(6609) Laboratorio de Microcomputadoras

Proyecto:

Trabajo Práctico Integrador

Profesor:	**Ing. Guillermo Campiglio**
Cuatrimestre / Año:	2 cuatrimestre / 2023
Turno de clases prácticas:	Miércoles
Jefe de Trabajos Prácticos:	
Docente guía:	

Autores			**Seguimiento del proyecto**								
Nombre	**Apellido**	**Padrón**									
Agustin	Zuretti	95605									

Observaciones:

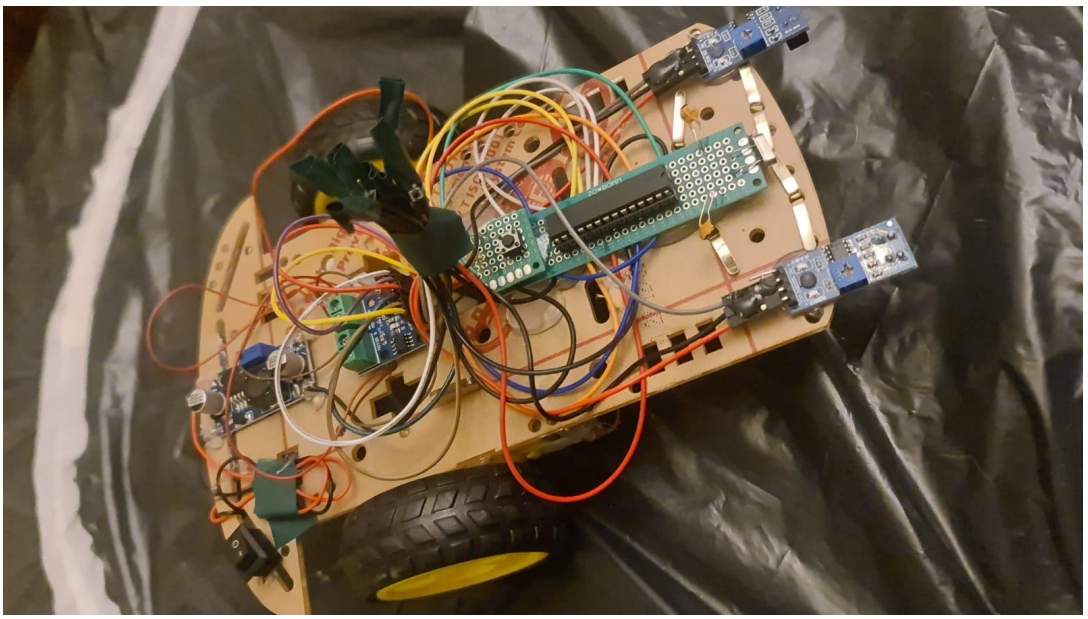
Fecha de aprobación		

Firma J.T.P.

COLOQUIO	
Nota final	
Firma Profesor	

Índice

Introducción

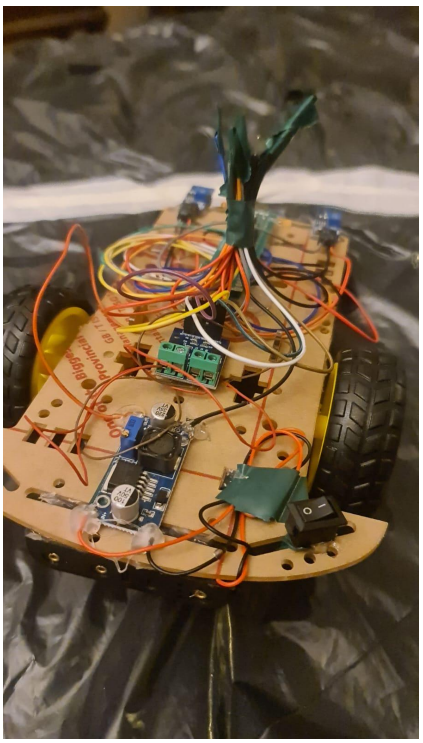


El propósito del trabajo práctico integrador fue la elaboración de un seguidor de línea. Para esta tarea se debían cumplir con ciertos requisitos: esperar 5 segundos al inicio, detectar el fondo (si es blanco o negro) y de pasar más de 2 segundos sin ver la linea, el seguidor debe detener la marcha. Además debíamos soldar directamente el microcontrolador una placa experimental y alimentarlo con una fuente.

Para cumplir con todo esto, se utilizaron materiales que ya poseía y se adquirieron los que no. También se disponía de las herramientas que se encuentran en la facultad.
A continuación se explicará el trabajo realizado en detalle.

COMPONENTES UTILIZADOS:

- Controlador Atmega 328p
- Socket 14x14
- cristal de cuarzo 16Mhz
- 2 capacitores 18 pf
- Placa experimental
- 2 Ruedas
- 2 Motorreductores de 3 a 6 v
- Controlador de motores L9110
- Fuente DC a DC
- 4 pilas recargables
- Soporte de estructura
- Pistola de plástico/Soldador/cables/etc
- 2 sensores infrarrojos
- Botón reset
- Resistencia de pullup 10 kohm



Configuración y elaboración

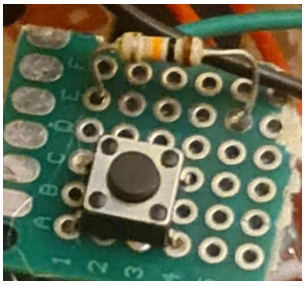
Como mencioné anteriormente, para escribir el código en el microcontrolador se utilizó la placa de arduino, es decir no se compró un programador. Los fuses y demás configuraciones del atmega quedaron de stock. Para reemplazar el cristal que viene en la placa arduino se usó uno idéntico de 16 Mhz, junto con unos capacitores de 18 pf. Se quisieron usar los capacitores de 22pf pero estos no estaban disponibles para su compra. Además se agregó una resistencia de 10k ohms pull up, junto con botón de reset.

Si bien el atmega posee un oscilador interno de 8 Mhz, en la bibliografía se aclara que puede no ser suficientemente preciso, por esto se optó por instalar un cristal externo. Como la configuración de fábrica ya hace uso de un cristal externo (se verificó por software) no fue necesario modificar los fuses del atmega.

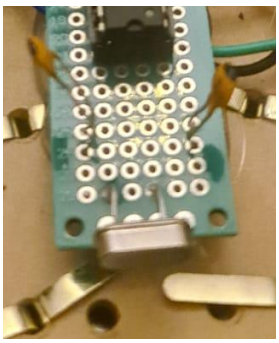
Para seguir la pista se utilizaron 2 sensores infrarrojos, los cuales tienen un canal para salida analógica y otro para digital (además de alimentación y tierra). Para las necesidades de este trabajo práctico se utilizó la salida digital, la cual devuelve 1 si NO se detecta la línea y 0, si se detecta. Para modificar la “sensibilidad” de los detectores se ajustó su potenciómetro hasta que se observó la detección de la línea, mediante el encendido de la luz del sensor (caso fondo negro linea blanca).

Para controlar los motorreductores se usó el controlador de motores L9110, el cual cuenta con dos pines para cada motor. Si los pines están configurados al mismo valor Ej: “1,1” o “0,0” el motor se detiene. Si son opuestos se activan. Entendiendo esto para avanzar para adelante se configuró la dirección de ambos motores en 1 y el duty cycle del pwm en mínimo. Como, si invertimos los valores se invierte el sentido, se hizo uso de esto para girar controlando dos ruedas. Cuando se quiere girar a la derecha, la rueda derecha funciona en reversa y la izquierda avanza. Lo contrario para girar a la izquierda. Cuando se quieren detener los motores ambos valores coinciden en cada motor.

Botón de reset y Resistencia Pull up

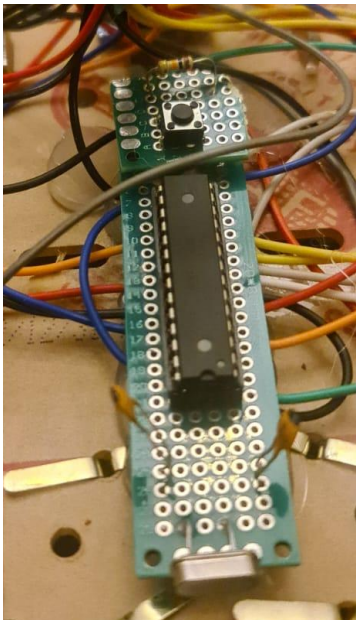


Cristal Externo 16 Mhz y capacitores



Placa arduino uno V3, sin el controlador. Se utilizó como programador

Atmega 328p en el socket



Implementación de la rutina: Explicación

Para implementar el código del seguidor, se seleccionaron los Timer 1 y Timer 2 del atmega328p. El timer 1 al ser de 16 bits permite más resolución (que los otros dos), lo que le otorga mayor precisión a la medición. El timer 2 fue suficiente para configurar una señal pwm capaz de controlar los motores. El timer 0 no se utilizó.

Cuando comienza la rutina el seguidor debe esperar 5 segundos. Para esto se configura el Timer 1 en modo normal, seleccionado prescaler y cargando el registro TCNT1 (High y Low) de forma de generar una interrupción de 1 segundo teniendo en cuenta la frecuencia del microprocesador de 16 Mhz. Luego se llama a esta rutina 5 veces para llegar al tiempo deseado. Una vez finalizado se limpian los registros de configuración del timer 1 para volver a ser utilizados más adelante.

Luego se setean los puertos entrada y salida. Esta parte se puede ver mejor en el diagrama esquemático: se usa el puerto C para la entrada de los sensores infrarrojos, las direcciones de los motores del controlador se pusieron arbitrariamente en PD5 y PD6 y se utilizaron los pines pwm del timer 2 como salida (PB3 Y PD3).

Después sigue la configuración del timer 2 para generar señal la pwm de los motores, para esto se optó por el modo Fast Pwm con el top de 255. Se ajustaron los valores de OC2A y OC2B de forma que el seguidor pudiera avanzar, girar y detenerse adecuadamente.

Se continua el código con la re configuración del timer 1 para un delay asíncrono de 2 segundos. Este delay es el encargado de la función de detener el motor si se pierde la línea. Como el seguidor está constantemente chocando con la línea para corregir el rumbo, se verifica que pasen varios ciclos sin haber detectado la línea para detener ambos motores. Para lograr lo anteriormente dicho se optó por el modo CTC (Clear Timer on Compare Match), el cual puede ser usado para generar

interrupciones cada ciertos intervalos de tiempo. Si excede un límite de varios ciclos desde que se vio por última vez la línea, se llama en loop a detener motores para finalizar la marcha.

Luego sigue la rutina que hace la detección del fondo, esto es simplemente llamar a leer sensores y guardar si el fondo es negro o blanco en un registro.

En Leer sensores se guarda el estado de entrada del puerto y se le aplica la máscara de los sensores. Se le agregó algo de lógica para detectar o no si vio la línea durante el ciclo.

A continuación dependiendo del resultado de “detectar fondo” se llama un loop principal que llama decidir movimiento y leer sensores continuamente. De esta manera me aseguro que el seguidor no se sale de la pista. La función de decidir movimiento se duplicó por simplicidad. Si el fondo es negro se entra al loop de “decidir_movimiento_negro”, e identicamente para blanco.

Decidir movimiento se mueve a rutinas para avanzar, detener, girar a la derecha o a la izquierda correspondientemente a la entrada de los sensores. Recordemos que el sensor se activa en 0 y si no detecta nada queda en 1. Entonces se configuraron los movimientos correspondientes a los motores.

Para el caso de **fondo negro** por ejemplo

Sensor Izquierdo	Sensor Derecho	Rutina
1	1	avanzar
1	0	girar a la derecha
0	1	girar a la izquierda
0	0	detener

E inversamente para fondo blanco.

Por último el controlador de motores se setea en cada caso con valores arbitrarios decididos con la experimentación en la pista. En la sección de configuración y elaboración se detalló cómo se configuran los motores a través del controlador L9110.

Diagrama de flujo

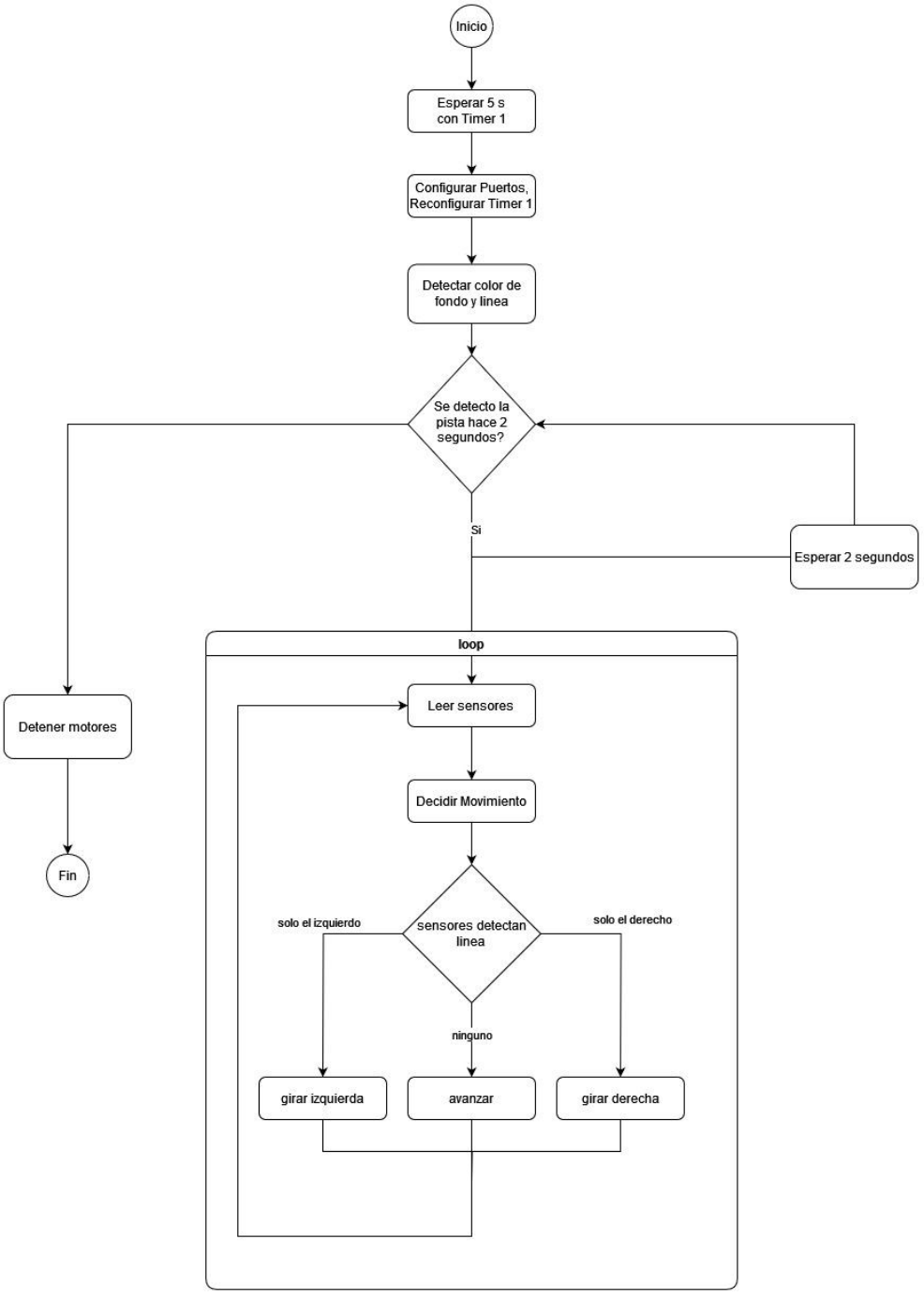
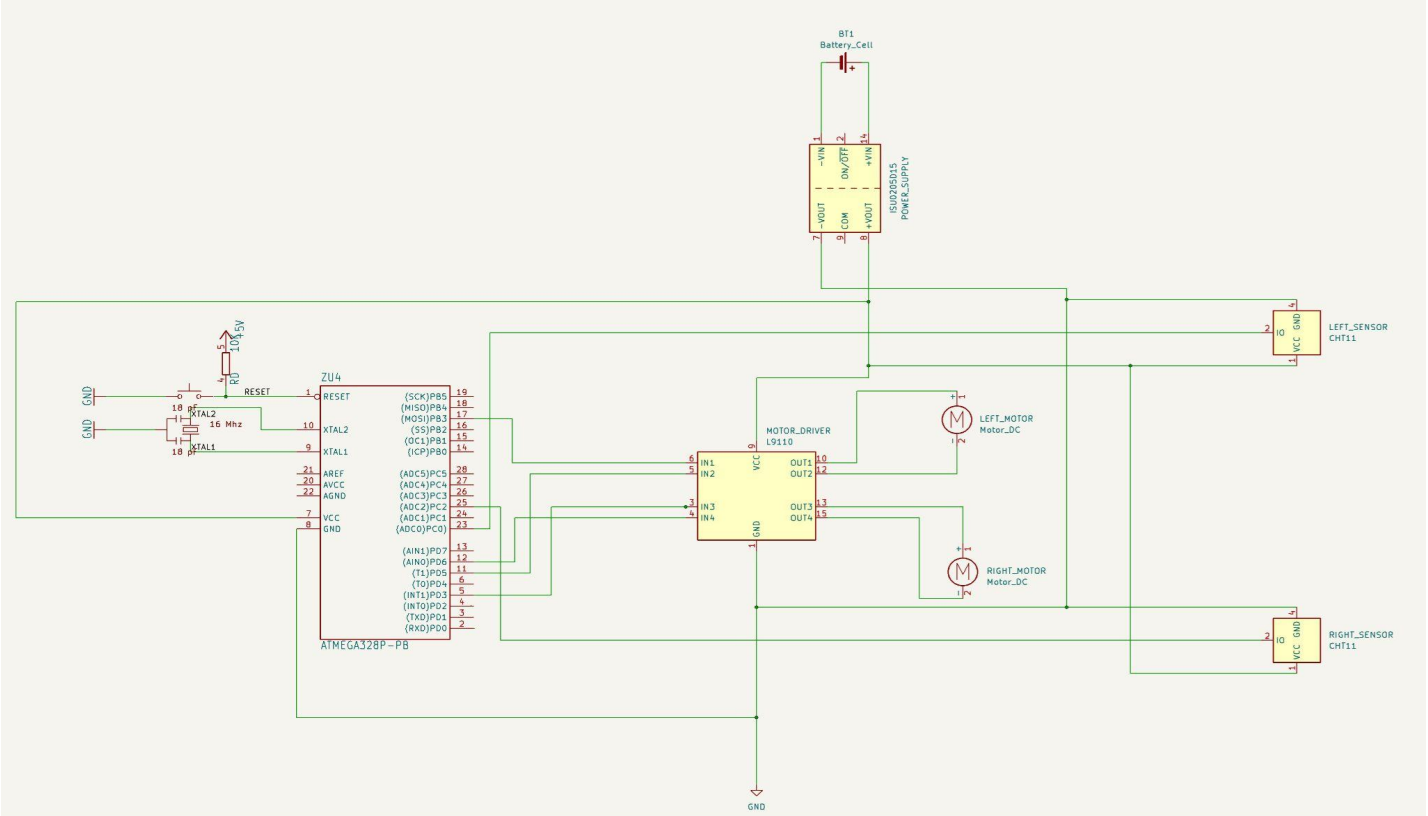


Diagrama Esquemático



Código

```
/*
 * Trabajo Practico Integrador.asm
 *
 *   Author: Zuretti Agustin
 */

.def temp = r20
.def ldiTemp = r16
.def temp2 = r17
.def fondo_detectado = r18
.def estadoSensores = r19
.def contadorSinLinea = r21

.equ PIN_DIR_IZQ = PD5
.equ PIN_DIR_DER = PD6
.equ PIN_SENSOR_IZQ = PC0
.equ PIN_SENSOR_DER = PC2

.equ DUTY_CYCLE_MIN = 20
.equ DUTY_CYCLE_STRONG = 80

.equ F_CPU = 16000000
.equ PRESCALER = 1024
.equ SECONDS = 2
.equ OCR1A_VAL = ((F_CPU / PRESCALER) * SECONDS) - 1

.equ TIMER1_CONTAR = 49911
.equ limiteSinLinea = 2

; macro para ajustar el duty cycle
.macro dutycycle
    ldi temp2, @0*255/100;
.endmacro

; macro para cargar registros menores al 16
.macro ldi_reg
    ldi ldiTemp, @1
    mov @0, ldiTemp
```

```
.endmacro

.cseg

.org 0x0000
    jmp main
.org        0x0016
    jmp     TIMER1_COMPA_vect
    rjmp loop_select

.org        INT_VECTORS_SIZE

configurar_puertos:
    ; Inicializo el contador de veces sin ver la linea en 0
    clr contadorSinLinea

    ; Configurar como entrada los pines conectados a los sensores
infrarrojos
    cbi DDRC, PIN_SENSOR_IZQ
    cbi DDRC, PIN_SENSOR_DER

    ; Configurar como salida los pines conectados a dirección de
motores
    sbi DDRD, PIN_DIR_IZQ
    sbi DDRD, PIN_DIR_DER

    ; Configurar PB3 y PD3 como salida para PWM (motores)
    sbi DDRB, PB3
    sbi DDRD, PD3
    ret

configurar_timer2_PWM:
    ; Configurar el Timer2 para Fast PWM
    clr temp
    ldi temp, (1<<WGM21)|(1<<WGM20)
    sts TCCR2A, temp

    ; Configurar para Clear OC2A/OC2B on compare match, set at BOTTOM
    ori Temp, (1<<COM2A1)|(1<<COM2B1)
    sts TCCR2A, temp

    ; Establecer prescaler a 64
    ldi temp, (1<<CS22)
```

```
    sts TCCR2B, temp

    ; Inicializar registros OCR2A y OCR2B a 0
    clr temp
    sts OCR2A, temp
    sts OCR2B, temp
    ret

limpiar_timer1:
    ; Limpio todos los registros
    ldi     temp, 0
    sts     TCCR1B, temp

    sts     TCCR1A, temp
    sts     TCNT1H, temp
    sts     TCNT1L, temp
    sts     TIMSK1, temp
    ret

configurar_timer1_primer_delay:
    ldi     temp, 0
    sts     TCCR1A, temp
    ldi     temp, (1<<CS12) | (1<<CS10)
    sts     TCCR1B, temp
    ret

Config_timer1_delay:
    clr temp
    ; Configurar el modo CTC del Timer1 y establecer el prescaler a
1024
    ldi temp, (1<<WGM12)|(1<<CS12)|(1<<CS10)
    sts TCCR1B, temp

    ldi temp, high(OCR1A_VAL)
    sts OCR1AH, temp
    ldi temp, low(OCR1A_VAL)
    sts OCR1AL, temp

    ; Habilitar la interrupción por comparación en A
    ldi temp, (1<<OCIE1A)
    sts TIMSK1, temp
    sei
    ret
```

```
leer_sensores:
    push r6

    in estadoSensores, PINC
    andi estadoSensores, 0b00000101

    ; Comprobar si al menos un sensor detecta la línea durante el ciclo
correspondiente

    cpi estadoSensores, 0b00000001
    breq linea_detectada
    cpi estadoSensores, 0b00000100
    breq linea_detectada
    cpi estadoSensores, 0b00000000
    breq linea_detectada

    ; Ningún sensor detecta la línea, cargo 0 en r6 usando la macro

    ldi_reg r6, 0                           ;  r6 = 0
    rjmp fin_leer_sensores

linea_detectada:
    ldi_reg r6, 1                           ;  r6 = 1

fin_leer_sensores:
    pop r6
    ret

chequeo_fondo:

    rcall leer_sensores             ; Llama a la subrutina que lee los
sensores
    cpi estadoSensores, 0b00000101
    brne fondo_es_blanco
    ldi fondo_detectado, 0              ; 'fondo_detectado' = 1
    rjmp fin_deteccion_fondo

fondo_es_blanco:
    ldi fondo_detectado, 1              ; 'fondo_detectado' = 1

fin_deteccion_fondo:
    ret
```

```
loop_select:
    cpi fondo_detectado, 0       ; Compara el registro con 0 (fondo
negro)
    breq fondo_negro
    rjmp loop_blanco
fondo_negro:
    rjmp loop_negro

main:
    ; Initializar Stack Pointer
    ldi     temp, LOW(RAMEND)
    out     SPL, temp
    ldi     temp, HIGH(RAMEND)
    out     SPH, temp

    rcall configurar_timer1_primer_delay

    ; Delay inicial de 5 segundos
    rcall delay1s
    rcall delay1s
    rcall delay1s
    rcall delay1s
    rcall delay1s

    ;Reseteo el timer 1
    rcall limpiar_timer1

    rcall configurar_puertos
    rcall configurar_timer2_PWM
    rcall config_timer1_delay
    rcall chequeo_fondo
    rjmp loop_select


loop_negro:
    rcall leer_sensores
    rcall decidir_movimiento_negro
    rjmp loop_negro

loop_blanco:
    rcall leer_sensores
    rcall decidir_movimiento_blanco
```

```
    rjmp loop_blanco


decidir_movimiento_negro:

    cpi estadoSensores, 0x00
    breq detener                    ; Ambos sensores en 0: detener

    ldi temp, 0b00000101
    cp estadoSensores, Temp
    breq avanzar                ; Ambos sensores en 1: avanzar

    ldi Temp, (1 << PIN_SENSOR_DER)  ; Solo sensor derecho en 1, giro
derecha
    cp estadoSensores, temp
    breq giro_derecha

    ldi Temp, (1 << PIN_SENSOR_IZQ) ; Solo sensor izquierdo en 1, giro
izquierda
    cp estadoSensores, Temp
    breq giro_izquierda

    rjmp end_decidir_mov_negro

end_decidir_mov_negro:
    ret

decidir_movimiento_blanco:

    cpi estadoSensores, 0b00000101
    breq detener                ; Ambos sensores en 1: detener

    ldi Temp, 0x00
    cp estadoSensores, Temp
    breq avanzar                ; Ambos sensores en 0: avanzar

    ldi Temp, (1 << PIN_SENSOR_DER)
    cp estadoSensores, Temp
    brne giro_izquierda         ; Solo sensor derecho en 0: giro
izquierda

    ldi Temp, (1 << PIN_SENSOR_IZQ)
    cp estadoSensores, Temp
```

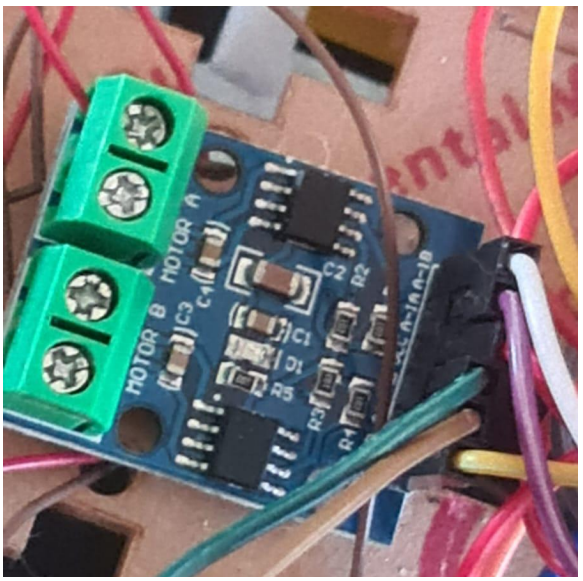
```
    brne giro_derecha           ; Solo sensor izquierdo en 0: giro
derecha

    rjmp end_decidir_mov_blanco

end_decidir_mov_blanco:
    ret


avanzar:

    sbi PORTD, PIN_DIR_IZQ
    sbi PORTD, PIN_DIR_DER


    dutycycle DUTY_CYCLE_MIN
    sts OCR2A, Temp2
    dutycycle DUTY_CYCLE_MIN
    sts OCR2B, Temp2
    ret

giro_derecha:

    cbi PORTD, PIN_DIR_IZQ
    sbi PORTD, PIN_DIR_DER


    dutycycle 75
    sts OCR2A, Temp2
    dutycycle 0
    sts OCR2B, Temp2
    ret

giro_izquierda:

    sbi PORTD, PIN_DIR_IZQ
    cbi PORTD, PIN_DIR_DER


    dutycycle DUTY_CYCLE_MIN
    sts OCR2A, Temp2
    dutycycle 75
    sts OCR2B, Temp2
```

```
    ret

detener:

    cbi PORTD, PIN_DIR_IZQ
    cbi PORTD, PIN_DIR_DER

    dutycycle 0
    sts OCR2A, Temp2
    dutycycle 0
    sts OCR2B, Temp2
    ret

TIMER1_COMPA_vect:
    push temp
    ; Verificar si se detectó la línea en el ultimo ciclo
    ldi temp, 0
    cp r6, temp
    breq linea_no_detectada  ; Si no se detecta la linea salto
    clr contadorSinLinea     ; Si se detecta la línea, reinicia el
contador
    rjmp fin_interupcion

linea_no_detectada:
    inc contadorSinLinea

    ldi temp, limiteSinLinea
    cp contadorSinLinea, temp
    brlo fin_interupcion

    rjmp detener_loop

detener_loop:
    rcall detener
    rjmp detener_loop

fin_interupcion:
    pop temp
    reti


delay1s:
```

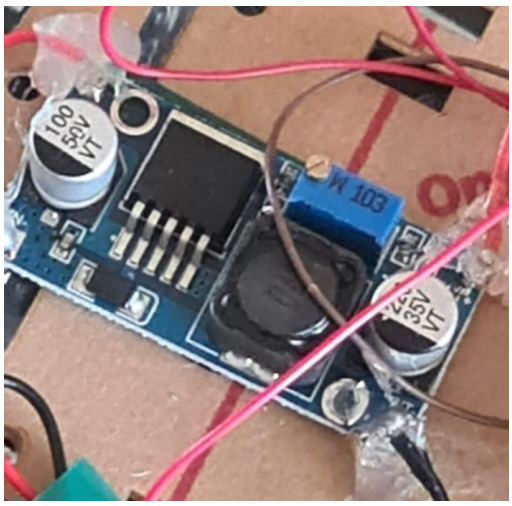
```
    ldi     temp, HIGH(TIMER1_CONTAR)
    sts     TCNT1H, temp
    ldi     temp, LOW(TIMER1_CONTAR)
    sts     TCNT1L, temp

esperar:
    in      temp, TIFR1
    sbrs    temp, TOV1
    rjmp    esperar

    ldi     temp, (1<<TOV1)
    out     TIFR1, temp

    ret
```

Anexo: Imágenes adicionales

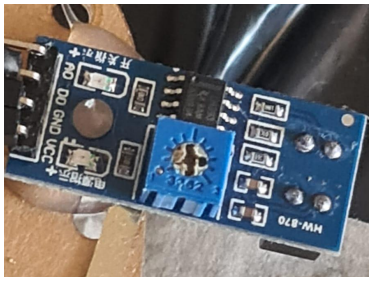
Driver de motores L9110



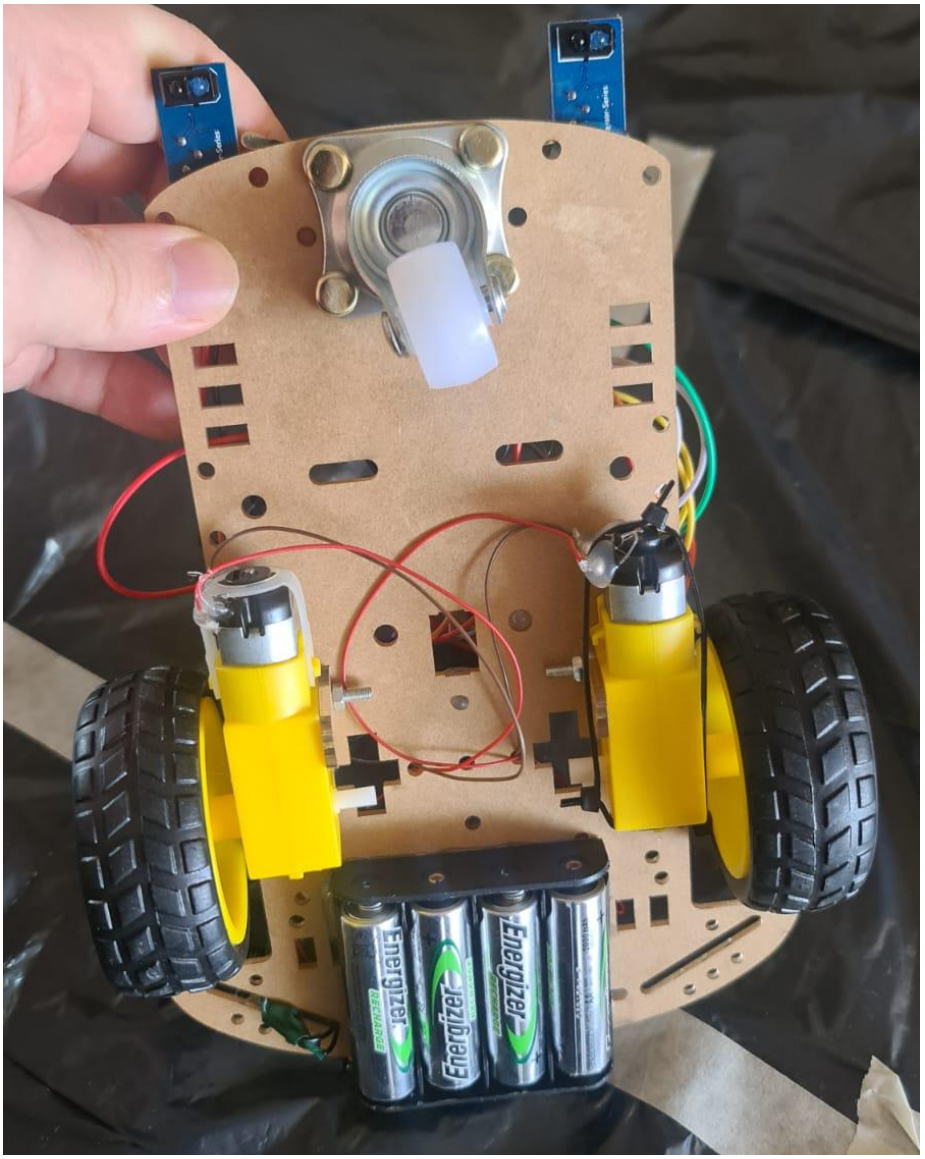
Fuente de alimentacion DC a DC



Sensor Infrarrojo



Vista por debajo del seguidor de linea: se observan los motorreductores y las baterias


Energizer
Energizer