FACULTAD
DE INGENIERIA
Universidad de Buenos Aires

86.07 - Laboratorio de Microprocesadores

2do Cuatrimestre 2023

Trabajo Práctico Integrador

Mosquera Sebastián	smosquera@fi.uba.ar	106744
Antipasti Lautaro Holwer	lantipasti@fi.ub.ar	106279

Índice

1. Objetivo del proyecto

El objetivo principal de este informe es documentar el proceso de desarrollo de código en C para manejar sensores y un controlador de motores puente h L293D mediante timers utilizando un microprocesador AVR ATmega328P. Se busca comprender en detalle el funcionamiento del L293D y su respectiva configuración relacionada al timer en modo PWM y CTC en complemento con el manejo de los sensores. Además, se incluirán diagramas de bloques, diagramas de de flujo y esquemáticos para ilustrar de manera clara y concisa la estructura del firmware.

2. Proyecto

Para este proyecto final integrador tenemos que desarrollar un código para que el vehículo construido a partir de los sensores, el shield de motores y los motores pueda seguir una linea. Para ello primero debemos comprender al detalle el funcionamiento del controlador de motores el cual esta formado principalmente por:

- 1 74HC595N (Registro de desplazamiento): Ubicado en el centro, permite la comunicación serial a paralelo entre el microcontrolador y los drivers de los motores, disminuyendo la cantidad de pines de salida utilizados.
- 2 L293D (Puentes H): Ubicados en los costados, están formados por transistores que dependiendo su configuración determinan el funcionamiento de los motores. Compuestos por amplificadores y controladores de corriente que permiten al chip a manejar corrientes más altas de las que podría entregar el microcontrolador.

Cada Circuito Integrado L293D tiene 4 entradas digitales que controlan 2 motores de continua bidireccionalmente. Las entradas se configuran de a pares y cada par controla un motor.

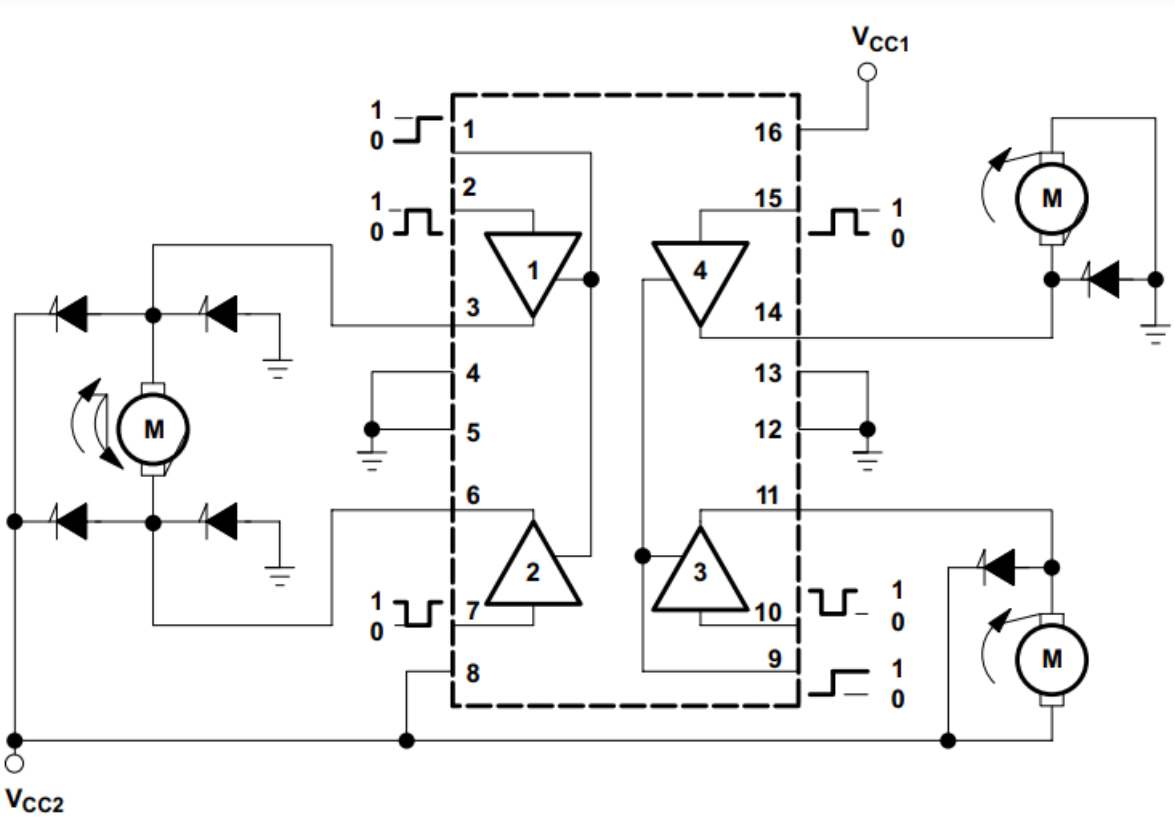

Figura 1: Diagrama en bloques funcional del L293D

Observando la figura, vemos por ejemplo que si el pin 2 está seteado en 1 (estado alto) y el pin 7 está en 0 (estado bajo) el motor de la izquierda se enciende.

Si estas entradas estuvieran invertidas el motor también se encendería pero rotaría en el sentido contrario. Por su parte, si ambas entradas se establecen en 0 el motor se apaga.

Si quisieramos controlar los 4 motores simultaneamente deberíamos conectar 8 pines a las salidas del microcontrolador.

Sin embargo, los pines son un recurso escaso y por ello se utiliza el integrado 74HCT595N que actúa como registro de desplazamiento.

Este integrado nos permite controlar los 8 pines de los drivers de los motores a través de únicamente 4 pines de entrada al chip: **DS** (DIR_SER), **SH_CP** (DIR_CLOCK), **ST_CP** (DIR_LATCH), **$\overline{\text{OE}}$** (DIR_EN).

- **DS**: Datos de entrada serial
- **SH_CP**: Entrada de clock para registro de desplazamiento
- **ST_CP**: Entrada de clock para registro de almacenamiento
- **$\overline{\text{OE}}$**: Pin de habilitación de salida

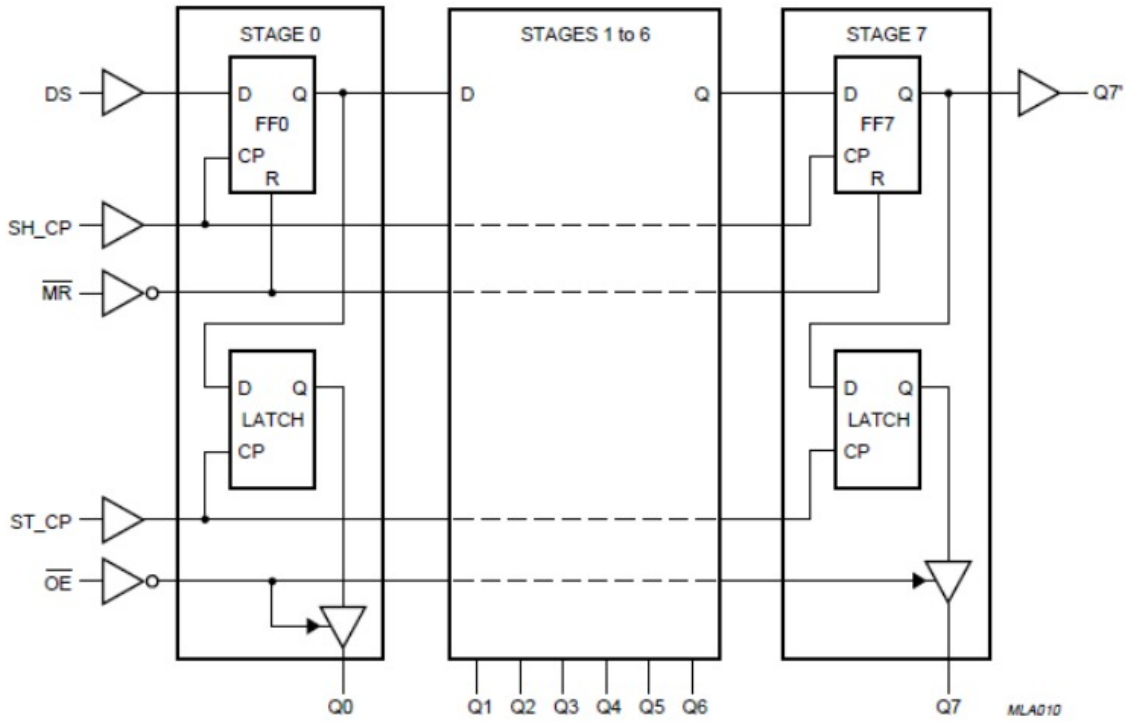

Figura 2: Diagrama Lógico del 74HC595

Analizando el datasheet del L293D observamos la información serial que debemos transmitirle al registro de desplazamiento para obtener la configuración del motor deseada.

Para ello, desarrollamos la función (config_motor), a la cual le pasamos el número correspondiente y esta mediante el uso de los pines de salida: B0, B4, D4, D7; maneja las 4 entradas al 74HC595, y este a través de las salidas Q0, Q1, Q2, Q3, Q4, Q5, Q6 y Q7 controla los drivers de los motores.

A la hora de configurar las macros, encontramos discrepancias entre lo mencionado en la tablas de datos y las verdaderas salidas de los pines. Para arreglar esto tuvimos que testearlas con un multímetro e ir determinándolas mediante prueba y error.

2.1. Esquemático

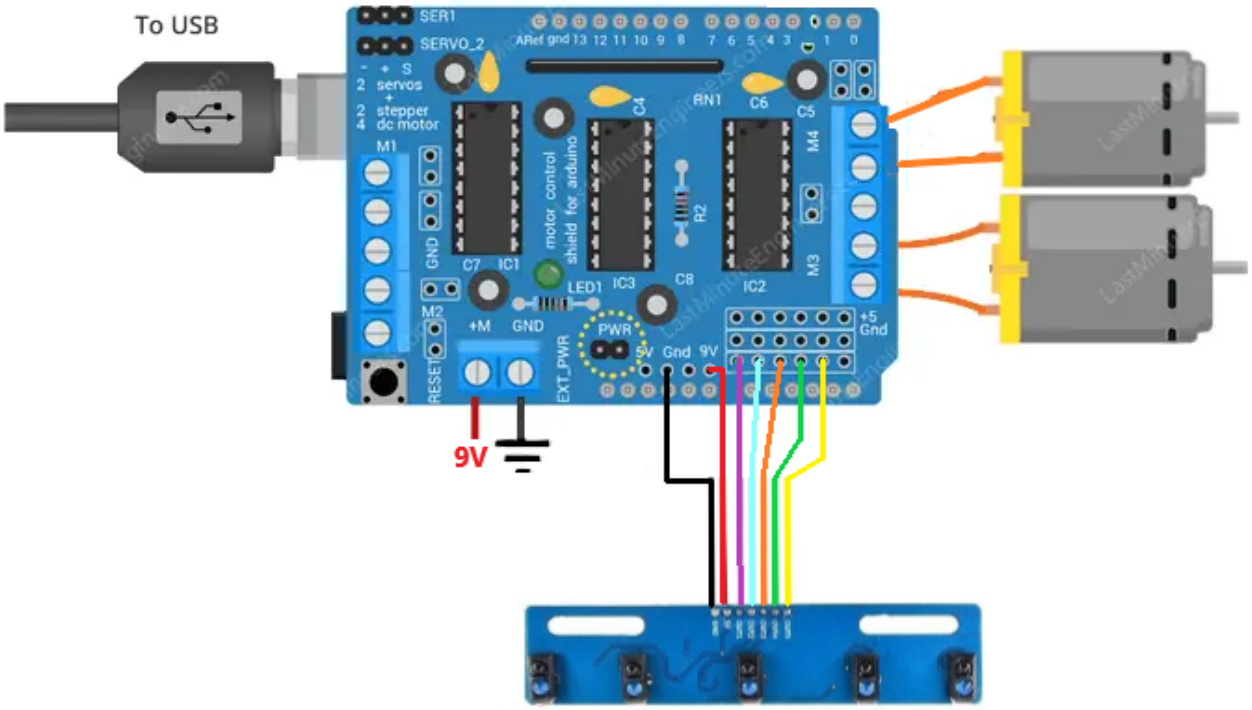

Figura 3: Esquemático conexión de 2 motores cc y sensores

2.2. Diagrama de bloques

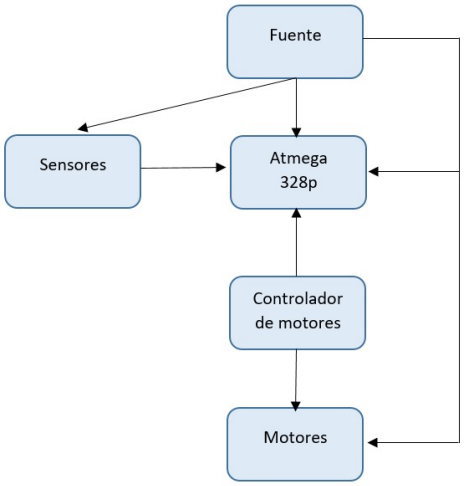

Figura 4: Diagrama de bloque

2.3. Diagrama de flujo

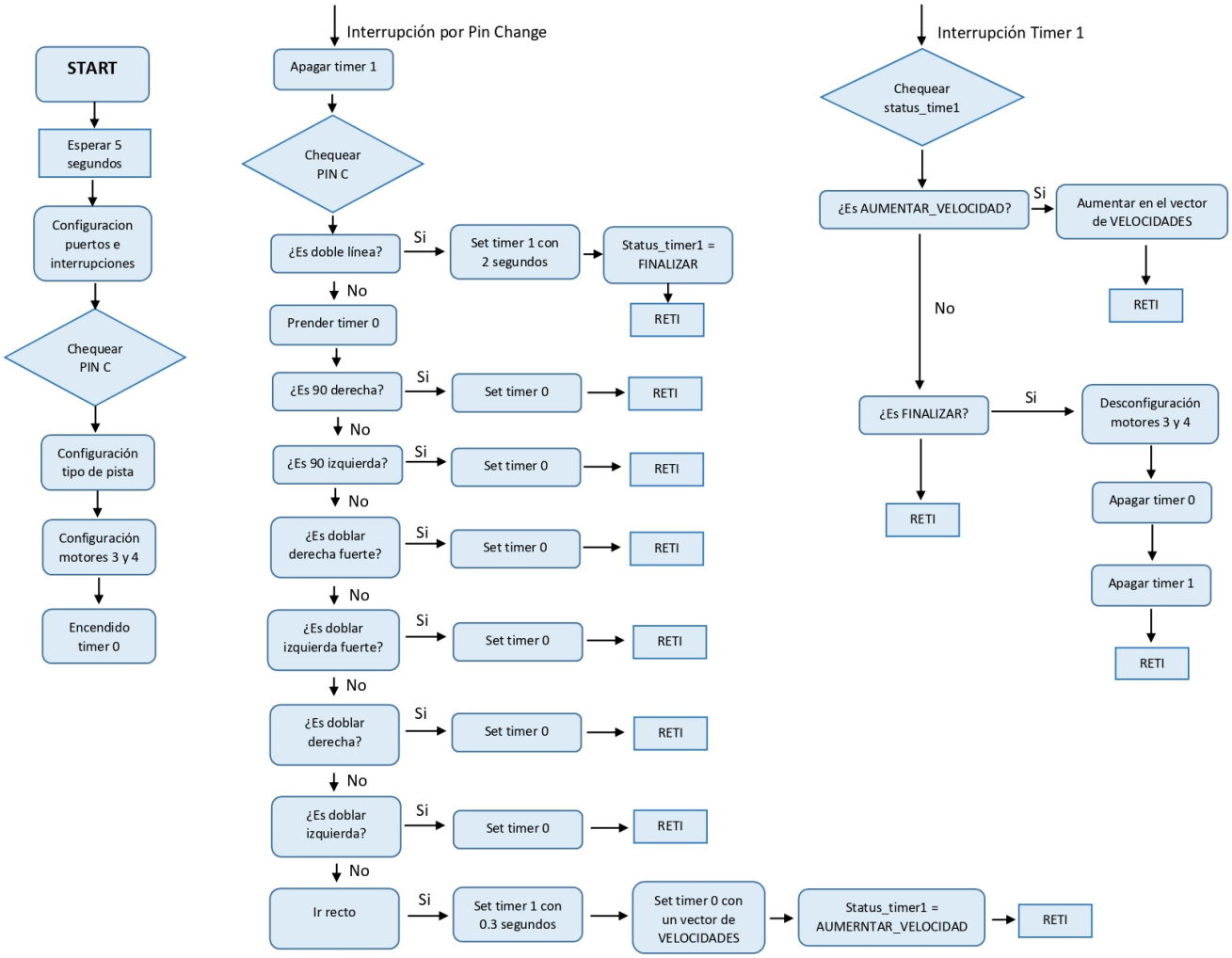

Figura 5: Diagrama de flujo

2.4. Código

```
#include <avr/io.h>
#include <avr/interrupt.h>
#include <util/delay.h>
#include <stdio.h>

const uint8_t M3_adelante = 0x04;// OCOA0
const uint8_t M3_atras = 0x01;
const uint8_t M4_adelante = 0x02;// OCOB0
const uint8_t M4_atras = 0x80;

const uint8_t velocidades[]={170,170,180,180};
volatile uint8_t velocidad;

volatile uint8_t status_timer1;
const uint8_t AUMENTAR_VELOCIDAD = 1;
const uint8_t FINALIZAR = 2;

volatile uint8_t tipo_de_linea;
const uint8_t NEGRA = 0;
```

```
20  const uint8_t BLANCA = 1;
21
22  volatile uint8_t lectura_pinc;
23
24  // CONFIGURACIONES
25  void configurar_puertos(void){
26          DDRB = 0xFF; // Todo B como salida
27          DDRD = 0xFF; // Todo D como salida
28          DDRC = 0x01;//  Todo C como entrada, Sensores
29  }
30
31  void configurar_interrupciones(void){
32          PCICR = (1 << PCIE1);
33          PCMSK1 = (1 << PCINT9) | (1 << PCINT10) | (1 << PCINT11) | (1 << PCINT12) | (1 << PCINT13) ;//
            ↪  A1, A2, A3, A4 y A5 ; Sensores
34          TIMSK1 = (1 << OCIE1A);
35
36          sei();
37  }
38
39  void config_motor(uint8_t motor){
40          PORTD &= 0x7F; // D7 en 0
41          PORTB &= 0xEF;
42          for(uint8_t i = 0; i < 8; i++){
43                  PORTD &= 0xEF;
44                  if((motor & (1 << i)) == 0){
45                          PORTB &= 0xFE;
46                          } else{
47                          PORTB |= 0x01;
48                  }
49                  PORTD |= 0x10;
50          }
51          PORTB |= 0x10;
52  }
53
54
55  // FUNCIONES DE TIMER
56  void apagar_timer0(void){
57          TCCR0A = 0;
58          TCCR0B = 0; // Cs en 0
59  }
60
61  void prender_timer0(void){
62          TCCR0A = (1 << COM0A1) | (1 << COM0B1) | (1 << WGM01) | (1 << WGM00); //Fast PWM ; Clear
            ↪  OC0A/OC0B on compare match, set OC0A at BOTTOM, (non-inverting mode);
63          TCCR0B = (1 << CS00);
64  }
65
66  void apagar_timer1(void){
67          TCCR1B = 0;
68  }
69
70  void prender_timer1(uint8_t tiempo_timer1){ // Prendemos un timer segun la cantidad de tiempo que nos
    ↪  pasan como parametro
71          TCCR1A = 0;
72          TCCR1B = (1 << WGM12) | (1 << CS12) | (1 << CS10);
73
74          if(tiempo_timer1 == 2) { // 2seg
75                  OCR1A = 31250;
```

```
76                  status_timer1 = FINALIZAR;
77          }
78
79          if(tiempo_timer1 == 1) { // 1seg
80                  OCR1A = 15625;
81                  status_timer1 = AUMENTAR_VELOCIDAD;
82          }
83
84          if(tiempo_timer1 == 3) { //0.3seg
85                  OCR1A = 5200;
86                  status_timer1 = AUMENTAR_VELOCIDAD;
87          }
88  }
89
90  // FUNCIONES DE VELOCIDAD
91  void doblar_der(void) {
92          OCR0A = 170;
93          OCR0B = 135;
94          return;
95  }
96
97  void doblar_izq(void) {
98          OCR0A = 135;
99          OCR0B = 170;
100         return;
101 }
102
103 void doblar_der_f(void) {
104         OCR0A = 180;
105         OCR0B = 120;
106         return;
107 }
108
109 void doblar_izq_f(void) {
110         OCR0A = 120;
111         OCR0B = 180;
112         return;
113 }
114
115 void ir_recto(uint8_t velocidad) {
116         prender_timer1(3);
117         OCR0A = velocidad;
118         OCR0B = velocidad;
119         return;
120 }
121
122 void doblar_90_der(void) {
123         OCR0A = 185;
124         OCR0B = 0;
125         return;
126 }
127
128 void doblar_90_izq(void) {
129         OCR0A = 0;
130         OCR0B = 185;
131         return;
132 }
133
134 // INTERRUPCIONES
```

```
135 ISR(PCINT1_vect) {
136         _delay_ms(50);
137 
138         if(tipo_de_linea == BLANCA) {
139                 // invertir PINC
140                 lectura_pinc = PINC ^ 0x3E;
141         }
142         if(tipo_de_linea == NEGRA) {
143                 lectura_pinc = PINC;
144         }
145 
146         apagar_timer1();
147 
148         if (lectura_pinc == 0x3E) {
149                 prender_timer1(2);
150                 return;
151         }
152 
153         prender_timer0();
154 
155         // 90 derecha
156         if (lectura_pinc == 0x30) {
157                 doblar_90_der();
158                 return;
159         }
160 
161         // 90 izquierda
162         if (lectura_pinc == 0x06) {
163                 doblar_90_izq();
164                 return;
165         }
166 
167         // Doblar derecha fuerte
168         if (lectura_pinc == 0x32) {
169                 doblar_der_f();
170                 return;
171         }
172 
173         // Doblar izquierda fuerte
174         if (lectura_pinc == 0x26) {
175                 doblar_izq_f();
176                 return;
177         }
178 
179         // Doblar derecha
180         if (lectura_pinc == 0x3A){
181                 doblar_der();
182                 return;
183         }
184 
185         // Doblar izquierda
186         if (lectura_pinc == 0x2E) {
187                 doblar_izq();
188                 return;
189         }
190 
191         // Ir recto
192         if (lectura_pinc == 0x36) {
193                 velocidad = 0;
```

```
                ir_recto(velocidades[0]);
                return;
        }
}

ISR(TIMER1_COMPA_vect) {
        if(status_timer1 == AUMENTAR_VELOCIDAD) {
                if(velocidad == 3) {
                        return;
                }
                velocidad += 1;
                ir_recto(velocidades[velocidad]);
                return;
        }
        if(status_timer1 == FINALIZAR) {
                config_motor(0x00);
                apagar_timer0();
                apagar_timer1();
                return;
        }
        return;
}

int main(void){
        _delay_ms(50000); // Delay inicial de 5 segundos
        configurar_puertos();

        if( (PINC & 0x22) == 0x22) { // Deteccion del tipo de pista
                // Negra
                tipo_de_linea = NEGRA;
        }
        else {
                // Blanco
                tipo_de_linea = BLANCA;
        }

        configurar_interrupciones();
        velocidad = 0;
        config_motor(0x06);
        prender_timer0();
        OCR0A = 120;
        OCR0B = 120;
        while(1){
        }
        return 0;
}
```

3. Listado de componentes

- 5 Sensores infrarrojos
- 2 Moto reductores de 6 a 9 V
- 1 Fuente de alimentación 9V 1A
- 1 Controlador de motores L293D
- 1 Móvil con ruedas de auto seguidor de línea
- 1 Microcontrolador ATmega328p

- 1 Programador de microcontroladores avr
- 10 Cables Dupont macho macho
- 1 Placa experimental

4. Conclusiones

Para el desarrollo del proyecto fue necesario leer las distintas hojas de datos y documentación de cada elemento. Si bien, estas nos brindaron información esencial sobre la función de cada integrado, y en particular de cada uno de sus pines, muchas veces no coincidían los pines proporcionados por el esquemático oficial con el nuestro. Es por ello, que debimos utilizar un tester y corroborar pin a pin para encontrar las discrepancias.

Un problema que encontramos a la hora de calibrar la velocidad en que giraban las ruedas fue que inicialmente no tuvimos una buena fuente de alimentación. Comenzamos utilizando una batería de 9V, pero esta no entregaba la potencia necesaria. Luego, una vez en clase con ayuda del transformador y probando el auto en una pista, realizamos las correcciones en los valores de velocidad necesarias.

Con respecto al código no se presentaron grandes dificultades ya que implementamos una lógica sencilla en la que se chequea cada uno de los principales estados en los que pueden estar los sensores y en base a eso modificamos las velocidades de las ruedas con valores fijos. Como necesitamos utilizar el Timer1 en más de una ocasión implementamos una variable llamada **status_timer1**que puede tomar los valores: **FINALIZAR, AUMENTAR_VELOCIDAD** donde **FINALIZAR** se utiliza para apagar movil en caso de que no detecte linea por 2 segundos y **AUMENTAR_VELOCIDAD** se usa para incrementar la velocidad del movil en linea recta cada 0.3 segundos.

Se realizaron pruebas del auto con distintas pistas en clase, y demostró cumplir con un funcionamiento adecuado y acorde a los requerimientos.