بسم الله الرحمن الرحیم

سیستم کنترل خودرو

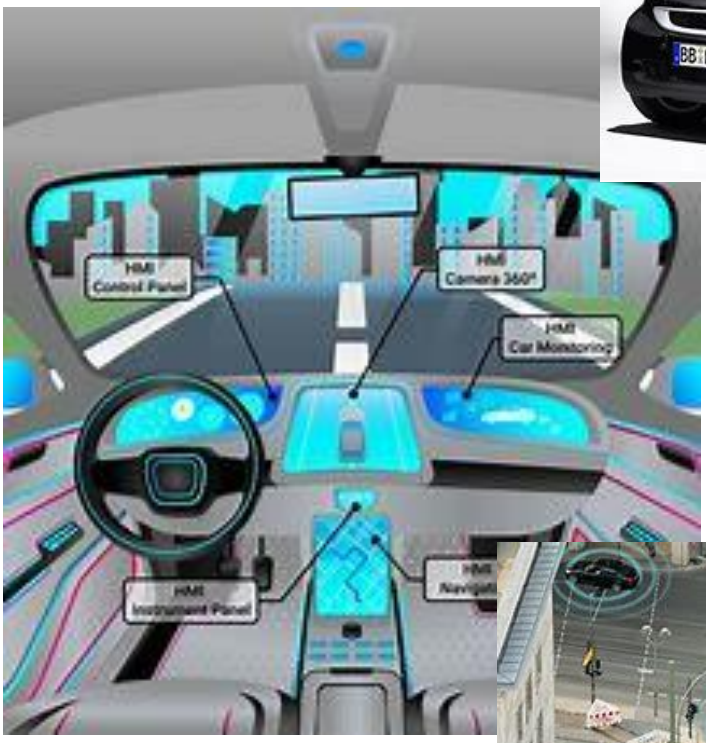
نام ارائه دهندگان: حسین جعفری 98242040

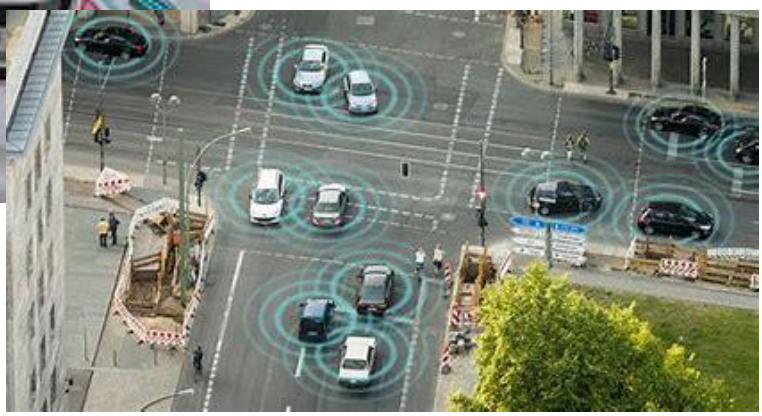
مهدی حبیبی 98242050

اساتید راهنما: استاد موسی محسن پوریان

استاد محمد ناظمی

فهرست





اعضای گروه:

1-حسین جعفری 98242040
2-مهدی حبیبی 98242050

فایل های پیوست شده در قالب avr با زبان c و اسمبلی بوده، دارای خروجی پروتئوس و به همراه فایل هگز می باشد.

ما در این پروژه سه بخش را کنترل و نظارت می کنیم که به شرح زیر است:

- کنترل موتور به صورت جداگانه
- کنترل مسیر حرکت موتور ها
- و یه سری اطلاعات مانند حرکت چرخش هر سمت و جهت حرکت آن ها

و بقیه اعمال آن به صورت خودکار کنترل و محافظت می شود، مثل جایگاه سوخت، میزان فاصله از ماشین جلویی و کنترل دمای ماشین و اقدامات مربوطه همانند روشن کردن خودکار فن ماشین برای خنک کردن.

ورودی ها و خروجی های خواسته شده به شرح زیر می باشد :

ورودی ها:

- با استفاده از کیپد اطلاعاتی که میخوایم رو بدست میاریم
- تنظیم روشنایی با استفاده از کلید تعبیه شده
- افزایش و کاهش سرعت با استفاده از کلید های تعبیه شده
- میزان دما رو در شبیه سازی میتونیم برای حالات مختلف آزمایش و شبیه سازی کنیم.
- میزان فاصله از ماشین مقابل را هم می توانیم با استفاده از سنسور فاصله سنج شبیه سازی کنیم و عملکرد متقابل سیستم را در برابر این اتفاق نظاره کنیم.
- میزان سوخت را هم میتوانیم شبیه سازی کنیم.

خروجی ها:

- تنظیم دمای داخلی ماشین با استفاده از فن
- اعلام هشدار و در صورت لزوم خاموش کردن کل سیستم به هنگام قرار گرفتن در شرایط بحرانی همانند آتش سوزی
- کنترل مقدار فاصله (2 متر حد مجاز با ماشین جلویی) و توقف آنی برای جلوگیری از تصادف
- کنترل حرکت خودرو ها به صورت عقب، جلو، چپ، راست و ایست ناگهانی
- کنترل حرکت خودرو به صورت زوج چرخ (چرخ های سمت چپ و راست)
- بدست اوردن اطلاعات همانند سرعت چرخ های سمت چپ یا راست و جهت حرکت آن ها
- هشدار در خصوص کمبود سوخت
- تنظیم سرعت خودرو
- و روشن کردن دستی و خودکار چراغ های عقب، جلو و راهنما

1. کیپد keypad
2. buttons (3 عدد)
3. LCD LM044L
4. چراغ ها (12 عدد)
5. بوق هشدار Buzzer
6. ترانزیستور BC337
7. مقاومت 10 کیلو اهم (3 عدد)
8. موتور DC (4 عدد)
9. منبع ولتاژ (24 ولت)
10. درایور L298 (2 عدد)
11. ATMega 32
12. سنسور تشخیص دما LM35
13. سنسور التراسونیک SRF04
14. فن DC

ابتدا کتابخانه های مورد نیاز را تعریف میکنیم.

کتابخانه های تعریف شده به شرح زیر است:

Mega32.h , alcd.h , delay.h , stdio.h , stdlib.h که به ترتیب برای پردازندمون، ال سی دی، تاخیر و تبدیل عدد اعشاری به رشته و کاربردهای دیگه به کار می رود.

```
1   //for atmega32
2   #include <mega32.h>
3
4   // Alphanumeric LCD functions
5   #include <alcd.h>
6   #include <delay.h>
7   //for ftoa and ...
8   #include <stdio.h>
9   #include <stdlib.h>
```

برای کالیبره کردن دماسنج میایم از define استفاده می کنیم.

```
11  //for calibre kardane damasanj
12  #define ADC_VREF_TYPE 0x00
13  #define calibr_lm35 4.836
```

سپس متغیرهای گلوبال مان را تعریف می کنیم.

```
15  char x=3,temp;
16  int t,f=10000;
17  int timer_overflow=0,timer_overflow2=0;
```

پس از نوبت به تعریف کردن توابع میباشد که به ترتیب:

تابع کیپد: با شیفت دادن پورت های خروجی منتظر گرفتن دستور می باشد وقتی دکمه زده می شود دستور را شناسایی کرده و مشخصات i و j ان را می خواند و دستور مربوطه را اعمال می کند.

تابع کنترل موتور: این تابع برای کنترل کردن موتورها به صورت مستقل به کار برده می شود.

تابع جابجایی: این تابع برای کنترل جهت حرکت خودرو مورد بهره قرار می گیرد.

تابع اطلاعات: اطلاعات مربوطه از جمله سرعت موتورهای سمت چپ و راست، جهت حرکت و میزان سوخت را به ما اطلاع می دهد.

تابع دما: برای خواندن دما و اعمال تغییرات روی ورودی adc به کار می رود.

تابع التراسونیک: برای سنجیدن فاصله با اشیا یا خودروی مقابل و جلوگیری از بروز حوادث ما این تابع را تعریف کردیم.

تابع سوخت: میزان سوخت را به ما می گوید و در صورت کمبود سوخت به ما هشدار می دهد.

تابع adc_init : برای فعال کردن رجیستر های مربوط به adc به کار می رود.

تابع adc_read : این تابع برای خواندن ورودی adc از پین مربوطه به کار می رود.

```
18  //functions
19  int key_pad(void);
20  void Control_Motor(void);
21  void Direction_of_movement(void);
22  void Information(void);
23  void temperature(void);
24  void ultrosonic(void);
25  void fuel(void);
26  void ADC_Init();
27  int ADC_Read(char channel);
```

سپس اینتراپت های مربوطه را برای نمایش adc و فاصله اشیا از سنسور التراسونیک (که در جلوی ماشین قرار دارد) بر روی lcd نمایش داده می شود.

```
29  // timer 0 , 1 interrupt
30  interrupt[TIM0_OVF] void timer1_ovf_isr(void)
31  {
32  #asm("cli")
33  if(timer_overflow>500)
34  {
35      timer_overflow=0;
36      ultrosonic();
37      temperature();
38      timer_overflow=0;
39  }
40  timer_overflow++;
41  TCNT0=0;
42  #asm("sei")
43  }
44  interrupt[TIM2_OVF] void timer2_ovf_isr(void)
45  {
46
47      t++;
48      timer_overflow2++;
49      TCNT2=0;
50  }
```

ما اکسترنال های اینتراپت خروجی را برای کاهش و افزایش سرعت موتور ها به طور مستقل به کار می بریم.

```
// External Interrupt 0 service routine
interrupt [EXT_INT0] void ext_int0_isr(void)
{
    delay_ms(10);
    switch(x)
      {
      case 1 :
            if(OCR0<=(225)) {OCR0=OCR0+20;}
            break;
      case 2:
            if(OCR2<=(225)) {OCR2=OCR2+20;}
            break;
      case 3:
            if((OCR0<=(225))&&(OCR2<=(225))) {OCR0=OCR0+20; OCR2=OCR2+20;}
            break;
      }

}

// External Interrupt 1 service routine
interrupt [EXT_INT1] void ext_int1_isr(void)
{
    delay_ms(10);
    switch(x)
      {
      case 1 :
            if(OCR0>30) {OCR0=OCR0-20;}
            break;
      case 2:
            if(OCR2>30) {OCR2=OCR2-20;}
            break;
      case 3:
            if((OCR0>30)&&(OCR2>30)) {OCR0=OCR0-20; OCR2=OCR2-20;}
            break;
      }
```

در تابع main ابتدا پورت ها را مشخص می کنیم که ورودی هستند یا خروجی. سپس به کاربر اجازه می دهیم تا از بین موارد موجود اعمال یا اطلاعات مورد نیاز را بدست اورد.

سپس در ادامه ی کد توابعی وجود دارد که در قسمت بالا توضیح داده شده است.

<u>تابع کیپد</u> : یه حلقه بی نهایت میزنیم. سپس دکمه ها را یک به یک بررسی میکنیم. در صورت فشرده شده هر کدام از دکمه ها s ما دارای یه مقدار می شود و این مقدار به برنامه مان ارجاع داده می شود.

```
int key_pad(void)
{
    int S;
    //get number
    while(1)
    {
        PORTC=0xdf;
        if(!PINC.2){while(PINC.2==0);S= 3;break;}
        if(!PINC.1){while(PINC.1==0);S= 6;break;}
        if(!PINC.0){while(PINC.0==0);S= 9;break;}
        PORTC=0xbf;
        if(!PINC.3){while(PINC.3==0);S= 0;break;}
        if(!PINC.2){while(PINC.2==0);S= 2;break;}
        if(!PINC.1){while(PINC.1==0);S= 5;break;}
        if(!PINC.0){while(PINC.0==0);S= 8;break;}
        PORTC=0x7f;
        if(!PINC.2){while(PINC.2==0);S= 1;break;}
        if(!PINC.1){while(PINC.1==0);S=4;break;}
        if(!PINC.0){while(PINC.0==0);S=7;break;}
    }
    return S;
}
```

تابع دما : با استفاده از اینتراپت های 0 و 1مون، هر موقع این رجیستر اورفلو کرد. میاد تو تابع اینتراپت، وقتی که میره تو اینتراپت برا اینکه رفرش سریع انجام نشه یه if میذاریم ک بعد از 100بار اورفلو شدن بره داخل تابع دما. تو تابع دما با استفاده از تابع خوندن دما، adc خونده میشه و سپس در تابع دما تبدیل به مقدار دما می شود و برای اعمال محافظت، اگر دما بین 70 تا 80 باشد هشدار داده و سعی در خنک کردن ماشین دارد. اگر دما از 80 بیشتر شود سیستم ماشین به طور کل خاموش میشود و پیام خطر صادر می کند. برای دماهای کمتر از 70 هم که شرایط ایده آل و بهینه هست.

سپس پورت پنجم D را صفر می کنیم.

وقفه اینتراپ های تایمرهای 0 و 2 را فعال می کنیم.

```
202 void temperature(void)
203 {
204         char Temperature[6];
205         float celsius;
206         while(1)
207         {
208
209            //for temperature
210            celsius = ADC_Read(0)*calibr_lm35;
211            celsius /= 10;
212            ftoa(celsius,1,Temperature);
213            //temperature protection section
214            if(celsius>70&&celsius<80) {lcd_gotoxy(16,0); lcd_puts("!!!!");PORTD.5=1;}
215            else if(celsius>80) {lcd_clear(); lcd_puts("warning !!");delay_ms(100);OCR0=0;OCR2=0;}
216            else {lcd_gotoxy(16,0); lcd_puts(Temperature); break;}
217            PORTD.5=0;
218         }
219     TIMSK=0b01000001;    //fa'al kardane vaghfeie 0 , 2
220     #asm("cli")
221 }
```

تابع کنترل موتور: ابتدا گزینه های کنترلی را به مصرف کننده نشان می دهیم.

سپس با گرفتن دستور از کیپد، مقدار x را متناسب با ورودی کاربر مقداردهی می کنیم. سپس با متغیری که داریم مقدار سرعت موتور را در اینتراپ خروجی بدست میاریم.

```
223 void Control_Motor(void)
224 {
225     char j;
226     //lcd
227     lcd_clear();
228     lcd_gotoxy(0,0);
229     lcd_puts("1-Left ");
230     lcd_gotoxy(0,1);
231     lcd_puts("2-Right ");
232     lcd_gotoxy(0,2);
233     lcd_puts("3-Both ");
234     lcd_gotoxy(0,3);
235     lcd_puts("4-STOP");
236     //control section
237     j=key_pad();
238     if((j==1)) {x=j;}
239     else if(j==2) {x=j;}
240     else if(j==3) {x=j;}
241     else if(j==4) {PORTD.0=0;PORTD.1=0;OCR0=0;OCR2=0;}
242 }
```

تابع کنترل جهت حرکت : ابتدا گزینه ها برای کاربر نمایش داده می شود. سپس با گرفتن دستور مورد نظر از کیپد، بیت های مناسب را در ورودی درایور l298 قرار داده می شود.

اگه بیت بالا یک شود و پایینی صفر موتور به سمت جلو (راستگرد) حرکت میکند و اگر بیت ها را متمم کنیم جهت حرکت برعکس می شود.

```
void Direction_of_movement(void)
{   //lcd
    char j;
    PORTD.4=0;
    PORTA.6=0;
    PORTA.7=0;
    lcd_clear();
    lcd_gotoxy(0,0);
    lcd_puts("1-go ahead");
    lcd_gotoxy(0,1);
    lcd_puts("2-Return");
    lcd_gotoxy(0,2);
    lcd_puts("3-go right");
    lcd_gotoxy(0,3);
    lcd_puts("4-go left");
    //rastgard chapgard aghabgard va jologard
    j=key_pad();
    switch(j)
    {
        case(1):
                PORTD.0=1; PORTD.1=0; OCR0=0x80; OCR2=0x80; temp=1; break;
        case(2):
                PORTD.0=0; PORTD.1=1; OCR0=0x80; OCR2=0x80; temp=0; PORTD.4=1; break;
        case(3):
                PORTD.0=1-temp; PORTD.1=temp; OCR0=0xff; OCR2=0x80; PORTA.6=1; break;
        case(4):
                PORTD.0=1-temp; PORTD.1=temp; OCR0=0x80; OCR2=0xff; PORTA.7=1; break;
    }
}
```

تابع اطلاعات : سرعت جهت های چپ و راست را با استفاده از مقادیر سرعتی که به صورت گلوبال ذخیره کرده بودیم بدست میاریم و نمایش می دهیم.

در خصوص مقدار سوخت هم با توجه به مسافت پیموده شده (میانگین سرعت چرخ هایمان ضرب در مدت زمان حرکتمون) در یه ضریب مشخص را از ماکزیمم گنجایش سوخت که 10 هزار در نظر گرفتیم کم می کنیم. و مقدار سوخت مان بدست می آید. اگر به حد کمتر از هزار رسید هشدار می دهد.

```
276 void Information(void)
277 {   //lcd and information
278     char p[10];
279     lcd_clear();
280     lcd_gotoxy(0,0);
281     lcd_puts("LeftSpeed:");
282     lcd_gotoxy(12,0);
283     ftoa(OCR0,1,p);
284     lcd_puts(p);
285     lcd_gotoxy(0,1);
286     lcd_puts("RightSpeed:");
287     lcd_gotoxy(12,1);
288     ftoa(OCR2,1,p);
289     lcd_puts(p);
290     fuel();
291     lcd_gotoxy(0,2);
292     lcd_puts("Fuel:");
293     lcd_gotoxy(12,2);
294     ftoa(f,1,p);
295     lcd_puts(p);
296     lcd_gotoxy(0,3);
297     lcd_puts("1-Back to Menu");
298     key_pad();
299 }
```

تابع ADC_Init : adc را فعال می کند. (این عمل در کد ویزارد انجام شده)

```
298 //active adc
299 void ADC_Init()
300 {                                                   /* Make ADC port as input */
301     ADCSRA = 0x87;              /* Enable ADC, with freq/128  */
302     ADMUX = 0x40;               /* Vref: Avcc, ADC channel: 0 */
303 }
```

تابع ADC_Read : این هم adc را می خواند (این عمل نیز در کد ویزارد صورت گرفته)

```
304 //read adc
305 int ADC_Read(char channel)
306 {
307     ADMUX = 0x40 | (channel & 0x07);   /* set input channel to read */
308     ADCSRA |= (1<<ADSC);               /* Start ADC conversion */
309     while (!(ADCSRA & (1<<ADIF)));     /* Wait until end of conversion by polling ADC interrupt flag */
310     ADCSRA |= (1<<ADIF);               /* Clear interrupt flag */
311     delay_ms(1);
312
313     return ADCW;                       /* Return ADC word */
314 }
```

تابع التراسونیک : هر موقع که تابع دما فراخوانی می شود، تابع التراسونیک هم فراخوانی می شود. در این تابع یک پالس فرستاده شده و مدت زمان برگشت ان را محاسبه کرده و اگر کمتر از یک مقدار مشخص باشد موتورها را خاموش کرده و مقدار فاصله را در lcd نمایش می دهد.

رجیستر شمارنده تایمر 2 را صفر می کند و زمان رفت و برگشت سیگنال را محاسبه می کند و با استفاده از ان مسافت را محاسبه می کند.

```
316 void ultrosonic(void)
317 {
318       float mm=0;
319       int timer;
320       char cmm[10];
321       #asm("sei")
322       TIMSK=0b01000000;
323       PORTA.1=1;
324       delay_us(10);
325       PORTA.1=0;
326       while(PINA.2==0);
327       timer_overflow2=0;TCNT2=0;
328       while(PINA.2==1){if(timer_overflow2>5){mm=1;break;}}
329       timer=(timer_overflow2*256+TCNT2);
330       if(mm==0)
331       {PORTD.0=0;PORTD.1=0;OCR0=0;OCR2=0;
332       mm=timer*0.17;
333       ftoa(mm,1,cmm);
334       lcd_gotoxy(14,2);
335       lcd_puts(cmm);}
336 }
```

تابع سوخت : این تابع ابتدا مقدار سرعت میانگین دو طرف چرخ را بدست می اورد. سپس در یه ضریب و زمان مشخص ضرب کرده و عدد بدست امده را از ماکزیمم گنجایش مخزن ماشین کم می کند.

```
340 void fuel(void)
341 {
342       int avg;
343       avg=(OCR0+OCR2)/2;
344       f=f-(avg*0.0001*t);
345       t=0;
346 }
```