Redőnyvezérlés
ARDUINO-val

Szoftver tulajdonságai:

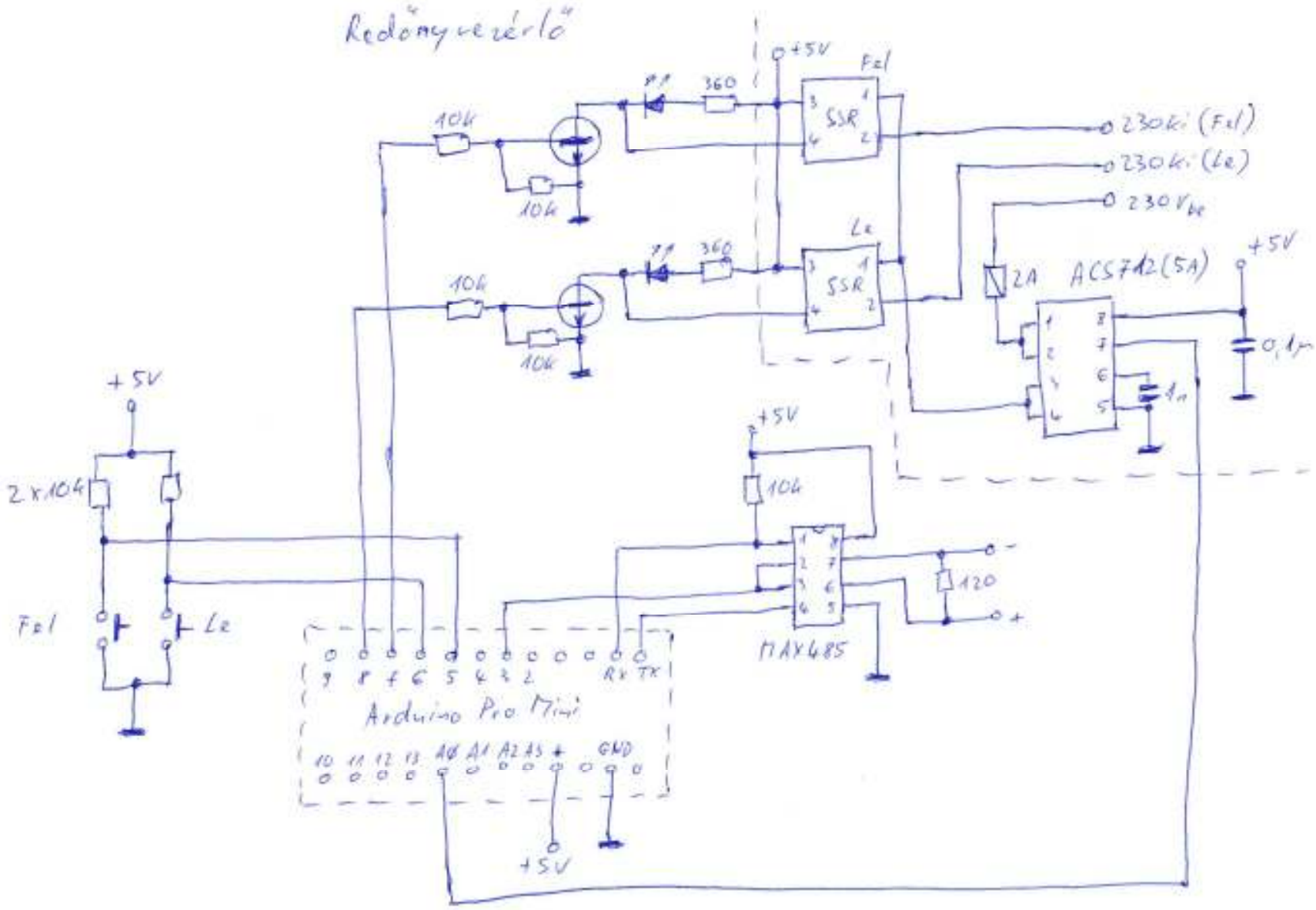
- Bármelyik gombot megnyomjuk, a redőny az adott irányba elindul, és a végállásig halad.
- Ha közben bármelyik gombot megnyomjuk, a redőny megáll.
- Ismételt gombnyomásra ismét elindul az adott irányba, a végállásig.
- A program elektronikusan is biztosítja a két gomb reteszelését.
- A felfelé gomb 3 mp folyamatos nyomva tartása után a redőny a gomb elengedéséig megy. Ezt az időt megjegyzi a program, és ez lesz az új felhúzási idő (pozíció).
- A programozás és programozott felhúzás csak az alsó pozícióból működik, teljesen leengedett állapotban.
- Programozott felhúzás után az ismételt gombnyomásra (felfelé) a végállásig halad a redőny.
- A redőny időben (msec) mért pontos pozíciója segítségével a végállások elektronikusan is be vannak állítva. A redőnyhöz beépített csőmotor mechanikus végállás kapcsolóval is rendelkezik.
- A millis() értékének átfordulása esetén a program újraindul (kb.50 nap).
- A programozott érték és a tényleges pozíció eltárolásra kerül az EEPROM-ban, így a program indításakor, valamint RESET esetén a redőny beállítása az eltárolt pozícióba történik.

Áramköri kialakítás:

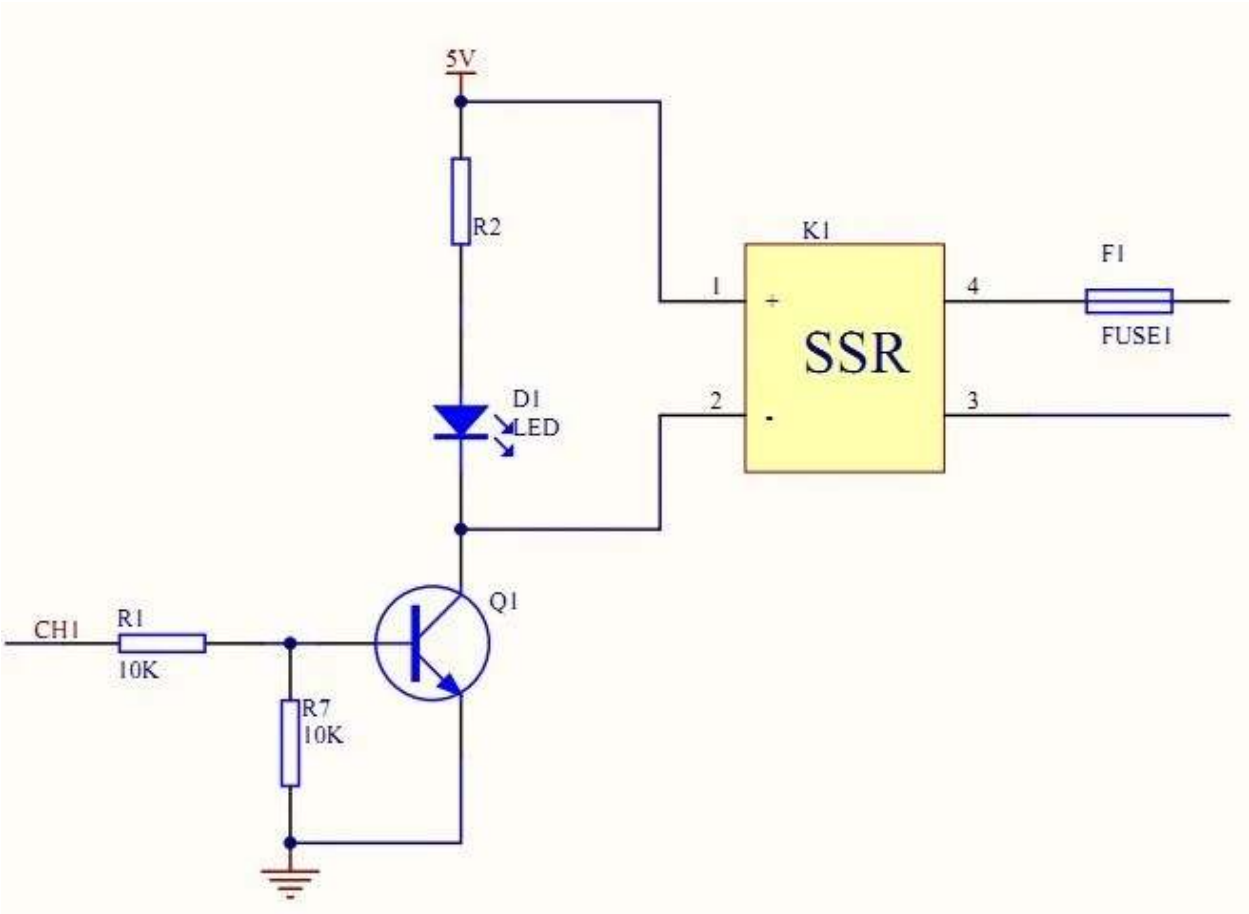
- A mikrovezérlő típusa Arduino Pro Mini
- A 230V kapcsolása SSR relékkel történik.
- Az aktív működést LED-ek jelzik.
- A későbbi fejlesztéshez áramérzékelő (ACS712, 5A-es) kerül beépítésre.
- Központi vezérlés kialakításához MAX-485-ös IC is beépítésre kerül.
- A panelek a szerelvénydobozba történő beépítéshez kerülnek kialakításra.
- Két panelen történik az alkatrészek elhelyezése. Külön a gyengeáramú, külön az erősáramú (230V) rész.

Jelenlegi program sajátosságai:

- Több olyan részt tartalmaz, melyek csak a fejlesztést és a tesztelést segítik.
- Az LCD kijelzővel kapcsolatos utasítások is ide tartoznak.
- A 13-as digitális pin használata (beépített LED) az EEPROM írás helyes kialakításához szükséges.
- A kész változatból ezeket ki kell törölni.
- A redőny teljes futási idejét méréssel kell meghatározni, és átírni a változót, illetve ahol szükséges (ahol 10000 szerepel). A 10 másodperces idő csak a gyorsabb tesztelés miatt lett beírva.

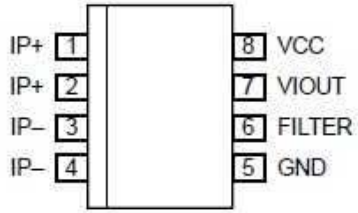

Redőnyvezérlő
+5V
Fel
360
10k
SSR
230Ki (Fel)
230Ki (Le)
230V ke
Le
2A
ACS712(5A)
0,1µ
1n
2x10k
Fel
Le
MAX485
120
9 8 7 6 5 4 3 2 RX TX
Arduino Pro Mini
10 11 12 13 A0 A1 A2 A3 A4 GND

SSR relé

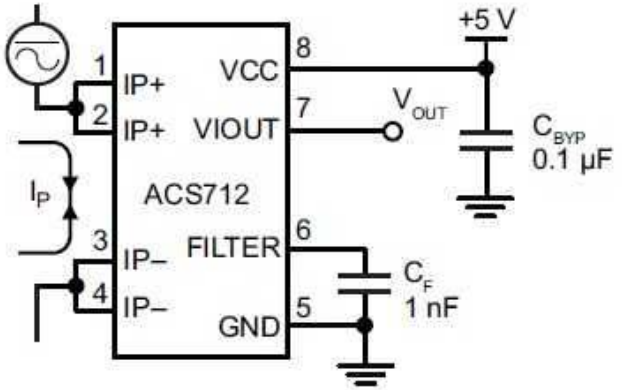


ACS712 áramérzékelő

Pin-out Diagram



Typical Application



Terminal List Table

Number	Name	Description
1 and 2	IP+	Terminals for current being sampled; fused internally
3 and 4	IP–	Terminals for current being sampled; fused internally
5	GND	Signal ground terminal
6	FILTER	Terminal for external capacitor that sets bandwidth
7	VIOUT	Analog output signal
8	VCC	Device power supply terminal

A működtető szoftver ismertetése

Program indítása, vagy reset esetén a korábbi pozíció beállítása:

void setup()

```
//Program indításakor a redőny teljesen leengedésre kerül.
digitalWrite(8, HIGH);
delay(felhuzasi_ido); //a millis() fügvény átfordulása miatt is szükséges
digitalWrite(8, LOW);
eepromread();          //eltárolt értékek beolvasása
if (pozicio!=0){
  digitalWrite(7, HIGH);
  delay(pozicio);
  digitalWrite(7, LOW);
```

Főprogram:

```
void loop() {
  gombok();                //gombnyomás figyelés
  futasido_figyeles();     //program módba belépéshez(3 sec után)
  futasi_ido_szamitas();   //pozíció számítás
  relay();
  kijelzes();
  szamlalo_atfordulas();   //millis() számláló átfordulásának figyelése
}
```

EEPROM:

```
void eepromread(){    //eltárolt értékek kiolvasása EEPROM-ból
  if (EEPROM.read(0) == 200){
    felhuzasi_ido = EEPROM.read(1)*1000+EEPROM.read(2)*100+EEPROM.read(3)*10+EEPROM.read(4);
    pozicio = EEPROM.read(5)*1000+EEPROM.read(6)*100+EEPROM.read(7)*10+EEPROM.read(8);
    }
}
//**********************
void eepromwrite_felhuzasi_ido(){     //programozott felhúzás eltárolása
  EEPROM.write(1,felhuzasi_ido/1000);
  EEPROM.write(2,felhuzasi_ido%1000/100);
  EEPROM.write(3,felhuzasi_ido%1000%100/10);
  EEPROM.write(4,felhuzasi_ido%1000%100%10);
  EEPROM.write(0, 200);     //ellenörzéshez
}
```

Nyomógombok (kapcsolók) kezelése:

```
void gombok(){
  ora_gombok1 = millis();                 //20 msec időközönként ellenörzi a nyomógomb állapotát
  if (ora_gombok1 - ora_gombok2 > 20){  //és engedélyezi a program végrehajtását
    ora_gombok2 = ora_gombok1;
    //program
    fel = digitalRead(5);
    if (fel==HIGH) fel_1=1;     //Ez a rész csak egyszer engedi a
    if (fel==LOW & fel_1==1){  //következő utasitásokat végrehajtani,
       fel_1=0;                 //függetlenül a nyomva tartás idejétől.
       //program
       if ((pozicio<10000)||(digitalRead(8)==HIGH)) felfele=true;
          //a pozicio figyelése szükséges, hogy a felső végállás után rövid időre se
          //engedje (20 msec) az újbóli felfelé kapcsolást. A kimenet figyelése lehetővé teszi,
          //hogy felső végállásból a lefelé tekerést a felfelé gomb is le tudja állitani.
       }
    le = digitalRead(6);
```

Futási idő számítása a pozíció meghatározásához:

```
void futasi_ido_szamitas(){    //pozíció meghatározása
  //felfele
  if ((digitalRead(7)==HIGH)&&(start_fel==0)){    //stopper start
    start_fel=millis();
    } else if((digitalRead(7)==LOW)&&(start_fel!=0)){ //stopper stop
        vege_fel=millis();
        eltelt_ido_fel=vege_fel-start_fel;    //msec
        pozicio=pozicio+eltelt_ido_fel;
        if (pozicio>10000) pozicio=10000;   //túlcsordulás miatt(fent)
        start_fel=0;
        eeprommwrite_pozicio();
        digitalWrite(13,HIGH);
        delay(5);
        digitalWrite(13,LOW);
        lcd.clear();
        }
  //lefele ***********************
```

Kijelzés:

```
void kijelzes(){
  lcd.setCursor (0,0);
  lcd.print("Fel:");
  lcd.print(eltelt_ido_fel/1000.0,1);
  lcd.setCursor (9,0);
  lcd.print("Le:");
  lcd.print(eltelt_ido_le/1000.0,1);
  lcd.setCursor (0,1);
  lcd.print("Pozicio:");
  lcd.print(pozicio/1000.0,1);
  lcd.print(" sec");
}
```

A belső számláló (millis) átfordulásának figyelése:

```
void szamlalo_atfordulas(){  //számláló átfordulásának figyelése - millis()
  if(millis()<3000) ujrainditas();    //reset
}
```

Szoftveres reset:

```
long pozicio=0;                    //a redőny pontos helyz
unsigned int felhuzas_idozitese=0;//tényleges felfelé tek
void (* ujrainditas) (void)=0;     //szoftveres reset

LiquidCrystal_I2C lcd(0x27,2,1,0,4,5,6,7,3,POSITIVE);
//*****************************************************
void setup() {
  pinMode(5, INPUT);
  pinMode(6, INPUT);
```

További fejlesztések:

- A belső számláló (millis függvény) használatának módosítása az átfordulási probléma kiküszöbölésére (szoftveres reset elhagyása).
- Az áramérzékelő segítségével a futási idő automatikus meghatározása.
- Túlterhelés védelem szoftveres kialakítása.
- Központi vezérléssel (RS485) napkelte – napnyugta szerinti működtetés.
- Egyéb, időjárás függő vezérlések kialakítása (viharos szél, alacsony külső hőmérséklet).

Napkelte – napnyugta szoftveres meghatározása

A program az adott hely koordinátája és a dátum alapján határozza meg a napkelte – napnyugta idejét, valamint a Nap delelését.

```
#include <Dusk2Dawn.h>
//*****************************************************************
void setup() {
  Serial.begin (9600);

  //Koordináta és időzóna megadása(GMT+1 óra, téli időszámítás)
  Dusk2Dawn Debrecen(47.528906, 21.625382, 1);
  //Dátum:év,hónap,nap
  int DebrecenSunrise  = Debrecen.sunrise(2024, 2, 22, false);
  int DebrecenSunset   = Debrecen.sunset(2024, 2, 22, false);
  Serial.println("Datum: 2024.02.22. ");
  Serial.println();

  char time[6];
  Dusk2Dawn::min2str(time, DebrecenSunrise);
  Serial.print("Debrecen napkelte: ");
  Serial.println(time);
```

```
  char time2[] = "00:00";
  Dusk2Dawn::min2str(time2, DebrecenSunset);
  Serial.print("Debrecen napnyugta: ");
  Serial.println(time2);

  //Delelése idő meghatározása
  int DebrecenSolarNoon = DebrecenSunrise + (DebrecenSunset - DebrecenSunrise) / 2;
  char time3[] = "00:00";
  Dusk2Dawn::min2str(time3, DebrecenSolarNoon);
  Serial.print("A Nap delel: ");
  Serial.println(time3);
}
//**************************************************************
void loop() {
}
//**************************************************************
```