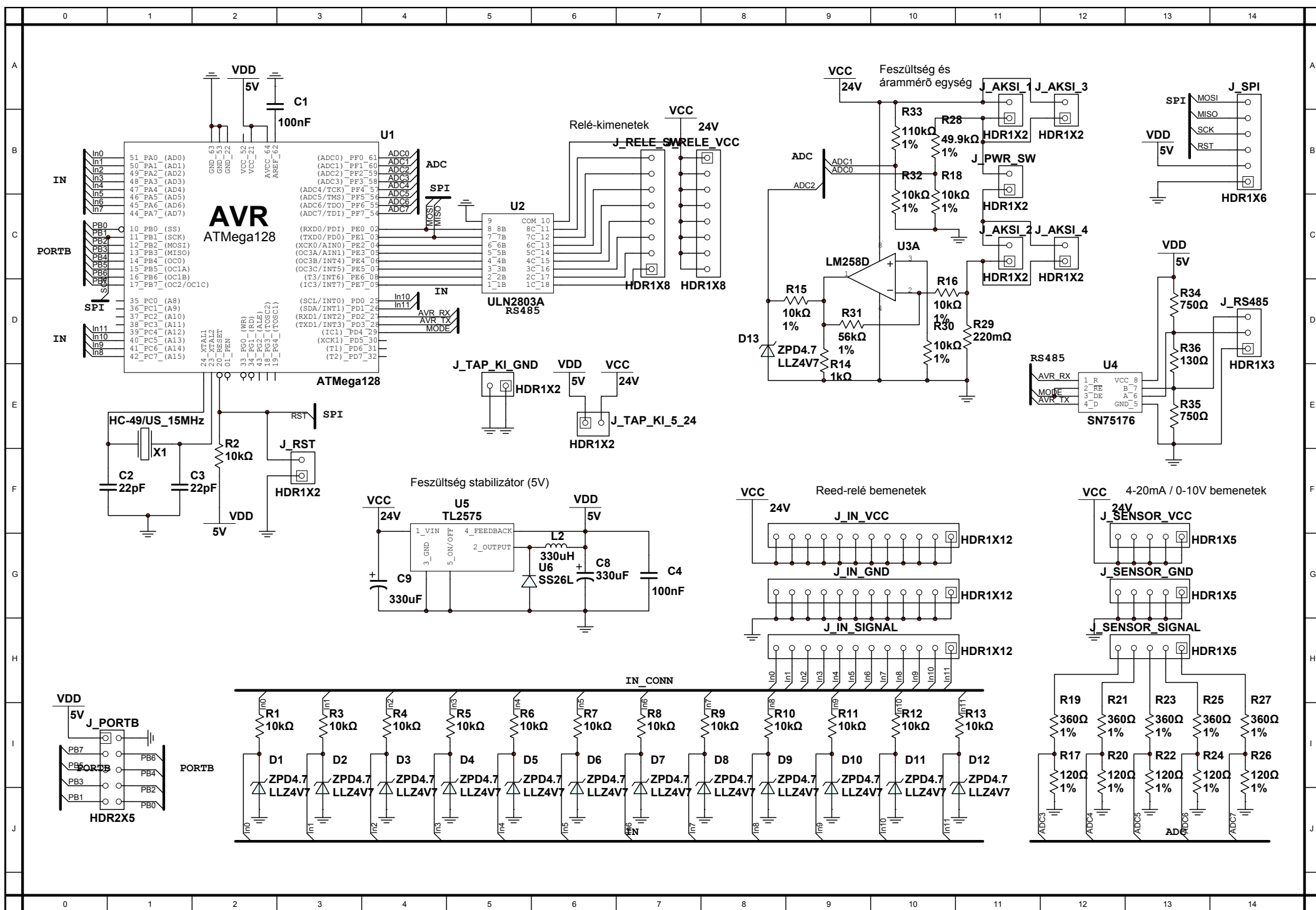
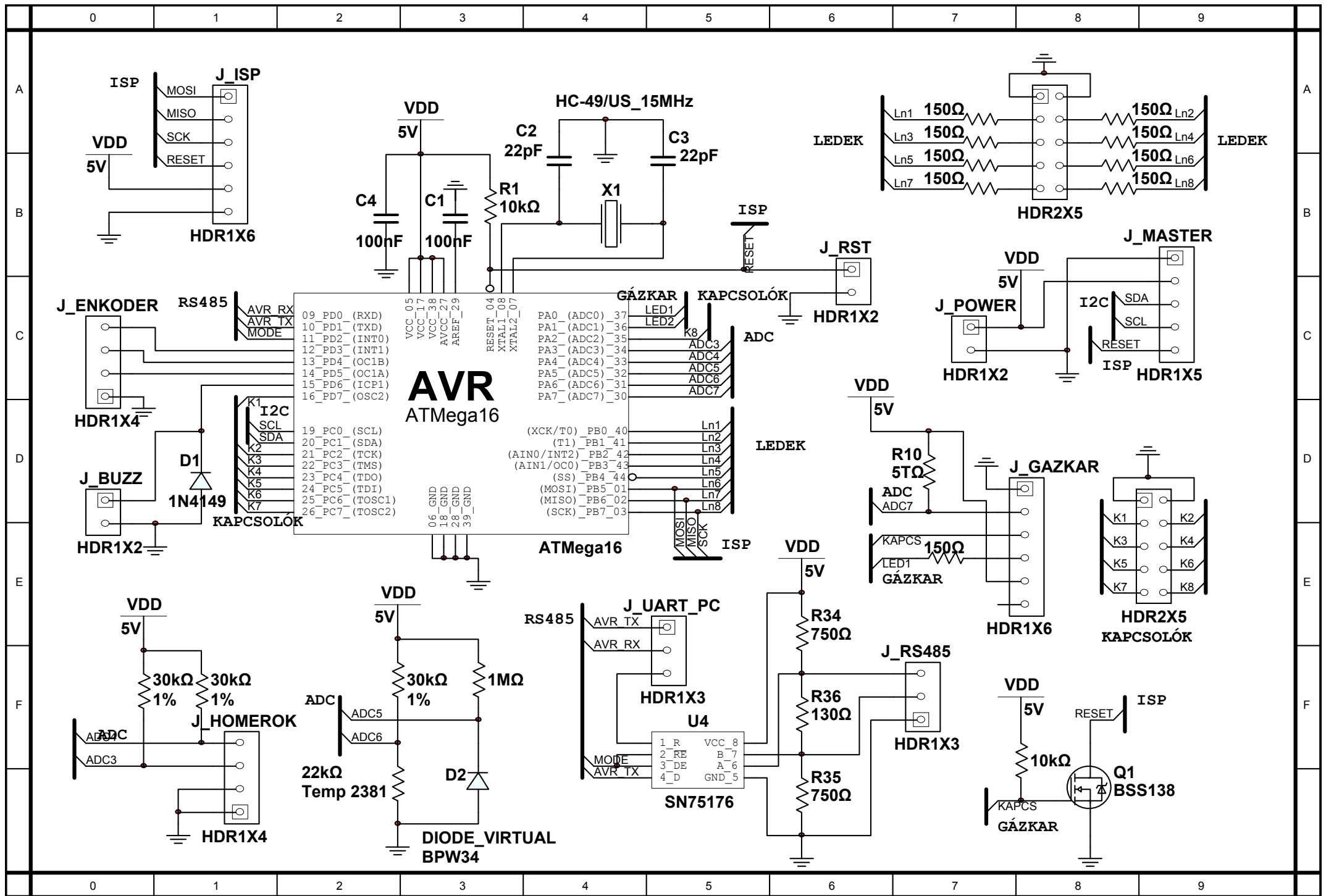






Változók listája és azonosítójuk a kommunikáció során:

ID	Változó	Típus	Mértékegység	Frissítés	Honnan
0	TimeSEC	unsigned int	s	1 Hz	Vezérlő
1	DugattyuPOZ	unsigned char	0,4%	10%	Vezérlő
2	DugattyuSEB	unsigned char	5 mm/s	4%	Vezérlő
3	DugattyuCiklus	unsigned int	db	dugattyu indítás	Vezérlő
4	KerekUt	unsigned int	m	jelre	Vezérlő
5	KerekSebesseg	unsigned int	0,1 km/h	jelre	Vezérlő
6	AkkuFesz1	unsigned int	mV	10 Hz	Vezérlő
7	AkkuFesz2	unsigned int	mV	10 Hz	Vezérlő
8	AkkuAram	unsigned int	mA	10 Hz	Vezérlő
9	AkkuAramUsed	unsigned int	0,01 mAh	5 sec	Vezérlő
10	Nyomas	unsigned int	0,01 Bar	10 Hz	Vezérlő
11	ExpandHatar	unsigned char	0,4%	dugattyu indítás	Vezérlő
12	PufferSzazalek	unsigned char	%	10 Hz	Vezérlő
13	ElhasznaltLevego	unsigned int	0,1 Nliter	dugattyu indítás	Vezérlő
14	Uzemmod	unsigned char		bekapcsolás	Vezérlő
15	VezeresiAllapot	unsigned char	8 bit	szelepváltás	Vezérlő
16	HibaKodok	unsigned int		1 Hz	Vezérlő
17	CPUMaxok	unsigned int		1 Hz	Vezérlő
	LOGFILE bejegyzés	array	-	10%, sebesség jel	Vezérlő
0	TempomatSebesseg	unsigned char	0,1 km/h		GUI
1	GazkarPozicio	unsigned char	8 bites ADC	25 Hz	Adatgyűjtő
2	Kapcsolobemenetek	unsigned char		folyamatosan	Adatgyűjtő
3	EnkoderTemp	signed char		?	Adatgyűjtő
4	EnkoderButton	bit		?	Adatgyűjtő
5	HomersekletBE	unsigned int	0,01°K	25 Hz	Adatgyűjtő
6	HomersekletKI	unsigned int	0,01°K	25 Hz	Adatgyűjtő
7	HomersekletDoboz	unsigned int	0,01°K	25 Hz	Adatgyűjtő
8	FenyeroDoboz	unsigned char	8 bites ADC	25 Hz	Adatgyűjtő
9	Buzzer()	bit			GUI

Hibakódok listája:

Ssz	Bit (HEX)	Hibaüzenet
1	0x0001	ADC érték nem állt időben rendelkezésre
2	0x0002	ADC nem fejezte be időben a méréseket!
3	0x0004	Kerék sebesség adat nem lett időben feldolgozva
4	0x0008	Túlcordult a naplófájl puffer
5	0x0010	Naplóbejegyzés sora túl hosszú
6	0x0020	Túl sok egyszerre elküldendő paraméter
7	0x0040	Ismeretlen adat küldése
8	0x0080	Szabad az USART mégis küldési hiba
9	0x0100	USART fogadási hiba
10	0x0200	USART fogadási timeout
11	0x0400	Analóg útdók váltása sikertelen
12	0x0800	ADC újabb pozíciót méri, de még nem lett feldolgozva az előző
13	0x1000	Útdó jele ugrásszerűen megváltozott!
14	0x2000	Sebességváltó beragadt, nem sikerült kiszedni az indítófokozatot
15	0x4000	Valószínűsített kerék mágnes kimaradás

```

/*****
|
|      MAIN
|
| *****/
void main(void){
    unsigned char i;
    Initialise();
    #asm("sei")

    LogFile[LogLinePointerRead][0]=0;
    Kapcsolobemenetek = ~( ((PINC&0xFC)>>1) | ((PINA.2)<<7) | PIND.7 );
    for (i=0;i<5;i++) ADC_Tomb[i]=0;
    for (i=0;i<IDCount;i++) FogadTomb[i]=0;
    FogadTomb[IDDugattyuPOZ]=0xFF;

    BuzzerBE();
    PORTB = 0xFF;
    delay_ms(100);
    BuzzerKI();
    PORTB = 0x00;

    while (1){
        Kapcsolobemenetek = ~( ((PINC&0xFC)>>1) | ((PINA.2)<<7) | PIND.7 );
        PORTB = FogadTomb[IDVezeresiAllapot];
        ADCCalculations();
        if (!DelayMS){
            //...barmi...
            DelayMS=1000;
        }
    }
}

/*****
|
|      ADC és feldolgozó
|
| *****/
#define ADC_VREF_TYPE 0x40
interrupt [ADC_INT] void ADC_InterruPT(void){
    unsigned char i;

    if(ADC_Scrap++){
        ADC_Tomb[ADC_Loop] += ADCW;
        if (++ADC_Loop >= 5){ ADC_Loop=0; ADC_AtlagDB++; }
        ADMUX = ADC_VREF_TYPE + ADC_Loop + 3;
        ADC_Scrap = 0;

        if(ADC_AtlagDB >= 40){
            for (i=0; i<5; i++){
                ADC_OsszegTomb[i] = ADC_Tomb[i];
                ADC_Tomb[i]=0;
            }
            ADC_Szamold = 1;
            ADC_AtlagDB = 0;
        }
    }
}

// Innen fogja tudni, hogy nincs küldendő adat a pufferben
// Kapcsolóbemenetek beolvasása (vezérlő egyből tudni szeretné)
// ADC átlagoló tömb nullázása
// Változók kezdeti értéke 0
// Dugattyú pozíciójá 255 = érvénytelen

// Bekapcsolás/reset esetén LED-ek felvillantása, hangjelzés adása

```

```

unsigned char Feny (unsigned int Feny40value) {
    Feny40value /= 40;
    Feny40value = 1023-Feny40value;
    Feny40value *= 49;
    Feny40value /= 110;
    Feny40value >>=1;
    return Feny40value;
}

unsigned int Homero (unsigned int ADC40value) {
    float Arany;
    Arany=(float)ADC40value/(40*1024-ADC40value);
    Arany*=1.363636;
    Arany=log(Arany);
    Arany*=2.744032e-04;
    Arany+=3.354016e-03;
    Arany=100/Arany;
    return Arany;
}

unsigned char Gazkar (unsigned int Gazkar40value) {
    Gazkar40value /= 40;
    if (Gazkar40value>840) Gazkar40value = 840-174;
    else if (Gazkar40value>174) Gazkar40value -= 174;
    else Gazkar40value=0;
    Gazkar40value *= 51;
    Gazkar40value /= 133;
    return Gazkar40value;
}

void ADCCalculations(void) {
    if (ADC_Szamold) {
        HomersekletBE = Homero(ADC_OsszegTomb[ADHomOsszegBE]);
        HomersekletKI = Homero(ADC_OsszegTomb[ADHomOsszegKI]);
        HomersekletDoboz = Homero(ADC_OsszegTomb[ADHomOsszegDoboz]);
        FenyeroDoboz = Feny(ADC_OsszegTomb[ADFenyOsszeg]);
        GazkarPozicio = Gazkar(ADC_OsszegTomb[ADGazkarPozOsszeg]);
        ADC_Szamold = 0;
    }
}

/*****
|
| Buzzer
|
*****/

const unsigned char UtemHossz=40;
unsigned char HangLoop;
unsigned char HangUtem;
unsigned int HangSav;

interrupt [TIM0_COMP] void Timer0_1kHz(void) {
    #asm("sei")
    #asm("wdr")
    BuzzerKimenet();
    if (DelayMS) DelayMS--;
}

```

```

// 40 összeadott érték átlaga
// Nem a diódát, hanem a pozitív oldali ellenállást mérjük
// Ellenállás feszültsége 0,1mV-ban
// I=U/R | R=100K -> [0,1uA] | R=1K -> [10uA]
// [uA] -> [lux]
// [lux] Fényerősség lux-ban

```

```

// Arany=R/Rin (40 összeadott érték átlaga)
// 30k az R1, 22k az NTC 25°C-os ellenállása (1.363636 = 30/22)

// Adatlappból vett érték. 100k-s NTC esetén: 2.460382e-04;
// Adatlappból vett érték. 100k-s esetén is ennyi
// Reciprok és szorzás 100-al
// [0,01K] Érték visszaadása század kelvinben

```

```

// 40 összeadott érték átlaga
// Gázkar itt éri el a maximumát (4,25V)
// Gázkar minimuma (0,85V)
// Alatta 0-nak vesszük
// Átkonvertálás 0-255-ös skálára:
// *255/(840-174) = *51/133
// Gázkar pozíció 0-255

```

```

// ADC értékek feldolgozása, átváltása
// Van 40 új mérés, számolni kell

```

```

// Egy ütem hossza [ms]-ben

```

```

void HangUzenet(unsigned int HangSavl){
    HangUtem=0;
    HangLoop=UtemHossz-1;
    HangSav=HangSavl;
}

void BuzzerKimenet(){
    if (HangSav){
        if ((++HangLoop)==UtemHossz){ HangUtem++; HangLoop=0; }
        if (HangUtem<=16){
            if ((HangUtem)&&(HangSav&(1<<(16-HangUtem)))) BuzzerBE();
            else BuzzerKI();
        }else{
            BuzzerKI();
            HangSav=0;
        }
    }
}

/*****
|                               TWI (I2C)
| *****/

unsigned char TWIPProgress;
unsigned int TWIReturnValue;
unsigned char TWIUtasitas;

void TWIError(){
    ERROR(6);
    Hibakodok &= 0x00FF;
    Hibakodok |= (unsigned int)TWSR<<8;
    TWIPProgress=0;
}

unsigned char LogByteToSend;
unsigned char ILogToSend(){ // Megvizsgálja van-e logfájl bejegyzés a pufforban, és bekészíti a küldendő bájtot LogByteToSend változóba
    // [0] Épp nincs küldendő bejegyzés [1] Van küldendő, de ez a sor utolsó eleme [2] Van küldendő, nem utolsó [3] Van küldendő, új sor kezdete
    unsigned char LogLinePointerReadNext;
    if (LogCharPointerRead<LogFile[LogLinePointerRead][0]){
        LogByteToSend=LogFile[LogLinePointerRead][LogCharPointerRead++];
        if (LogCharPointerRead==LogFile[LogLinePointerRead][0]) return 1;
    }else return 2;
    if (LogLinePointerReadNext==LogLinePointerRead+1;
        if (LogLinePointerReadNext>=LogLineCount) LogLinePointerReadNext=0;
        if (LogLinePointerReadNext==LogLinePointerWrite){
            return 0;
        }else{
            LogLinePointerRead=LogLinePointerReadNext;
            LogByteToSend=LogFile[LogLinePointerRead][0];
            LogCharPointerRead=1;
            return 3;
        }
    }
}

```

```

void TWIReturn() {
    if (TWIUtasitas<IDCount) TWIReturnValue=FogadTomb[TWIUtasitas];
    else switch (TWIUtasitas-IDCount) {
        case IDTempomatSebesseg: TWIReturnValue=TempomatSebesseg; break;
        case IDGazkarPozicio: TWIReturnValue=GazkarPozicio; break;
        case IDKapcsolobemenetek: TWIReturnValue=Kapcsolobemenetek; break;
        case IDEnkoderTemp: TWIReturnValue=EnkoderTemp; break;
        case IDEnkoderButton: TWIReturnValue=EnkoderButton; break;
        case IDHomersekletBE: TWIReturnValue=HomersekletBE; break;
        case IDHomersekletKI: TWIReturnValue=HomersekletKI; break;
        case IDHomersekletDoboz: TWIReturnValue=HomersekletDoboz; break;
        case IDFenyeroDoboz: TWIReturnValue=FenyeroDoboz; break;
        case IDHibakodok2: TWIReturnValue=Hibakodok; break;
        default: TWIReturnValue=0x0000; break;
    }
}

void TWIAdatErkezett() {
    switch ((TWIUtasitas&0x3F)-IDCount) {
        case IDTempomatSebesseg: TempomatSebesseg=TWIReturnValue; break;
        case IDBuzzer: HangUzenet(TWIReturnValue); break;
    }
}

interrupt [TWI] void TWI_Interrupt(void) {
    if ((TWSR&0xF8)==0x60) TWIProgress=10;
    else {
        switch(TWIProgress) {
            // Utasítás értelmezése
            case 11: if ((TWSR&0xF8)!=0x80) TWIError();
                    else {
                        TWIUtasitas=TWDR;
                        if (TWIUtasitas&0x80) TWIProgress=20;
                        else if (TWIUtasitas&0x40) TWIProgress=30;
                        else TWIReturn();
                    }
                    break;
            // Változó küldése
            case 12: if ((TWSR&0xF8)!=0xA0) TWIError();
                    break;
            case 13: if ((TWSR&0xF8)!=0xA8) TWIError();
                    else TWDR=(TWIReturnValue>>8);
                    break;
            case 14: if ((TWSR&0xF8)!=0xB8) TWIError();
                    else TWDR=TWIReturnValue;
                    break;
            case 15: if ((TWSR&0xF8)!=0xC0) TWIError();
                    break;
            // Változó fogadása
            case 21: if ((TWSR&0xF8)!=0x80) TWIError();
                    else TWIReturnValue=(unsigned int)TWDR<<8;
                    break;
            case 22: if ((TWSR&0xF8)!=0x80) TWIError();
                    else TWIReturnValue+=TWDR;
                    break;
        }
    }
}

// Változó kiolvasás ID alapján
// Vezérlőtől (RS485) származó változók lekérdezése
// Saját adatok lekérdezése

// Kapott integer eltárolása ID alapján

// Own SLA+W has been received, ACK has been returned

// Previously addressed with own SLA+W, data received, ACK returned

// Utasítás kiolvasása TWI regiszterből
// Ha írni akar, ugrás a fogadás részhez
// Ha adatot vár (logfile-t), ugrás ahhoz a részhez
// Ha adatot vár (változót), előkészítjük az értéket küldésre

// A STOP or repeated START condition received while still addressed as slave

// Own SLA+R has been received, ACK has been returned
// Integer első felének küldése (felső bájt)

// Data byte in TWDR has been transmitted, ACK has been received
// Integer második felének küldése (alsó bájt)

// Data byte in TWDR has been transmitted, NOT ACK has been received

// Previously addressed with own SLA+W, data received, ACK returned
// Adat első felét megkaptuk, tárolás

// Previously addressed with own SLA+W, data received, ACK returned
// Adat második felét is megkaptuk, egyesítés és tárolás

```

```

case 23: if ((TWSR&0xF8)!=0xA0) TWIError();
        else TWIAdatErkezett();
        break;

// Logfájl küldése
case 31: if ((TWSR&0xF8)!=0xA0) TWIError();
        break;
case 32: if ((TWSR&0xF8)!=0xA8) TWIError();
        else{
            if (IsLogToSend()==3) TWDR=LogByteToSend;
            else { TWDR=0; TWIProgress++; }
        }break;
case 33: if ((TWSR&0xF8)!=0xB8) TWIError();
        else{
            if (IsLogToSend()>1) TWIProgress--;
            else TWIProgress++;
            TWDR=LogByteToSend;
        }break;
case 34: if ((TWSR&0xF8)!=0xB8) TWIError();
        else TWDR=0xFF;
        break;
case 35: if ((TWSR&0xF8)!=0xC0) TWIError();
        break;

// Egyéb esetben hiba
default: TWIError();
        break;
    }
}

TWCR=0xC5;
TWIProgress++;
}

/*****
|
| RS 485
|
*****/

void Transmit(void){
    delay_us(50); // Megvárjuk míg a vezérlő átadja a csatornát. 10 us-el már megy, de a késések miatt szkópon az csak csak 1-2 us (necces)
    // Azért kell, mert különben az UART várja a hosszú szünet "nullájának" stopbitjét (ami akár milyen hosszú is lehet)
    if (UCSRA & (1<<UDRE)){
        PORTD.2 = 1;
        switch(ReceiveID){
            case IDGazkarPozicio : UDR = GazkarPozicio; break;
            case IDKapcsolobemenetek : UDR = Kapcsolobemenetek; break;
            case IDTempomatSebesseg : UDR = TempomatSebesseg; break;
            default : UDR = 0x00; break;
        }
    }else ERROR(7);
}

interrupt [USART_TXC] void usart_tx_isr(void){
    PORTD.2 = 0;
    Process=0;
}

```

```

interrupt [USART_RXC] void usart_rx_isr(void) {
    unsigned char Status;
    if ((Status == UCSRA) & RX_COMPLETE){
        if ((Status & (FRAMING_ERROR | PARITY_ERROR | DATA_OVERRUN))==0){
            ReceivedByte = UDR;
            if(ReceivedByte==0xFF) Szinkron++;
            else
                Szinkron=0;
            if(Szinkron==6){
                if (!SzinkronOK) ERROR(1);
                Szinkron=SzinkronOK=0;
                Process=0;
            }else if (!SzinkronOK){
                if(!LogFogadas){
                    switch(Process){
                        case 0: if (ReceivedByte & 0x80){
                                    if (ReceivedByte & 0x40){
                                        if (ReceivedByte==0xFF) SzinkronOK=1;
                                    }else{
                                        ReceiveID=ReceivedByte&0x3F;
                                        Transmit();
                                    }
                                }else
                                    }else ReceiveID=ReceivedByte&0x3F;
                                Process++;
                            }else{
                                LogFogadas=1;
                                LogCharPointerWrite=0;
                                ReceivedTemp=ReceivedByte&0x3F;
                                if (LogLinePointerWrite==LogLinePointerRead){
                                    LogWroteDisabled=1;
                                    ERROR(2);
                                }else LogWroteDisabled=0;
                                }break;
                            // Csak változó fogadás esetén:
                        case 1: ReceivedTemp=ReceivedByte;
                                Process++;
                                }break;
                        case 2: if (ReceiveID<IDCount)
                                    FogadTomb[ReceiveID]=(unsigned int)(ReceivedTemp<<8)+ReceivedByte; // Változó mentése
                                else ERROR(5);
                                Process=0;
                                }break;
                    }
                }
            }
        if (LogFogadas){
            if (!LogWroteDisabled){
                if (LogCharPointerWrite>=LogCharSize) ERROR(3);
                else LogFile[LogLinePointerWrite][LogCharPointerWrite]=ReceivedByte; // Mentés a pufferbe
            }
            if (++LogCharPointerWrite==ReceivedTemp){
                if (!LogWroteDisabled)
                    if (++LogLinePointerWrite>=LogLineCount) LogLinePointerWrite=0; // Pointer léptetése
                LogFogadas=0;
            }
        }
    }else ERROR(4);
}

```