```
#include <stdio.h>
#include <regc515c.h>
#include <stub.h>
#include <math.h>

//welcher DIP-Schalter den Modus (Hase/Fuchs) regelt
#define DIP_PIN_HASENMODE 0
//welcher DIP-Schalter die Max-Geschwindigkeit regelt
#define DIP_PIN_MAXSPEED 3
// definiert welcher DIPPIN die Rundenzeit einstellt
#define DIP_PIN_ROUNDTIME 1
//Countdown bis zum Start
#define STARTDELAY 30000 // Default == 30000 (30Sek.)
//Digitaleingang für den Startknopf
#define STARTTRIGGEREINGANG 0
//Rundenzeit bis Poink sich abschaltet
#define ROUNDTIME2MIN 120000 //120000 == 2Min.
#define ROUNDTIME1MIN 60000 //60000 == 1Min.

long globaleVariableEndzeit;

int IR_PROC_PERIODS_FOR_OK;
int IR_PROC_HALF_PERIOD;
int IR_PROC_MAX_ERROR_PER_PERIOD;

//Statusvariablen für IR Empfänger

int WinkelNeu;
int IR0 = 0;
int IR1 = 0;
int IR2 = 0;
int IR3 = 0;

//Koordinaten des berechneten WandVektors
int WandX;
int WandY;
long IgnoreWand;

//Koordinaten des berechneten GengnerVektors
float GegnerX;
float GegnerY;


//Variablen MyServo
int w_step = 3;
int winkel;
int WinkelGegner;
char richtung = 0;  //Drehrichtung Servo, 0:rechts, 1;links

void FindeWand(void)
{
    #define DELTAIR 0

    //was der entsprechende Sensor mit deaktivierter IR-LED empfängt
    int Umgebungswert;

    //while(1){
        //von allen IR-Sensoren Differenzwert ermitteln
        WandX = WandY = 0;

        Umgebungswert = analog(0);                  //Umgebungshelligkeit messen
        led(0,1);   //die ersten 4 led's ansteuern
        led(1,1);   //die letzten 4 led's ansteuern
        //Helligkeit mit eingeschaltetem IR-Sender messen und Differenz errechnen
        WandX += -1 * (Umgebungswert - analog(0)) * 0.7071067812;
        WandY += 1 * (Umgebungswert - analog(0)) * 0.7071067812;
        led(0,0);
        led(1,0);

        Umgebungswert = analog(1);                  //Umgebungshelligkeit messen
        led(0,1);   //die ersten 4 led's ansteuern
        led(1,1);   //die letzten 4 led's ansteuern
        //Helligkeit mit eingeschaltetem IR-Sender messen und Differenz errechnen
        WandX += 0 * (Umgebungswert - analog(1));
        WandY += 1 * (Umgebungswert - analog(1));
```

```
        led(0,0);
        led(1,0);

        Umgebungswert = analog(2);                  //Umgebungshelligkeit messen
        led(0,1);   //die ersten 4 led's ansteuern
        led(1,1);   //die letzten 4 led's ansteuern
        //Helligkeit mit eingeschaltetem IR-Sender messen und Differenz errechnen
        WandX += 1 * (Umgebungswert - analog(2)) * 0.7071067812;
        WandY += 1 * (Umgebungswert - analog(2)) * 0.7071067812;
        led(0,0);
        led(1,0);

        Umgebungswert = analog(3);                  //Umgebungshelligkeit messen
        led(0,1);   //die ersten 4 led's ansteuern
        led(1,1);   //die letzten 4 led's ansteuern
        //Helligkeit mit eingeschaltetem IR-Sender messen und Differenz errechnen
        WandX += 1 * (Umgebungswert - analog(3));
        WandY += 0 * (Umgebungswert - analog(3));
        led(0,0);
        led(1,0);

        Umgebungswert = analog(4);                  //Umgebungshelligkeit messen
        led(0,1);   //die ersten 4 led's ansteuern
        led(1,1);   //die letzten 4 led's ansteuern
        //Helligkeit mit eingeschaltetem IR-Sender messen und Differenz errechnen
        WandX += 1 * (Umgebungswert - analog(4)) * 0.7071067812;
        WandY += -1 * (Umgebungswert - analog(4)) * 0.7071067812;
        led(0,0);
        led(1,0);

        Umgebungswert = analog(5);                  //Umgebungshelligkeit messen
        led(0,1);   //die ersten 4 led's ansteuern
        led(1,1);   //die letzten 4 led's ansteuern
        //Helligkeit mit eingeschaltetem IR-Sender messen und Differenz errechnen
        WandX += 0 * (Umgebungswert - analog(5));
        WandY += -1 * (Umgebungswert - analog(5));
        led(0,0);
        led(1,0);

        Umgebungswert = analog(6);                  //Umgebungshelligkeit messen
        led(0,1);   //die ersten 4 led's ansteuern
        led(1,1);   //die letzten 4 led's ansteuern
        //Helligkeit mit eingeschaltetem IR-Sender messen und Differenz errechnen
        WandX += -1 * (Umgebungswert - analog(6)) * 0.7071067812;
        WandY += -1 * (Umgebungswert - analog(6)) * 0.7071067812;
        led(0,0);
        led(1,0);

        Umgebungswert = analog(8);                  //Umgebungshelligkeit messen
        led(0,1);   //die ersten 4 led's ansteuern
        led(1,1);   //die letzten 4 led's ansteuern
        //Helligkeit mit eingeschaltetem IR-Sender messen und Differenz errechnen
        WandX += -1 * (Umgebungswert - analog(8));
        WandY += 0 * (Umgebungswert - analog(8));
        led(0,0);
        led(1,0);

        lcd_puts("X: ");
        lcd_int(WandX);
        lcd_puts("     ");
        lcd_setxy(1,0);
        lcd_puts("Y: ");
        lcd_int(WandY);
        lcd_puts("     ");
        lcd_home();
}



void IR_PROC(void)
{
        if(mod_ir0_status()>= IR_PROC_PERIODS_FOR_OK){IR0=1;}
        if(mod_ir1_status()>= IR_PROC_PERIODS_FOR_OK){IR1=1;}
        if(mod_ir2_status()>= IR_PROC_PERIODS_FOR_OK){IR2=1;}
        if(mod_ir3_status()>= IR_PROC_PERIODS_FOR_OK){IR3=1;}
```

```
        if(IR0==1)
        {
            WinkelGegner = winkel - 45;
            IR0=0;
            WinkelNeu = 1;
            IgnoreWand = akt_time() + 500;
        }else{
            if(IR0==0 && IR1==1 && IR2==0)
            {
                WinkelGegner = winkel + 25;   //+45
                IR1=0;
                WinkelNeu = 1;
            }else{
                if(IR2==1 && IR0==0)
                {
                    WinkelGegner = winkel + 135;
                    IR2=0;
                    WinkelNeu = 1;
                }else{
                    if(IR0==0 && IR3==1 && IR2==0)
                    {
                        WinkelGegner = winkel - 115;    //-135
                        IR3=0;
                        WinkelNeu = 1;
                    }
                }
            }
        }
    }

/*
void IR_TEST(void)
{
    while(1)
    {
        led(0,1);
        led(1,1);

        lcd_setxy(0,0);
        lcd_int(analog(0));
        lcd_putchar(' ');

        lcd_setxy(0,4);
        lcd_int(analog(1));
        lcd_putchar(' ');

        lcd_setxy(0,8);
        lcd_int(analog(2));
        lcd_putchar(' ');

        lcd_setxy(0,12);
        lcd_int(analog(3));
        lcd_putchar(' ');

        lcd_setxy(1,0);
        lcd_int(analog(4));
        lcd_putchar(' ');

        lcd_setxy(1,4);
        lcd_int(analog(5));
        lcd_putchar(' ');

        lcd_setxy(1,8);
        lcd_int(analog(6));
        lcd_putchar(' ');

        lcd_setxy(1,12);
        lcd_int(analog(7));
        lcd_putchar(' ');
    };//Endlos-Schleife
}
*/

void Fahre(void){
```

```
        // Richtung: 1 == Rückwärts | 0 == Vorwärts
        // Motor: 0 == Links | 2 == Rechts
        if (WandY >= 5){
            if(WandX > 0) {
                motor_richtung(0,1);
                motor_richtung(2,0);
            } else {
                if(WandX <= 0) {
                    motor_richtung(2, 1);
                    motor_richtung(0, 0);
                }
            }
        }else{
            motor_richtung(2, 0);
            motor_richtung(0, 0);
        }

}


void Drehe(int grad){
            // Richtung: 1 == Rückwärts | 0 == Vorwärts
            // Motor: 0 == Links | 2 == Rechts
    if(grad <=0){
        motor_richtung(2,1);
        //motor_pwm(2, 0);
        if(grad>-23){
        sleep((int) (grad*-1));
        }
        else{
            sleep((int) (grad*-0.5));
        }
        //sleep((int) (1,5*((grad*(-1)+1)/log10f((grad*-1)+1))));
        motor_richtung(2,0);
        //motor_pwm(2, 10);
        WinkelNeu = 0;
    }
    else{
        motor_richtung(0,1);
        //motor_pwm(0, 0);
        if(grad>23){
            sleep((int) (grad*0.5));
        }
        else
        {
            sleep((int) (grad));
        }
        //sleep((int) (1,5*(grad/(log10f(grad)+1))));
        motor_richtung(0,0);
        //motor_pwm(0, 9);

        WinkelNeu = 0;
    }
}

void DreheHase(int grad){
            // Richtung: 1 == Rückwärts | 0 == Vorwärts
            // Motor: 0 == Links | 2 == Rechts
    if(grad <=0){
        motor_richtung(0,1);
        //motor_pwm(2, 0);
        sleep((int)(67-grad*-1.5));
        motor_richtung(0,0);
        //motor_pwm(2, 10);
        WinkelNeu = 0;
    }
    else{
        motor_richtung(2,1);
        //motor_pwm(0, 0);
        sleep((int)(67-grad*1.5));
        motor_richtung(2,0);
        //motor_pwm(0, 9);

        WinkelNeu = 0;
    }
```

```
}


void VerfolgeHase(void){
    if(WinkelNeu){
        Drehe(WinkelGegner);
        WinkelNeu = 0;
    }
    else{
        Fahre();
    }

}

void Hase(void){
    if(WinkelNeu) {
        DreheHase(WinkelGegner);
        WinkelNeu = 0;
    }
    else{
        Fahre();
    }


}

 /*
  * rotiert den Kopf
  */
 void MyServo(void)
 {
    //#define KOPF_GRAD 3 //Winkel Schrittweite


    if((winkel <= 90)&&richtung == 0)
    {
        winkel+=w_step;
    }

    else
    {
        richtung=1;         //Richtungswechsel
                            //IR_Empfänger aktivieren???
        winkel-=w_step;
    }



    if(IR0 == 1){
        //IR0 = 0;
                            //IR_Empfänger deaktivieren???
        w_step =7;          //Schrittweite ehöhen wenn Beacon gefunden
        //WinkelGegner = winkel;
    }

    if(winkel<=0)
    {
        richtung=0;         //Richtungswechsel
                            //IR_Empfänger aktivieren===
        w_step = 3;
    }

    if(winkel>=90)
    {
        w_step = 3;
    }

    servo_arc(1, winkel);

}

/**
 *  Startbedingungen
 */
```

```
void startup(int delay)//delay sollte regulär 30sek. betragen
{
    lcd_home();
    lcd_puts("Ready");
    while(digital_in(STARTTRIGGEREINGANG));
    lcd_home();
    lcd_puts("Starte Countdown");
    lcd_setxy(1,0);
    if(dip_pin(DIP_PIN_HASENMODE)) {
        lcd_puts("Hase");
    }
    else {
        lcd_puts("Fuchs");
    }
    lcd_puts("  ");
    if(!dip_pin(DIP_PIN_MAXSPEED)) {
        lcd_puts("langsam");
    }
    else {
        lcd_puts("schnell");
    }
    //Countdownzeit abwarten
    sleep(delay);
    lcd_home();
    lcd_puts("Countdown beendet");
}

//true wenn Rundenzeit vorbei ist
int isRoundtimeOver()
{
    if(globaleVariableEndzeit < akt_time())
        return 1;
    else
        return 0;
}

/*
 * sub-Main (Start)
 */
void AksenMain(void)
{
    char MotorNullMax;
    char MotorZweiMax;

    if(!dip_pin(DIP_PIN_MAXSPEED))
    {
        MotorNullMax = 4;
        MotorZweiMax = 5;

    }else
    {
        MotorNullMax = 8;
        MotorZweiMax = 9;
    }

    startup(STARTDELAY);
    if(dip_pin(DIP_PIN_ROUNDTIME)){
        globaleVariableEndzeit = akt_time() + (long) ROUNDTIME2MIN;
    }
    else {
        globaleVariableEndzeit = akt_time() + (long) ROUNDTIME1MIN;
    }

    if(dip_pin(DIP_PIN_HASENMODE))              //Hase //erster Dipschalter auf OFF
    {
        //Motoren starten
        motor_pwm(0, MotorNullMax);
        motor_pwm(2, MotorZweiMax);


        //IR Empfänger initialisieren
        IR_PROC_PERIODS_FOR_OK = 2;             //notw. Anzahl für Detektion
        IR_PROC_HALF_PERIOD = 4;                //125Hz empfangen
        IR_PROC_MAX_ERROR_PER_PERIOD = 4;       //erlaubte Fehlertakte
        setze_ir_senden(5);                     //100Hz senden
```

```
        mod_ir0_maxfehler(IR_PROC_MAX_ERROR_PER_PERIOD);
        mod_ir1_maxfehler(IR_PROC_MAX_ERROR_PER_PERIOD);
        mod_ir2_maxfehler(IR_PROC_MAX_ERROR_PER_PERIOD);
        mod_ir3_maxfehler(IR_PROC_MAX_ERROR_PER_PERIOD);
        mod_ir0_takt(IR_PROC_HALF_PERIOD);      /* Detektor 1 an */
        mod_ir1_takt(IR_PROC_HALF_PERIOD);      /* Detektor 2 an */
        mod_ir2_takt(IR_PROC_HALF_PERIOD);      /* Detektor 3 an */
        mod_ir3_takt(IR_PROC_HALF_PERIOD);      /* Detektor 4 an */
        mod_ir_empf0_gutzaehler = 0;            /* mod_irX_status auf 0 */
        mod_ir_empf1_gutzaehler = 0;            /* mod_irX_status auf 0 */
        mod_ir_empf2_gutzaehler = 0;            /* mod_irX_status auf 0 */
        mod_ir_empf3_gutzaehler = 0;            /* mod_irX_status auf 0 */
        while(!isRoundtimeOver()){
        //while(1){
            FindeWand();
            Hase();
            MyServo();
            IR_PROC();
        }

    }
    else                                        //Fuchs //erster Dipschalter auf ON
    {
        //Motoren starten
        motor_pwm(0, MotorNullMax);
        motor_pwm(2, MotorZweiMax);

        //IR Empfänger initialisieren
        IR_PROC_PERIODS_FOR_OK = 2;             //notw. Anzahl für Detektion
        IR_PROC_HALF_PERIOD = 5;                //100Hz empfangen
        IR_PROC_MAX_ERROR_PER_PERIOD = 4;       //erlaubte Fehlertakte
        setze_ir_senden(4);                     //125Hz senden
        mod_ir0_maxfehler(IR_PROC_MAX_ERROR_PER_PERIOD);
        mod_ir1_maxfehler(IR_PROC_MAX_ERROR_PER_PERIOD);
        mod_ir2_maxfehler(IR_PROC_MAX_ERROR_PER_PERIOD);
        mod_ir3_maxfehler(IR_PROC_MAX_ERROR_PER_PERIOD);
        mod_ir0_takt(IR_PROC_HALF_PERIOD);      /* Detektor 1 an */
        mod_ir1_takt(IR_PROC_HALF_PERIOD);      /* Detektor 2 an */
        mod_ir2_takt(IR_PROC_HALF_PERIOD);      /* Detektor 3 an */
        mod_ir3_takt(IR_PROC_HALF_PERIOD);      /* Detektor 4 an */
        mod_ir_empf0_gutzaehler = 0;            /* mod_irX_status auf 0 */
        mod_ir_empf1_gutzaehler = 0;            /* mod_irX_status auf 0 */
        mod_ir_empf2_gutzaehler = 0;            /* mod_irX_status auf 0 */
        mod_ir_empf3_gutzaehler = 0;            /* mod_irX_status auf 0 */

        while(!isRoundtimeOver()){
            if(akt_time() > IgnoreWand){
                FindeWand();
            }
            else
            {
                sleep(40);//Simuliert Laufzeit von findeWand()
            }

            VerfolgeHase();
            //Fahre();
            MyServo();
            IR_PROC();
        }
    }

    //Motoren stoppen
    motor_pwm(0,0);
    motor_pwm(2,0);
    lcd_home();
    lcd_puts("    THE         ");
    lcd_setxy(1,0);
    lcd_puts("        END     ");
    while(1);
}
```