PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ

FACULTAD DE CIENCIAS E INGENIERÍA

DISEÑO E IMPLEMENTACIÓN DE UN ROBOT DE BAJO COSTO PARA ESTIMULAR LA FOTOSÍNTESIS EN PLANTAS DE PEQUEÑO TAMAÑO

ANEXO IV

```
;*****************************************************
; Programa de Tesis
; Angel Barragán Neira
; PB0-5 Alimentación de emisores Infrarrojos
;                   PB 1. (arriba)
;    PB 2.          PB 4. (abajo)    PB 0.
;
;
;                   PB 5.(arriba)
;                   PB 3.(abajo)
; PC0-5 Entrada de ADC interno
;                   PC 1.
;    PC 2.                          PC 0.
;
;    PC 3.                          PC 5.
;                   PC 4.
; PD0-5 Entrada de receptores infrarrojos
;                   PD 1. (arriba)
;    PD 2.          PD 4. (abajo)    PD 0.
;
;
;                   PD 5. (arriba)
;                   PD 3. (abajo)
; PD6 y PD7 Salida a los servos (PD7->Izquierda / PD6->Derecha)
; *****************************************************

.include "D:\PROGRA~1\VMLAB\include\m8def.inc"
.dseg
                    .org $60
LDR0: .byte 1               ;Se guardarán en estos Registros los valores digitales de los sensores
LDR1: .byte 1               ;de luz LDRs
LDR2: .byte 1
LDR3: .byte 1
LDR4: .byte 1
LDR5: .byte 1
.cseg
                    .org $0
reset:
   rjmp start
   reti
   reti
   reti
   reti
   reti
   rjmp OCR1A_alta          ;Interrupción de OCR1A de Timer 1
   reti
   reti
   reti
   reti
   reti
   reti
   reti
   rjmp adc_int             ;Interrupción de culmincación de conversión de un valor del ADC
   reti
   reti
```

```
;***********************PROGRAMA PRINCIPAL**********************************
;***************************************************************************
start:
  ldi r16,high(RAMEND)          ;Se comienza guardando la Pila
  out SPH,r16
  ldi r16,low(RAMEND)
  out SPL,r16

  rcall Inireg              ;Se inicializa los registros que se utilizaran en el programa.Se configuran los
  rcall Iniports            ;puertos de entrada y salida como lo especifica el encabezado del programa.
  rcall TimerConfig             ;Además se configura el Timer 1 y el ADC
  rcall ADCConfig

  sei                           ;Se habilitan las Interrupciones

Prog_Principal:
push r16

  cpi r30, $0f                  ;Se compara si es que el robot ha pasado 2 minutos encendido
  brlo seguir_con_logica        ;Se va a modo sleep si la comparación es existosa de lo
  rjmp ir_sleep_segundos        ;contrario sigue con el programa

seguir_con_logica:
        rcall logica

pop r16
rjmp Prog_Principal

;***********************SUBRUTINAS DE PROGRAMACIÓN**************************
;***************************************************************************
logica:
push r16

        rcall obtener_max       ;Subrutinas que obtiene el máximo y mínimo valor entre los registro
                                de valores de los LDRs.
        rcall obtener_min       ;en donde cada una guarda el valor máximo y minimo y el número del
                                LDR correspondiente
                                ;en registros independientes
    mov r16, r29
    andi r16, 0b00000111
                cpi r16, 0b00000101     ; Comparamos si ha transcurrido 0.25 segundos
                breq ir_chequear_IRs
                cpi r16, 0b00000000     ; Comparamos si ha transcurrido 0.25 segundos
                breq ir_chequear_IRs    ; Si efectivamente transcurrieron 0.25 segundos, el programa
                                        ; va a revisar los valores de los
    rjmp no_chequear_IRs                ; sensores infrarrojos en busca de obstáculos
ir_chequear_IRs:
  rcall chequear_IR05           ; Se evalua si el Sensor de Piso de adelante no sensa el piso
  rcall chequear_IR04           ; Se evalua si el Sensor de Piso de atras no sensa el piso
        rcall chequear_IR00             ; Se evalua si el Sensor de Obstáculos de la izquierda atras
                                        ; sensa un obstáculo
        rcall chequear_IR01             ; Se evalua si el Sensor de Obstáculos de atras sensa un
                                        ; obstáculo
        rcall chequear_IR02             ; Se evalua si el Sensor de Obstáculos de la derecha atras
                                        ; sensa un obstáculo
        rcall chequear_IR03             ; Se evalua si el Sensor de Obstáculos de adelante sensa un
                                        ; obstáculo
no_chequear_IRs:
        cpi r28, $ff                    ; Compara si es que la bandera de obstaculos o falta de piso
                                        ; esta activa
```

```
        breq obviar_logica          ; Si se encuentra activa, se salta la lógica que contiene el
                                    ; accionamiento de los motores
        rcall logica_claro_oscuro
obviar_logica:

pop r16
ret

;Revisa si el sensor de adelante sensa el piso
; Entradas: -
; Salidas: r28
chequear_IR05:
push r16

  ldi r31, $00
  ldi r16, 0b11011111
  out portb, r16

confirmar_IR05:
  in r16, pind
  andi r16, 0b00100000
  cpi r16, 0b00000000
  breq terminar_y_limpiar_IR05
  inc r31
  cpi r31, $19
  brne confirmar_IR05
  ldi r28, $00
  rcall adelante
  ldi r29, $00

esperar_1_seg_IR05:
  cpi r29, $19
  brlo esperar_1_seg_IR05
  rjmp ir_sleep_minutos

terminar_y_limpiar_IR05:
  ldi r31, $00
  ldi r28, $00

pop r16
ret

;Revisa si el sensor de atras sensa el piso
; Entradas: -
; Salidas: r28
chequear_IR04:
push r16

  ldi r31, $00
  ldi r16, 0b11101111
  out portb, r16

confirmar_IR04:
  in r16, pind

  andi r16, 0b00010000
  cpi r16, 0b00000000
  breq terminar_y_limpiar_IR04
  inc r31
  cpi r31, $19
```

```
   brne confirmar_IR04
   ldi r28, $00
   rcall atras
   ldi r29, $00

esperar_1_seg_IR04:
   cpi r29, $19
   brlo esperar_1_seg_IR04
   rjmp ir_sleep_minutos

terminar_y_limpiar_IR04:
   ldi r31, $00
          ldi r28, $00

pop r16
ret

;Revisa si el sensor izquierdo de atras sensa un obstaculo
; Entradas: -
; Salidas: r28
chequear_IR00:
push r16

   ldi r31, $00
   ldi r28, $00
   ldi r16, 0b11111110
   out portb, r16

confirmar_IR00:
   in r16, pind
   andi r16, 0b00000001
   cpi r16, 0b00000001
   breq terminar_y_limpiar_IR00
   inc r31
   cpi r31, $19
   brne confirmar_IR00
   rcall atras_derecha
   ldi r16, $31
   rcall esperar_1_seg_IRs
   ldi r16, $20
   rcall esperar_1_seg_IRs
   rcall atras
   ldi r16, $31
   rcall esperar_1_seg_IRs
   ldi r16, $31
   rcall esperar_1_seg_IRs
   rcall atras_izquierda
   ldi r16, $31
   rcall esperar_1_seg_IRs
   ldi r16, $20
   rcall esperar_1_seg_IRs
   rcall adelante
   ldi r16, $31
   rcall esperar_1_seg_IRs

terminar_y_limpiar_IR00:
   ldi r16, $ff
   out portb, r16
   ldi r31, $00
```

```
pop r16
ret

;Revisa si el sensor de atras sensa un obstaculo
; Entradas: -
; Salidas: r28
chequear_IR01:
push r16

   ldi r31, $00
   ldi r28, $00
   ldi r16, 0b11111101
   out portb, r16

confirmar_IR01:
   in r16, pind
   andi r16, 0b00000010
   cpi r16, 0b00000010
   breq terminar_y_limpiar_IR01
   inc r31
   cpi r31, $19
   brne confirmar_IR01
   rcall adelante_derecha
   ldi r16, $27
   rcall esperar_1_seg_IRs
   rcall adelante
   ldi r16, $31
   rcall esperar_1_seg_IRs
   ldi r16, $31
   rcall esperar_1_seg_IRs
   rcall adelante_izquierda
   ldi r16, $27
   rcall esperar_1_seg_IRs
   rcall atras
   ldi r16, $31
   rcall esperar_1_seg_IRs

terminar_y_limpiar_IR01:
   ldi r16, $ff
   out portb, r16
   ldi r31, $00

pop r16
ret

;Revisa si el sensor derecho de atras sensa un obstaculo
; Entradas: -
; Salidas: r28
chequear_IR02:
push r16

   ldi r31, $00
   ldi r28, $00
   ldi r16, 0b11111011
   out portb, r16

confirmar_IR02:
   in r16, pind
   andi r16, 0b00000100
   cpi r16, 0b00000100 ;Para implementar, probamos 1er sensor solamente
```

```
   breq terminar_y_limpiar_IR02
   inc r31
   cpi r31, $19
   brne confirmar_IR02
   rcall adelante_izquierda
   ldi r16, $27
   rcall esperar_1_seg_IRs
   rcall adelante
   ldi r16, $31
   rcall esperar_1_seg_IRs
   ldi r16, $31
   rcall esperar_1_seg_IRs
   rcall adelante_derecha
   ldi r16, $27
   rcall esperar_1_seg_IRs
   rcall atras
   ldi r16, $31
   rcall esperar_1_seg_IRs

terminar_y_limpiar_IR02:
          ldi r16, $ff
          out portb, r16
          ldi r31, $00

pop r16
ret

;Revisa si el sensor de adelante sensa un obstaculo
; Entradas: -
; Salidas: r28
chequear_IR03:
push r16

   ldi r31, $00
   ldi r28, $00
   ldi r16, 0b11110111
   out portb, r16

confirmar_IR03:
   in r16, pind
   andi r16, 0b00001000
   cpi r16, 0b00001000 ;Para implementar, probamos 1er sensor solamente
   breq terminar_y_limpiar_IR03
   inc r31
   cpi r31, $19
   brne confirmar_IR03
   rcall adelante_derecha
   ldi r16, $31
   rcall esperar_1_seg_IRs
   ldi r16, $20
   rcall esperar_1_seg_IRs
   rcall adelante
   ldi r16, $31
   rcall esperar_1_seg_IRs
   ldi r16, $31
   rcall esperar_1_seg_IRs
   rcall adelante_izquierda
   ldi r16, $31
   rcall esperar_1_seg_IRs
   ldi r16, $20
```

```
   rcall esperar_1_seg_IRs
   rcall atras
   ldi r16, $31
   rcall esperar_1_seg_IRs

terminar_y_limpiar_IR03:
   ldi r16, $ff
   out portb, r16
   ldi r31, $00

pop r16
ret

;Espera 1 segundo
; Entradas: -
; Salidas: -
esperar_1_seg_IRs:

   ldi r29, $00

esperar_1_seg_IRs_lazo:
   rcall chequear_IR05
   rcall chequear_IR04
   cp r29, r16
   brlo esperar_1_seg_IRs_lazo

ret

logica_claro_oscuro:
push r16

   mov r16, r14
   cpi r16,05
   brlo no_moverse
   rcall apuntar_a_luz
   pop r16

ret

;Comanda a los motores no moverse
; Entradas: -
; Salidas: -
no_moverse:
push r16

   rcall detenido

pop r16
ret

;Obtiene el mínimo valor entre los registro de valores de los LDRs.
; Entradas: -
; Salidas:          r14: Valor del LDR mayor
;                                   r24: Identifica el LDR con el valor en r14
obtener_min:
push r23

   ldi r23, $00
   ldi r24, $00
   ldi XH, high(LDR0)
```

```
   ldi XL, low(LDR0)
   ld r15, x+
   mov r14, r15

seguir_comparando_min:
   ld r15, x+
   cp r14,r15
   brlo sigue_min
   mov r14, r15
   mov r24,r23
   inc r24
   cpi r23, $04
   breq terminar_obtener_min
   inc r23
   rjmp seguir_comparando_min

sigue_min:
   cpi r23, $04
   breq terminar_obtener_min
   inc r23
   rjmp seguir_comparando_min

terminar_obtener_min:

pop r23
ret

;Obtiene el máximo valor entre los registro de valores de los LDRs.
; Entradas: -
; Salidas:          r13: Valor del LDR menor
;                                   r23: Identifica el LDR con el valor en r13

obtener_max:
push r23

   ldi r23, $00
   ldi r25, $00
   ldi XH, high(LDR0)
   ldi XL, low(LDR0)
   ld r15, x+
   mov r13, r15

seguir_comparando_max:
   ld r15, x+
   cp r15,r13
   brlo sigue_max
   mov r13, r15
   mov r25,r23
   inc r25
   cpi r23, $04
   breq terminar_obtener_max
   inc r23
   rjmp seguir_comparando_max

sigue_max:
   cpi r23, $04
   breq terminar_obtener_max
   inc r23
   rjmp seguir_comparando_max
```

```
terminar_obtener_max:

pop r23
ret

;Ejecuta subrutinas de movimiento para poder alinearse con la fuente de luz
; Entradas: r24: Valor del LDR mayor
; Salidas:
apuntar_a_luz:
push r16

   cpi r24, $00
   brne apuntar1
   rcall adelante

apuntar1:
   cpi r24, $01
   brne apuntar2
   rcall adelante_derecha

apuntar2:
   cpi r24, $02
   brne apuntar3
   rcall atras_izquierda

apuntar3:
   cpi r24, $03
   brne apuntar4
   rcall atras

apuntar4:
   cpi r24, $04
   brne apuntar5
   rcall atras_derecha

apuntar5:
   cpi r24, $05
   brne no_hay_luz
   rcall adelante_izquierda

no_hay_luz:

pop r16
ret

;Subrutina que activa conversión en el ADC
Lee_ADC:

   out ADMUX,r21
   sbi ADCSR, 6

ret

; Subrutina que envia al robot a un modo sleep durante 1 minuto
ir_sleep_segundos:

push r16

   cbi ADCSR, 7                                    ; Paramos el ADC
   ldi          R16, 0b00000000                    ; reconfiguramos Timer 1
```

```
  out         TCCR1A,R16
  ldi         R16, 0b00001101
  out         TCCR1B,R16
  ldi         R16,0b00010100
  out         TIMSK,R16
  ldi         R16,high(9200)
  out         OCR1AH,R16
  ldi         R16,low(9200)
  out         OCR1AL,R16
  ldi r16, $80
  out MCUCR, r16                                ; Enable Sleep
  ldi r16, $00
  out TCNT1H, r16
  out TCNT1L, r16
  sleep

pop r16
rjmp start

; Subrutina que envía al robot a un modo sleep durante 30 minutos
ir_sleep_minutos:
push r16

  cbi ADCSR, 7                                  ; Paramos el ADC
  ldi         R16, 0b00000000                   ;reconfiguramos Timer 1
  out         TCCR1A,R16
  ldi         R16, 0b00001101
  out         TCCR1B,R16
  ldi         R16,0b00010100
  out         TIMSK,R16
  ldi         R16,high(19530)
  out         OCR1AH,R16
  ldi         R16,low(19530)
  out         OCR1AL,R16
  ldi r16, $80
  out MCUCR, r16                                ; Enable Sleep
  ldi r16, $00
  out TCNT1H, r16
  out TCNT1L, r16
  sleep

pop r16
rjmp start


;**********************INTERRUPCIONES*************************************
;*****************************************************************************
; Esta Interrupción comanda el PWM de los servomotores, los cuales son los que indican la direccion
; de giro de estos. Ademas efectua la conversión del siguiente LDR que necesita ser convertido.
OCR1A_alta:
push r16
in r16, SREG
push r16
  out PORTD, r20
  cpi r18, $00
  breq motor_derecha
  cp r18,r17
  brne motor_derecha
  andi r20, 0b01111111
```

```
motor_derecha:
   cpi r19, $00
   breq control_terminar
   cp r19,r17
   brne control_terminar
   andi r20, 0b10111111

control_terminar:
   inc r17
   cpi r17,$50
   brsh reiniciar_ciclo
   rjmp motores_terminar

reiniciar_ciclo:
   ldi r20, 0b11000000
   out PORTD, r20
   ldi r17, $00
   inc r29
   cpi r29, $32 ;1 segundo
   brne motores_terminar
   ldi r29, $00
   inc r30
   cpi r30, 121
   brne motores_terminar
   ldi r30, 00

motores_terminar:
   rcall Lee_ADC

pop r16
out SREG, r16
pop r16
reti

;Subrutina que realiza la conversion del LDR siguiente en la fila a ser leido y lo guarda en su registro
correspondiente
adc_int:
push r16
in r16, SREG
push r16
   in r16, ADCH
   cpi r21, $60                    ;proceso de guardar
   brne sigue1
   sts LDR0, r16
   rjmp terminar_guardar

sigue1:
   cpi r21, $61
   brne sigue2
   sts LDR1, r16
   rjmp terminar_guardar

sigue2:
   cpi r21, $62
   brne sigue3
   sts LDR2, r16
   rjmp terminar_guardar

sigue3:
   cpi r21, $63
```

```
   brne sigue4
   sts LDR3, r16
   rjmp terminar_guardar

sigue4:
   cpi r21, $64
   brne sigue5
   sts LDR4, r16
   rjmp terminar_guardar

sigue5:
   cpi r21, $65
   brne terminar_guardar
   sts LDR5, r16

terminar_guardar:
          inc r21
          cpi r21, 0b01100110
          brne terminar_adc_int
          ldi r21, 0b01100000

terminar_adc_int:

pop r16
out SREG, r16
pop r16
reti

;**********************MOVIMIENTO DEL ROBOT*********************************
;*****************************************************************************
;para servo izquierdo: 3 adelante y 5 atras
;para servo derecho: 5 adelante y 3 atras
detenido:
   ldi r18, $00              ; servo izq
   ldi r19, $00              ; servo der
ret

adelante_izquierda:
   cpi r28, $ff
   breq terminar_adelante_izquierda
   ldi r18, $00              ; servo izq
   ldi r19, $05              ; servo der
terminar_adelante_izquierda:
ret

adelante:
   cpi r28, $ff
   breq terminar_adelante
   ldi r18, $03              ; servo izq
   ldi r19, $05              ; servo der
terminar_adelante:
ret

adelante_derecha:
   cpi r28, $ff
   breq terminar_adelante_derecha
   ldi r18, $03              ; servo izq
   ldi r19, $00              ; servo der
terminar_adelante_derecha:
ret
```

```
atras_izquierda:
   cpi r28, $ff
   breq terminar_atras_izquierda
   ldi r18, $05              ; servo izq
   ldi r19, $00              ; servo der
terminar_atras_izquierda:
ret

atras_derecha:
   pi r28, $ff
   breq terminar_atras_derecha
   ldi r18, $00              ; servo izq
   ldi r19, $03              ; servo der
terminar_atras_derecha:
ret

atras:
   cpi r28, $ff
   breq terminar_atras
   ldi r18, $05              ; servo izq
   ldi r19, $03              ; servo der
terminar_atras:
ret

;***********************SUBRUTINAS DE CONFIGURACION*************************
;*******************************************************************************

Inireg:
   ldi r16, $00
   ldi r17, $00
   ldi XH, high(LDR0)
   ldi XL, low(LDR0)

ini_tablas:
   st x+,r17
   inc r16
   cpi r16, $06
   brlo ini_tablas

   ldi r17, $00                                    ; Contador para mover Servos
   ldi r18, $00             ; Variable de Servo izquierdo
   ldi r19, $00             ; Variable de Servo derecho
   ldi r20, $00                                    ; CONTROLADOR DE SERVOS
   ldi r21, 0b01100000          ; Selector del ADMUX
   ldi r22, $00                                    ; Indicador de LDR más alto
   ldi r24, $00                                    ; Indicador de que LDR es el más bajo
   ldi r25, $00                                    ; Indicador de que LDR es el más alto
   ldi r28, $00                                    ; Bandera indicadora de obstaculo
   ldi r29, $00                                    ; Contador de 1s
   ldi r30, $00                                    ; Contador de 2 min
   ldi r31, $00                                    ; Contador de Sensores IR

ret

Iniports:
push r16

   ldi r16,0b11000000                    ;PB0-5 Alimentación de Emisores Infrarrojos
   out PORTB,r16             ;
```

```
  ldi r16, 0b00111111
  out DDRB,r16
  ldi r16,0b11000000        ;PC0-5 Entrada de ADC interno
  out PORTC,r16             ;
  ldi r16, 0b00000000
  out DDRC,r16
  ldi r16,0b11000000        ;PD0-5 Entrada de Receptores Infrarrojos
  out PORTD,r16             ;PD6 y PD7 Salida a los Servos(Izquierda/derecha)
  ldi r16, 0b11000000
        out DDRD,r16

pop r16
ret

TimerConfig:
push r16

  ldi           R16, 0b00000000     ; Configura timer1 en modo CTC, PRE=1
  out           TCCR1A,R16          ; OCR1A=249
  ldi           R16, 0b00001001
  out           TCCR1B,R16
  ldi           R16,high(249)       ; en periodo de activacion de OCF1A es de 2.5ms
  out           OCR1AH,R16
  ldi           R16,low(249)
  out           OCR1AL,R16
  ldi           R16,0b00010000
  out           TIMSK,R16

pop r16
ret

ADCConfig:
push r16

  sbi ACSR, 7   ; paramos el Analog Comparator

  ldi r16, 0b01100000
  out ADMUX,r16
  ldi r16, 0b10001011
  out ADCSR,r16

pop r16
ret
```