmitter ☐ Tx Interrupt

Baud Rate: 9600 ☐ x2

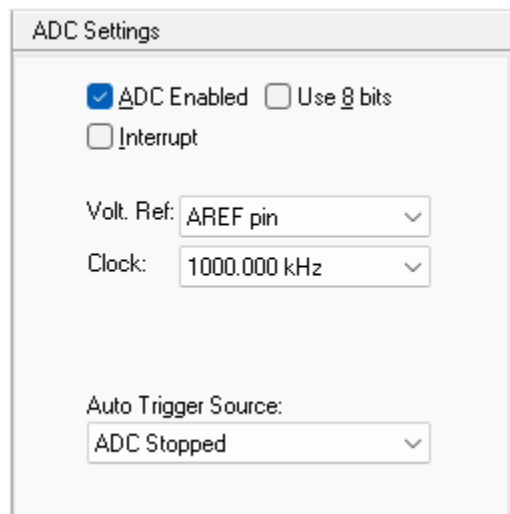
Baud Rate Error: 0.2%

Communication Parameters:

8 Data, 1 Stop, No Parity

Mode: Asynchronous

از ADC با تنظیمات زیر در کد استفاده است، AREF در شبیه سازی به مقدار VCC متصل است.



ADC Settings

☒ ADC Enabled ☐ Use 8 bits

☐ Interrupt

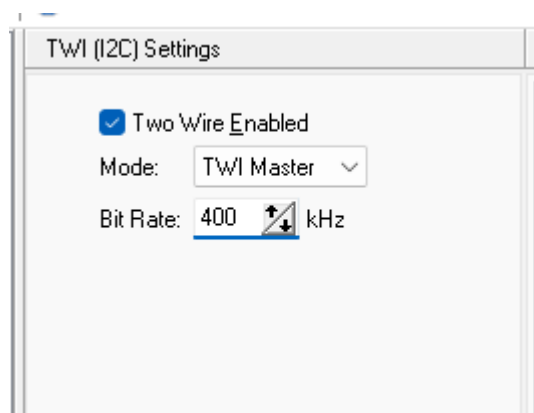
Volt. Ref: AREF pin

Clock: 1000.000 kHz

Auto Trigger Source:

ADC Stopped

برای استفاده از I2C سخت افزاری از TWO WIRE INTERFACE در تنظیمات استفاده میکنیم، کلاک آن را به ۴۰۰ کیلوهرتز افزایش داده ایم:



به دلیل استفاده از PCF8574A برای اتصال به LCD به روشن کردن Alphanumeric LCD نیازی نداریم و از کتابخانه alcd نیز استفاده نمیکنیم.