

Vehículo a Control Remoto

Informe de Avance

Taller de Proyecto I | Ingeniería en Computación

Arreche, Cristian
Blasco, Federico
Borini, Ángel
Paradiso, Martín

25 de febrero de 2020

Facultad de Ingeniería
Universidad Nacional de La Plata



Índice general

1. Introducción	3
2. Objetivos	4
3. Requerimientos	6
3.1. Requerimientos Funcionales	6
3.1.1. Requerimientos de Software	6
3.1.2. Requerimientos de Hardware	6
3.2. Requerimientos No Funcionales	7
4. Diseño de Hardware	8
4.1. Diagrama en Bloques	8
4.1.1. Recepción Inalámbrica	8
4.1.2. Controlador de Motores	10
4.1.3. Detector de Obstáculos	12
4.1.4. Indicadores de Estado	13
4.2. Cálculos de Corriente	14
4.3. Diseño de PCB	15
4.3.1. Ancho de pistas	15
4.3.2. Consideraciones para cada dispositivo	16
4.3.3. BOM simplificado	16
5. Diseño de Software	18
5.1. Pseudocódigo de los Módulos	18
5.1.1. Leer Bluetooth	18
5.1.2. Detectar Colisiones	19
5.1.3. Actualizar Motores	19
5.1.4. Software Android	20
5.2. Interfaz de Usuario	22
5.2.1. Pasos a seguir para la configuración del sistema	22
5.2.2. Utilización del Vehículo	23
5.3. Librerías de Software	23
5.4. Ensayos	24
5.4.1. Motores y L293D	24
5.4.2. PWM	25
6. Ensayos y Mediciones	26
6.1. Motores y L293D	26
6.2. Sensor HC-SR04	27
6.3. Control remoto	27
6.4. Buzzer	29
6.5. Baterías de litio	29
6.6. Fuente step down	30

7. Conclusiones	31
7.1. Cumplimiento de los objetivos	31
7.2. Cumplimiento de Requerimientos	32
7.2.1. Requerimientos Funcionales	32
7.2.2. Requerimientos No Funcionales	35
7.2.3. Evaluación	36
7.3. Actividad de los integrantes	36
7.4. Presupuesto	38
8. Bibliografía	39
9. Anexo	40
9.1. Esquemático Completo	40
9.2. Imágenes de Capas	41
9.3. Imágenes en 3D	42
9.4. Información general	43
9.5. Aplicaciones Externas	43
9.6. Datasheets	43
9.7. BOM	44
9.8. Imágenes de los Componentes	45
9.8.1. mini360	45
9.9. Cronograma	46
9.10. División de Tareas del Grupo	47
9.11. Manual de Usuario	48
9.12. Imágenes del Proyecto	49
9.12.1. Desarrollo	49
9.12.2. Proyecto terminado	56

1. Introducción

Una de las principales aplicaciones que se le ha dado a la electrotecnia es el control de motores. En el ámbito de la electrónica se hace uso de pequeños motores con poca potencia que funcionan con corriente continua, como por ejemplo para mover autos a escala.

En los últimos años, con la aparición de plataformas de desarrollo como Arduino, se han desarrollado autos a control remoto artesanales; que proveen una gran diversidad de resultados debido a la plataforma abierta y variedad de componentes electrónicos.

La motivación del proyecto es obtener un producto que cuente con cierto nivel de complejidad y, al mismo tiempo, que sea visualmente gratificante. Consideramos que el control de un vehículo a control remoto conlleva una parte importante de investigación y desarrollo; y brinda un resultado que es tangible.

El proyecto es un vehículo móvil, equipado este con motores eléctricos que reaccionan dependiendo de las señales enviadas desde un mando inalámbrico del que depende el movimiento del automóvil. Además, colocaremos una serie de sensores para agregar mayor funcionalidad al proyecto reaccionando ante distintos eventos.

2. Objetivos

El objetivo principal del proyecto es el diseño de un vehículo a escala, que pueda ser controlado remotamente por los usuarios.

El proyecto se divide en 4 partes principales:

- Estructura física del vehículo
- Control de motores
- Control remoto
- Medición de sensores y respuesta en consecuencia

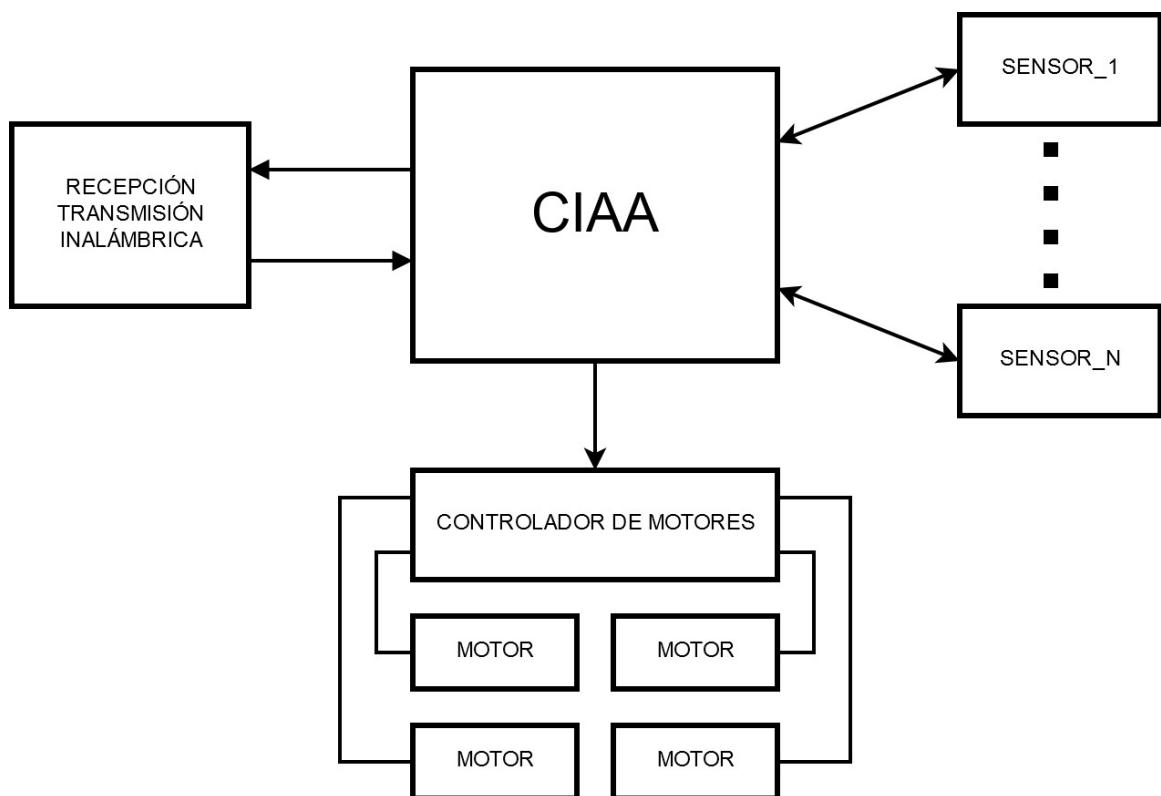


Figura 1. Diagrama en bloques del sistema

Estructura física del vehículo El desarrollo del vehículo implica tener una estructura sobre la cual se puedan montar los motores con sus respectivos controladores, sensores, la EDU-CIAA y la/s baterías necesarias para alimentar el sistema. La estructura deberá ser lo suficientemente rígida para soportar el peso del hardware mencionado anteriormente, y al mismo tiempo debe ser liviana para que los motores puedan transportarla sin la necesidad de una potencia excesiva.

Control de Motores Los motores deben permitir el movimiento en ambos sentidos, de manera de rotar el vehículo sin utilizar un complejo sistema de servos y bisagras. Los controladores se encuentran sobre el poncho, con salidas para cada motor, haciendo el sistema independiente de los motores y estructura del vehículo, como se puede observar en la *Figura 1*.

Control Remoto El objetivo es controlar el vehículo de manera remota a través de protocolo bluetooth. Debe permitir manejar el vehículo desde una distancia considerable (entre 10m y 20m), permitiendo realizar todos los movimientos: avanzar, retroceder, y girar hacia izquierda y derecha. También es deseable disponer de comandos auxiliares para agregar funcionalidad adicional, como por ejemplo, encender/apagar luces manualmente.

Medición de sensores y respuesta en consecuencia Como objetivos secundarios del proyecto, consideramos la colocación de algunos sensores:

- Sensor LDR, el cual ante la ausencia de luz se encienden luces delanteras y traseras del vehículo
- Sensores de proximidad por infrarrojo, los cuales controlan el vehículo ante los obstáculos con los que se encuentre

3. Requerimientos

Aquellos requerimientos indicados con **OPT** son completamente opcionales.

3.1. Requerimientos Funcionales

3.1.1. Requerimientos de Software

Control de Motores

1. El sistema debe accionar 4 motores individualmente, permitiendo girar en ambas direcciones
2. El sistema debe accionar motores en conjunto, permitiendo avanzar, retroceder y girar
3. El sistema debe permitir el bloqueo de movimiento en una dirección (por ejemplo debido a un obstáculo detectado por el sensor de proximidad)
4. **OPT** El sistema debe indicar mediante una serie de LEDs el estado de movimiento del vehículo

Control Remoto

1. El sistema debe establecer una conexión bluetooth con el sistema de control remoto
2. El sistema debe leer las indicaciones del control remoto y ejecutar las acciones correspondientes
3. El sistema debe indicar mediante un LED el estado de la conexión bluetooth

OPT Medición Sensores

1. Ante ausencia de luz externa, se deben encender las luces del vehículo
2. Ante la presencia de obstáculos, se debe frenar el desplazamiento en esa dirección

3.1.2. Requerimientos de Hardware

Control de Motores

1. El vehículo debe utilizar puentes H para motores de corriente continua
2. **OPT** El vehículo debe Indicar con un LED para cada motor si este está en movimiento

Baterías y cargador

1. El vehículo debe utilizar baterías de Litio para lograr la autonomía del vehículo
2. El vehículo debe indicar mediante un LED cuando haya baja tensión
3. El vehículo debe permitir cargar las baterías mediante un puerto USB

Componentes

1. Se debe contar con una llave de encendido/apagado y un LED que determine su estado
2. **OPT** El sistema debe poseer un buzzer para indicar eventos o fallas

3.2. Requerimientos No Funcionales

- Utilización de la placa de desarrollo EDU-CIAA
- Estructura sólida y liviana, donde se colocan los motores y la EDU-CIAA
- El conexionado a los motores debe ser a través de borneras para poder cambiar la estructura.
- Programación en lenguaje C
- Desarrollo de PCB en formato de poncho
- Mecanismo de control simple
- El sistema debe manejar respuestas en tiempo real.
- Fecha de finalización y entrega del proyecto el día 16/12

4. Diseño de Hardware

4.1. Diagrama en Bloques

4.1.1. Recepción Inalámbrica

Para implementar el manejo por control remoto se utiliza el protocolo Bluetooth, más específicamente el módulo HM-10, que funciona como una comunicación serie inalámbrica. Del lado del usuario se utiliza la aplicación de Android "BLEJoystick", disponible en Google Play Store. También se cuenta con un LED para indicar la correcta vinculación entre la app y el módulo bluetooth.

Componentes a Utilizar

Componente	Código	Valor	Cantidad	Función	Protocolo
Rx/Tx BLE LED	HM-10 -	- Rojo 3mm	1 1	Recepción Informar estado de la conexión Bluetooth	UART 8N1 Digital

Rx/Tx BLE

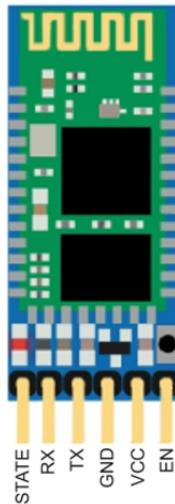


Figura 4.1: HM-10

El módulo Bluetooth a utilizar es el HM-10 provisto por la cátedra, el cual ya fue utilizado en asignaturas anteriores por lo cual se conocen sus características y su funcionamiento. Además, la sAPI de la CIAA provee una librería para comunicación Bluetooth, facilitando su programación.

Respecto a las características de Hardware, el HM-10 funciona a 3.3v, por lo que no hay que realizar ninguna adaptación para su funcionamiento.

LED

Para la alimentación del LED se utiliza el puerto GPIO1, realizando los siguientes cálculos se obtiene el valor de la resistencia que necesita para su conexión::

$$R = \frac{V - LED_V}{LED_I} = 420\Omega$$

$V = 3,3v$; $LED_V = 1,2v$; $LED_I = 5mA$;

Conexionado

HM-10 El módulo Bluetooth se comunica a través de un protocolo serie 8N1, a 9600 bps, y se configura a través de comandos AT. Las librerías de la CIAA ya proveen funciones para establecer una comunicación con un módulo HM-10.

LED Se conecta de manera directa a una salida digital del integrado, teniendo en cuenta la resistencia mencionada.

Esquemático

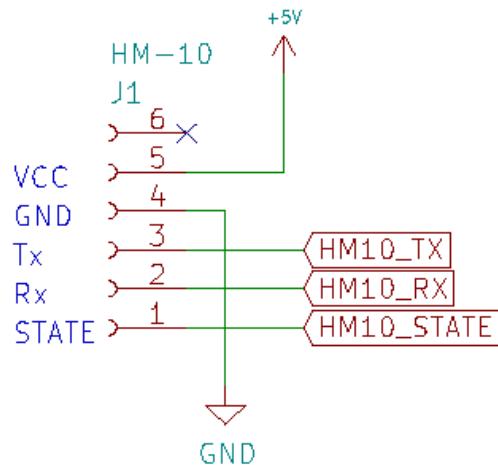


Figura 4.2: Conexionado del Módulo Bluetooth HM-10

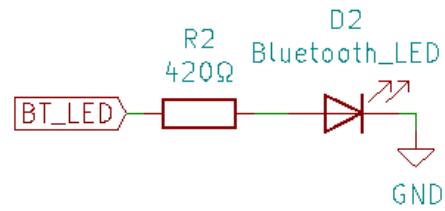


Figura 4.3: Conexión del LED Bluetooth

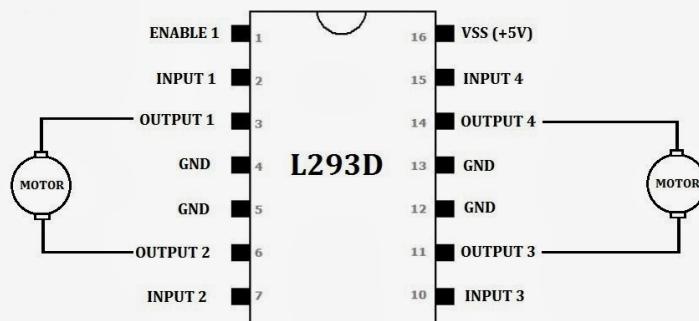
4.1.2. Controlador de Motores

Para el control de motores se utiliza el circuito integrado L293D, en nuestro caso utilizamos la versión de STMicroelectronics, el cual provee dos puente H completos, permitiendo así controlar hasta dos motores en ambos sentidos.

Componentes a Utilizar

Componente	Código	Valor	Cantidad	Función	Protocolo
Puente H	L293D	-	2	Controlar motores	Analógico/PWM

Puente H



Dual DC Motor Controller

Figura 4.4: L293D

Al investigar sobre circuitos que provean puentes H para controlar motores de continua, rápidamente encontramos el L293D. Este tiene 4 canales, cada uno formando

medio puente H. Dado que para girar motores en ambos sentidos se requiere un puente completo, se pueden controlar dos motores con un solo L293D.

En términos de conexionado provee un *enable* por cada par de canales, alimentación independiente para lógica y motores, y 6 pines centrales para disipación de calor.

Respecto a la alimentación, requiere entre 4.5v y 30v. Dado el elevado consumo es imposible alimentarlo directamente a través de la CIAA, por lo que se conecta de manera directa a la fuente de alimentación.

Soporta corrientes pico de 1.2A, por lo que es correcto colocar un disipador en caso de que los motores consuman mucha corriente.

Conexionado

L293D Las entradas de los L293D se controlan mediante PWM, permitiendo controlar la velocidad de cada motor de manera independiente. Esto requiere utilizar 8 PWM, dos para cada motor.

Esquemático

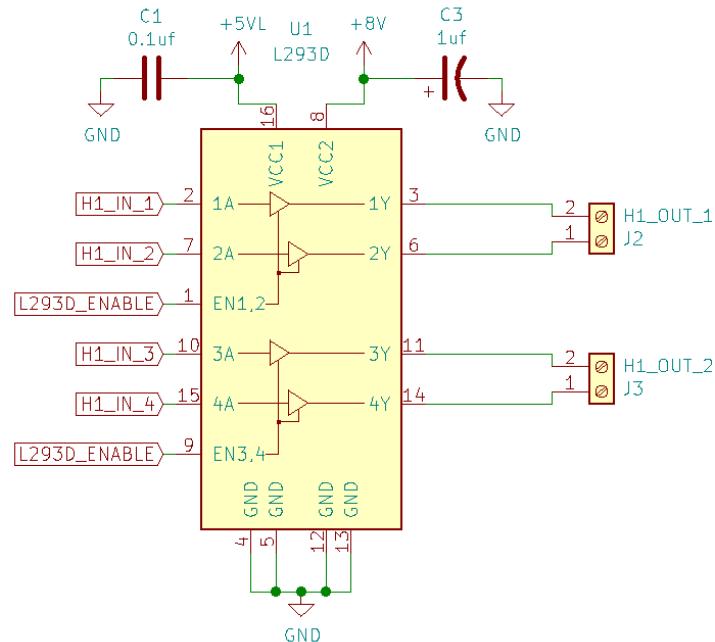


Figura 4.5: Conexionado de un L293D

Capacitores Se colocan capacitores en la alimentación según indica la hoja de datos del integrado L293D de Texas Instruments.

4.1.3. Detector de Obstáculos

Para detectar obstáculos se utiliza el sensor de proximidad de Arduino HC-SR04, que permite captar objetos entre 2cm y 400cm, con un ángulo de 15° . Estos rangos resultan ideales para detectar obstáculos y actuar en consecuencia en un vehículo a control remoto.

Componentes a Utilizar

Componente	Código	Valor	Cantidad	Función	Protocolo
Sensor Proximidad	HC-SR04	-	1	Medir distancia a obstáculos	Digital
Buzzer	-	-	1	Indicar mediante sonido	Digital

Sensor de Proximidad

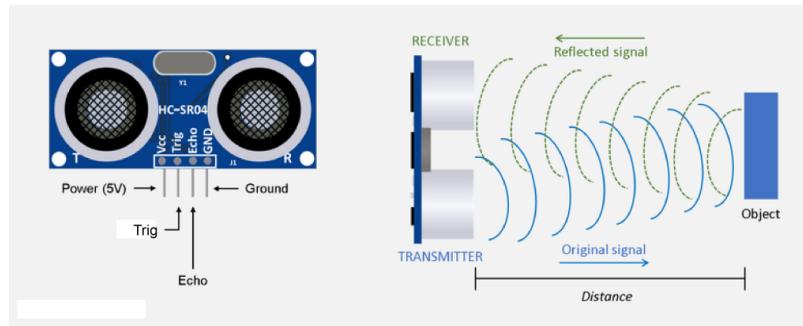


Figura 4.6: HC-SR04

El HC-SR04 es un sensor ultrasónico que permite calcular la distancia de los objetos que se encuentran en frente a él, con un ángulo de visión de 15° . Utiliza una onda sonora de 40kHz, inaudible para el oído humano, que se emite cuando se coloca un 1 en el pin *trigger*. Cuando el módulo detecta el rebote del pulso emitido, coloca un 1 en el pin *echo*. Midiendo el tiempo entre la emisión del pulso y la recepción del eco, puede determinarse la distancia al objeto.

Buzzer

Se utiliza como componente auxiliar para indicar la presencia de obstáculos mediante sonido.

Conexionado

HC-SR04 Dado que requiere medir con presición el tiempo transcurrido, es necesario utilizar un timer en modo *input capture* para lograr la mayor presición posible.

El *trigger* se generará con una salida GPIO común.

Buzzer Dado que el consumo del buzzer supera el máximo tolerado por los puertos GPIO de la CIAA, se utiliza un transistor para alimentar el buzzer y asegurar su correcto funcionamiento sin dañar ningún componente de hardware. Se utiliza el pin de GPIO7 como salida para el LED.

Esquemático

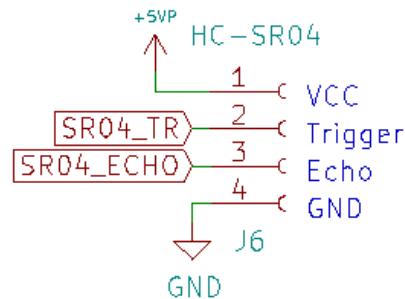


Figura 4.7: Conexionado del HC-SR04

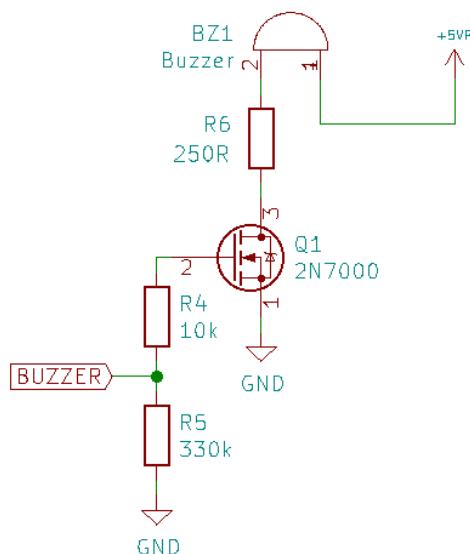


Figura 4.8: Conexionado del Buzzer

4.1.4. Indicadores de Estado

Se utiliza como componente auxiliar para indicar el estado del vehículo, un LED que indica si el vehículo está encendido.

Esquemático

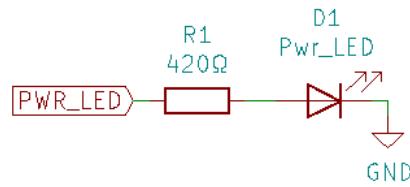


Figura 4.9: Conexionado del LED de encendido

4.2. Cálculos de Corriente

Subsistema	Tensión [V]	Corriente [mA]
Principal	5	200
UART2 (BT)	-	0.5
SCT * 3	-	0.16 *3
GPIO * 6	-	5 * 6
HM-10	5	10
L293D * 2	5	32 * 2
Motor * 4	8	100 * 4
<hr/>		
Total	5	280mA
Total	8	400mA

Como se puede observar, dado la variedad de dispositivos y componentes, se requieren dos valores de tensión diferentes. Además, al tratarse de un vehículo manejado inalámbricamente, es necesario el uso de baterías. Utilizando dos baterías de Litio en serie se obtiene una tensión de entrada de aproximadamente 8V, por lo tanto, es necesario utilizar un conversor DC-DC step-down para obtener tensiones de 5V, y el conversor interno de la EDU-CIAA para obtener tensión de 3.3V.

Para alimentar la EDU-CIAA se opta por utilizar el conector Molex de alimentación externa, utilizando uno igual en el poncho.

Esquemático

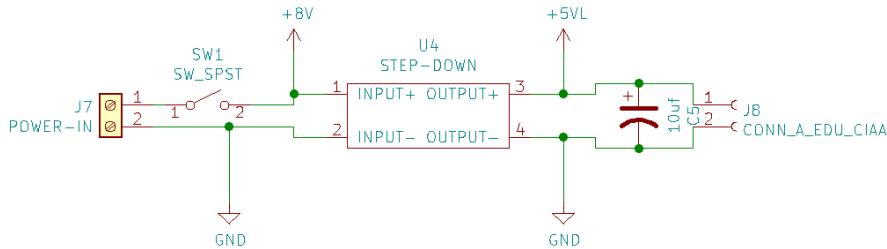


Figura 4.10: Conexionado de la alimentación

4.3. Diseño de PCB

Para el diseño de la PCB se tuvieron en cuenta las siguientes características:

- Los conectores para los motores se deben disponer de manera que representen la rueda que está accionando.
- Los integrados L293D deben alejarse del resto de los componentes, ya que manejan una elevada corriente, y mucha temperatura.
- Los integrados L293D deben tener un plano de tierra en sus pines centrales, tal como indica la hoja de datos, para poder disipar calor.
- El sensor ultrasonido HC-SR04 debe encontrarse en un extremo de la placa, de manera que la PCB pueda montarse con este sensor apuntando hacia adelante.
- El conector de baterías debe colocarse cerca de los extremos para no tener cables muy largos.
- El conector de alimentación para la EDU-CIAA debe encontrarse lo más cerca posible del conector en la EDU-CIAA.
- Tener en cuenta las recomendaciones de la cátedra respecto al ancho de pista, y distancia entre estas. Como así también el tamaño de los pads.

4.3.1. Ancho de pistas

Además, debe tenerse en cuenta que dado el elevado consumo de los motores, es necesario tener en cuenta el ancho de las pistas, para no sobrecalentarlas. Utilizando la calculadora incorporada en KiCAD, se ve que una pista de 0.6mm (lo mínimo recomendado por la cátedra) puede tolerar sin problemas hasta 1.6A, por lo qué para todas las aplicaciones de GPIO, PWM, y alimentación de periféricos no hay problemas.

Respecto a los L293D, están diseñados para soportar 300mA por canal, con una corriente máxima total de 1.2A, por lo qué, tanto las pistas de los motores como las de alimentación no requieren más que el ancho mínimo de 0.6mm. Sin embargo hay que tener en cuenta que el consumo de ambos combinados puede llegar a 2.4A, por lo que, la pista “compartida” de alimentación, si debe tener un ancho mayor, calculado en 1mm.

El consumo total de la placa se estipula en 3A, dado que la CIAA consume alrededor de 200mA, y los motores 2.4A en los casos más extremos. Los consumos se resumen en la siguiente tabla:

Pista	Consumo	Ancho real	Ancho a utilizar
Alimentación L293D	1.3A	0.4mm	0.6mm
Alimentación de 2 L293D	2.6A	1mm	1mm
8v (total)	3A	1.3mm	1.3mm
5v	0.4A	0.08mm	0.6mm

Nota: para los cálculos finales, se agregó un 10% de consumo por seguridad. Además, los cálculos para los L293D se realizan en base a las capacidades máximas de los mismos, y no con los motores que se disponen, que tienen un consumo mucho menor.

4.3.2. Consideraciones para cada dispositivo

Se trató de agrupar físicamente los componentes según su asociación lógica. Por ejemplo, las resistencias para un LED, cerca de ese LED.

L293D Además de los requisitos mencionados previamente para los puentes H, debe tenerse en cuenta que requieren de capacitores filtro en las entradas de alimentación, cerca de cada integrado.

4.3.3. BOM simplificado

A continuación se presenta una versión simplificada de la lista de materiales en un formato legible, para un BOM más detallado que incluya las footprints de KiCAD, ver Anexo.

Referencia	Valor	Componente
BZ1	-	Buzzer 12x9.5x7.6
C1	100nF	Capacitor cerámico
C2	100nF	Capacitor cerámico
C3	1uF	Capacitor electrolítico
C4	1uF	Capacitor electrolítico
C5	10uF	Capacitor electrolítico

Referencia	Valor	Componente
D1	LED 5mm	LED 5mm
D2	LED 5mm	LED 5mm
D3	LED 5mm	LED 5mm
J1	Header HM10	Tira de pin hembra x6
J2	-	Bornera Azul de 2
J3	-	Bornera Azul de 2
J4	-	Bornera Azul de 2
J5	-	Bornera Azul de 2
J6	Header HC-SR04	Tira de pin hembra x4
J7	-	Bornera Azul de 2
J8	-	Molex KK254 Macho 2 pines
Q1	2n7000	Transistor 2n7000
R1	420R	Resistencia carbón 1/4
R2	420R	Resistencia carbón 1/4
R3	420R	Resistencia carbón 1/4
R4	10k	Resistencia carbón 1/4
R5	330k	Resistencia carbón 1/4
R6	250R	Resistencia carbón 1/4
SW1	Switch SPST	Interruptor DIP 1 posición
U1	L293D	Puente H L293D
U2	L293D	Puente H L293D
U4	Mini360	Fuente Step-Down mini360
X	Pines	Tira de pines p/edu-ciaa
N/A	HM-10	HM-10
N/A	HC-SR04	HC-SR04

5. Diseño de Software

Se optó por utilizar una arquitectura Time-Triggered cooperativa, donde las tareas se corresponden con los subsistemas de Hardware:

- Control de motores
- Recepción inalámbrica
- Detector de obstáculos

Además de la interrupción de tiempo, propia de la arquitectura Time Triggered, se necesita una interrupción de tiempo independiente utilizada por el sensor de proximidad para medir con presición la distancia a la que se encuentran los obstáculos.

El psuedocódigo del programa principal se resume a lo siguiente:

```
interrupcion 50ms:  
    flag_tiempo = 1  
  
variables compartidas:  
    accion_vehiculo # Enumerativo que indica el desplazamiento que  
                    # se debe realizar  
  
programa principal:  
    inicializacion motores  
    inicializacion bluetooth  
    inicializacion detector de colisiones  
    prender led power  
  
super loop:  
    si flag_tiempo = 1:  
        leer bluetooth  
        detectar colisiones  
        actualizar motores  
        flag_tiempo = 0
```

5.1. Pseudocódigo de los Módulos

5.1.1. Leer Bluetooth

```
si hay mensajes nuevos:  
    leer el mensaje  
    si es una A o a:  
        motor = avance
```

```

    si es una C o C:
        motor = retrocede
    si es una D o d:
        motor = giro izquierda
    si es una B o b:
        motor = giro derecha
    si es un 0:
        motor = giro libre
    si es una F o f:
        motor = freno

```

5.1.2. Detectar Colisiones

El eco producido por el módulo se genera -aproximadamente- entre 60uS y 12mS. Creemos que no es necesario realizar las mediciones cada estos periodos tan cortos de tiempo, dado que, puede ser contraproducente para los tiempos del sistema considerando el objetivo del sensor. Por esto, el procesamiento del se realiza cada 250ms:

```

inicializar detector de colisiones:
    tick_colision = 0

detectar colisiones:
    tick_colision++
    si tick_colision = 5:
        tick_colision = 0
        tomar tiempo
        prender input capture
        enviar señal de comienzo de medicion

    si llego el echo:
        calcular distancia
        si hay un obstaculo cerca:
            bloquear movimiento hacia adelante

```

5.1.3. Actualizar Motores

Los motores se controlan en dos niveles: a bajo nivel se puede controlar cada motor de manera independiente, indicando la dirección de giro. Una capa más “alta” permite controlar el vehículo más facilmente, brindando funciones para indicar como se debe mover el vehículo.

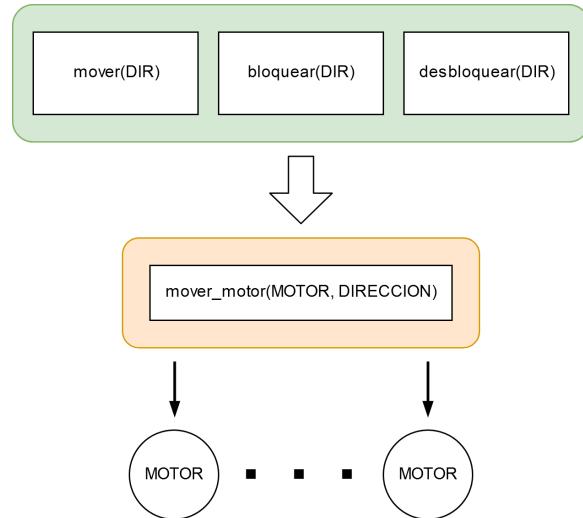


Figura 5.1: Capas de los Motores

Por lo que el pseudocódigo se reduce a lo siguiente

```

inicializacion de motores:
    setear movimiento en libre

actualizacion de motores:
    ver accion_vehiculo
    mover el vehiculo en la direccion solicitada

    # A modo de ejemplo se explica como mover en una sola direccion
    # si movimiento es hacia adelante:
        motor trasero izquierdo girar horario
        motor trasero derecho girar antihorario
        motor delantero izquierdo girar horario
        motor delantero derecho girar horario
    ...

```

A bajo nivel se controla cada motor individualmente:

```

si movimiento es horario:
    colocar canal 1 en alto
    colocar canal 0 en bajo
si movimiento es anti horario:
    colocar canal 0 en bajo
    colocar canal 1 en alto

```

5.1.4. Software Android

BLEJoystick BLEJoystick es la aplicación a utilizar para el control inalámbrico del vehículo, está disponible para múltiples dispositivos android y se puede descargar

de manera gratuita desde Google Play Store. Esta aplicación se conecta al módulo bluetooth HM-10 mediante el protocolo bluetooth low energy 4.0 (BLE). Cuenta con 8 botones, donde el funcionamiento de cada uno es el siguiente:

- Cuando se presiona por primera vez, envía una letra en mayúscula.
- Mientras siga presionado el mismo botón, envía la misma letra en minúscula de manera constante.
- Cuando se suelta el botón, envía un 0.

A continuación se describe cada uno de los botones y su letra correspondiente:

Botón	Letra inicial	Letra constante
Arriba	A	a
Derecha	B	b
Abajo	C	c
Izquierda	D	d
Triángulo	E	e
Círculo	F	f
Equis	G	g
Cuadrado	H	h

5.2. Interfaz de Usuario

5.2.1. Pasos a seguir para la configuración del sistema

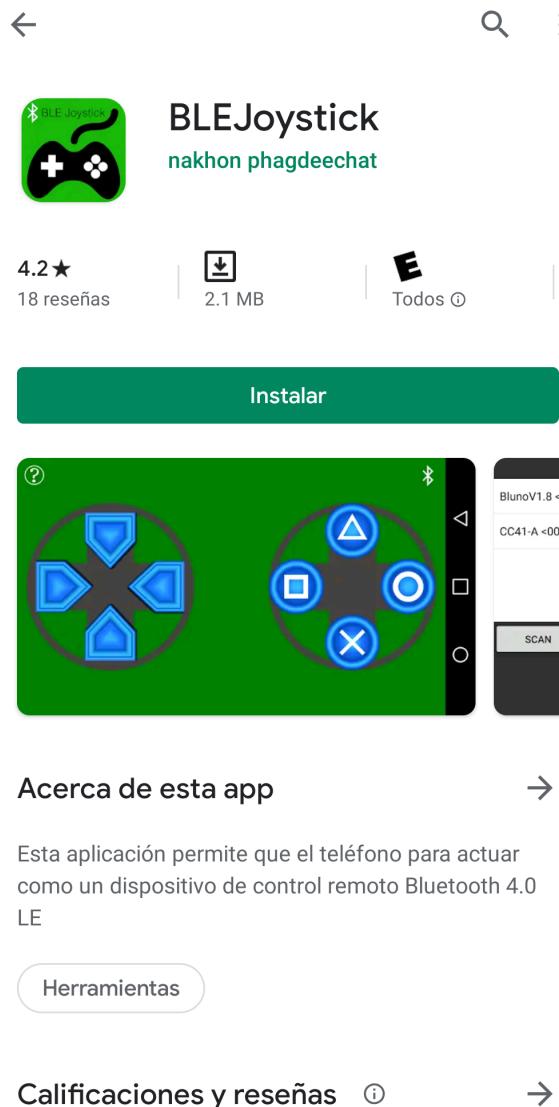


Figura 5.2: Aplicación en el Play Store

1. Ingresar a Google Play Store
2. Descargar la aplicación BLEJoystick
3. Iniciar la aplicación.
4. Pulsar el ícono de vinculación bluetooth y seleccionar *vehiculo*.

5. Corroborar la correcta vinculación mediante el LED indicador.
6. Sistema listo para usar.

5.2.2. Utilización del Vehículo

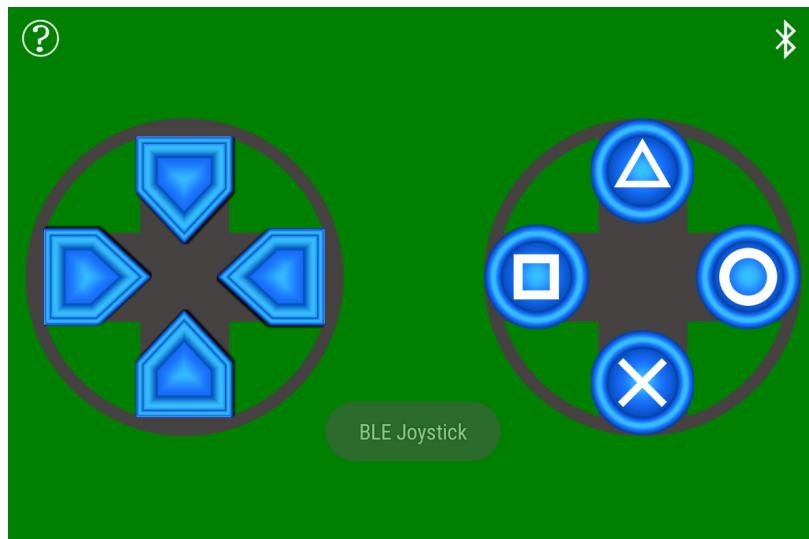


Figura 5.3: La aplicación BLEJoystick

A continuación se detalla y explica el funcionamiento de cada uno de los botones disponibles en la aplicación

Botón	Función
Arriba	Desplazar hacia adelante
Derecha	Girar hacia la derecha
Abajo	Desplazar hacia abajo
Izquierda	Girar hacia la izquierda
Círculo	Frenar

5.3. Librerías de Software

sAPI Dada la extensión de la librería provista por la CIAA, solo se requiere usar esta. Más específicamente se utilizaran las librerías `sapi_pwm.h`, `sapi_gpio.h` y `sapi_timer.h`.

5.4. Ensayos

5.4.1. Motores y L293D

Realizamos una prueba de los motores con los L293D, para esto realizamos un prototipo de la estructura del vehículo con el objetivo de observar el correcto funcionamiento tanto de los motores como del chip L293D y verificar si la potencia de los motores será suficiente para mover el vehículo con todos sus componentes a una velocidad considerable.

El ensayo se realizó con una placa de Arduino con un shield para controlar motores, donde todos los motores fueron configurados con la máxima velocidad posible, generando el siguiente el código de prueba:

```
#include <AFMotor.h>

AF_DCMotor motor1(1);    // Declara los motores a utilizar
AF_DCMotor motor2(2);
AF_DCMotor motor3(3);
AF_DCMotor motor4(4);

void setup() {
    Serial.begin(9600);      // Inicializa la terminal serie a
                           // 9600 bps
    Serial.println("Prueba de motores");

    motor1.run(RELEASE);    // Desbloquea el motor
    motor1.run(FORWARD);   // Mueve para adelante
    motor1.setSpeed(255);   // Velocidad entre 0 y 255,
                           // configurada al maximo

    motor2.run(RELEASE);
    motor2.run(FORWARD);
    motor2.setSpeed(255);

    motor3.run(RELEASE);
    motor3.run(FORWARD);
    motor3.setSpeed(255);

    motor4.run(RELEASE);
    motor4.run(FORWARD);
    motor4.setSpeed(255);
}

void loop() {
```

Conclusiones Los motores son lo suficientemente potentes como para mover el vehículo a una velocidad considerable, siempre y cuando los L293D sean alimentados con aproximadamente 8V, ya que hay una diferencia entre la entrada del L293D y la

salida a cada uno de los motores de =2V (los motores van a su máxima velocidad con 6V).

5.4.2. PWM

Para el análisis preliminar de PWM, se utilizó como código de prueba el ejemplo `pwm_dimmer`. Dicho código utiliza la modulación de ancho de pulso para variar la intensidad de los LEDs alojados en la placa de la EDU CIAA. A partir de dicho análisis se determinó la necesidad de configurar las correspondencias entre los canales del SCT con los pines del microcontrolador, ya que el proyecto requiere un total de 8 salidas PWM y con la configuración por defecto no eran suficientes; los canales asociados a CTOUT2, CTOUT4 y CTOUT5 están vinculados a los pines P2_10, P2_11 y P_12 correspondientes a los leds de la placa, dicha relación se establece en el archivo `sapi_sct.c` ubicado en las librerías del firmware v3 correspondientes a la sAPI. La modificación requerida consiste en derivar la salida del CTOUT2 a LCD1 de la siguiente manera:

```
static pinInitLpc4337_t SCTdataList[] = {  
  
    // Configuracion por default:  
    ...  
    /* Sct n / port / pin / name in board */  
    /* CTOUT2 */ { 4 , 10 }, /* LED1 */  
    ...  
  
    // Configuracion requerida:  
    ...  
    /* Sct n / port / pin / name in board */  
    /* CTOUT2 */ { 4 , 4 }, /* LCD1 */  
    ...  
}
```

Finalmente se verificó el correcto funcionamiento corriendo el programa y conectando en el terminal 30 de la placa, indicado como LCD1, un LED con su correspondiente resistencia.

6. Ensayos y Mediciones

6.1. Motores y L293D

Con el objetivo de observar el funcionamiento del L293D, los motores y verificar si estos eran capaces de mover el vehículo con todos los componentes a una velocidad aceptable, se realizó un prototipo con un Arduino, una caja de cartón que se utilizó como estructura provisoria (ver figura 9.8 y 9.9) y el shield que contiene dos L293D, configurando el PWM con un ciclo de trabajo del 100 %. Para facilitar el prototipado, se utilizó la librería para control de motores desarrollada por Adafruit. El código de prueba para el Arduino es el siguiente:

```
#include <AFMotor.h>

AF_DCMotor motor1(1);                                //Declara los motores a utilizar
AF_DCMotor motor2(2);
AF_DCMotor motor3(3);
AF_DCMotor motor4(4);

void setup() {
    Serial.begin(9600);                            //Inicializa la terminal serie a
    9600 bps
    Serial.println("Prueba de motores");

    motor1.run(RELEASE);                          //Desbloquea el motor
    motor1.run(FORWARD);                         //Mueve para adelante
    motor1.setSpeed(255);                         //Velocidad entre 0 y 255,
    configurada al máximo

    motor2.run(RELEASE);
    motor2.run(FORWARD);
    motor2.setSpeed(255);

    motor3.run(RELEASE);
    motor3.run(FORWARD);
    motor3.setSpeed(255);

    motor4.run(RELEASE);
    motor4.run(FORWARD);
    motor4.setSpeed(255);
}

void loop() {
```

Conclusiones Los motores son lo suficientemente potentes como para mover el vehículo a una velocidad considerable, siempre y cuando los L293D sean alimentados con aproximadamente 8V, ya que hay una diferencia entre la entrada del L293D y la salida a cada uno de los motores de aproximadamente 2V (los motores van a su máxima velocidad con 6V).

6.2. Sensor HC-SR04

Una vez ensamblado el auto en su correspondiente chasis en conjunto con la totalidad de sus componentes, realizamos las pruebas de detección de objetos. Estas pruebas consistieron en direccionar el auto hacia un obstáculo, contemplando el margen de frenado requerido para que el vehículo no choque contra dicho obstáculo, realizando sucesivas modificaciones en las variables de distancia de frenado hasta dar con la indicada. El código en cuestión se muestra a continuación:

```
void evaluar_colisiones() {  
  
    // obtenemos la distancia actual  
    distancia = ultrasonicSensorGetDistance(ULTRASONIC_SENSOR_0, CM)  
;  
  
    // antes de frenar por un obstaculo, se activa el buzzer para indicar la  
    // presencia del mismo  
    if(distancia <= 100 && auto_avanzando()){  
        sonido_colision();  
    }else{  
        desactivar_buzzer();  
    }  
  
    //Activamos el led de estado cuando detectamos una colision  
    if(distancia <= 30){  
        activar_led_colision();  
        bloquear();  
    }else{  
        desactivar_led_colision();  
        desbloquear();  
    }  
  
}  

```

6.3. Control remoto

Por simplicidad, se utilizó una aplicación disponible en el play store de Android, llamada BLEJoystick para manejar el vehículo vía el protocolo BLE y el módulo HM-10. Se probó el funcionamiento del Bluetooth utilizando el programa de ejemplo que viene en la librería SAPI para el uso del módulo HM-10, y el programa Hercules, que permite visualizar todos los comandos enviados desde la aplicación. Luego de configurar el Hercules en modo serie, y encontrar el puerto COM donde estaba conectada la EDU-CIAA, pudimos ver que la aplicación funcionaba correctamente, enviando el carácter correspondiente según la tecla presionada.

```
// FUNCION PRINCIPAL, PUNTO DE ENTRADA AL PROGRAMA LUEGO DE ENCENDIDO O RESET.  
int main( void )  
{  
    // ----- CONFIGURACIONES -----
```

```

// Inicializar y configurar la plataforma
boardConfig();

// Inicializar UART_USB para conectar a la PC
uartConfig( UART\_PC, 9600 );
uartWriteString( UART_PC, "UART_PC\u201aconfigurada.\r\n" );

// Inicializar UART_232 para conectar al modulo bluetooth
uartConfig( UART_BLUETOOTH, 9600 );
uartWriteString( UART_PC, "UART_BLUETOOTH\u201apara\u201amodulo\u201abluetooth\u201a
configurada.\r\n" );

uint8_t data = 0;

uartWriteString( UART_PC, "Testeo\u201asi\u201ael\u201amodulo\u201aesta\u201aconectado\u201a
enviando:\u201aAT\r\n" );
if( hm10bleTest( UART_BLUETOOTH ) ){
    uartWriteString( UART_PC, "Modulo\u201aconectado\u201acorrectamente.\r\n
" );
}
else{
    uartWriteString( UART_PC, "No\u201afunciona.\r\n" );
}

// ----- REPETIR POR SIEMPRE -----
while( TRUE ) {

    // Si leo un dato de una UART lo envio a al otra (bridge)
    if( uartReadByte( UART_PC, &data ) ) {
        uartWriteByte( UART_BLUETOOTH, data );
    }
    if( uartReadByte( UART_BLUETOOTH, &data ) ) {
        if( data == 'h' ) {
            gpioWrite( LEDB, ON );
        }
        if( data == 'l' ) {
            gpioWrite( LEDB, OFF );
        }
        uartWriteByte( UART_PC, data );
    }

    // Si presiono TEC1 imprime la lista de comandos AT
    if( !gpioRead( TEC1 ) ) {
        hm10blePrintATCommands( UART_BLUETOOTH );
        delay(500);
    }

    // Si presiono TEC3 enciende el led de la pantalla de la app
    if( !gpioRead( TEC3 ) ) {
        uartWriteString( UART_BLUETOOTH, "LED_ON\r\n" );
        delay(500);
    }
    // Si presiono TEC4 apaga el led de la pantalla de la app
}

```

```

    if( !gpioRead( TEC4 ) ) {
        uartWriteString( UART_BLUETOOTH , "LED_OFF\r\n" );
        delay(500);
    }
}

// NO DEBE LLEGAR NUNCA AQUI, debido a que a este programa se ejecuta
// directamente sobre un microcontroladore y no es llamado por ningun
// Sistema Operativo, como en el caso de un programa para PC.
return 0;
}

```

6.4. Buzzer

Al igual que con el sensor HC-SR04 los ensayos se realizaron con el auto una vez ensamblado. Determinamos la necesidad de utilizar dos tipos de intermitencias para diferenciar la detección de un obstáculo con la reversa de vehículo. La practica de dicho ensayo no implicó ningún inconveniente ya que solo consintió en modificar el código a gusto de manera que los tiempos sean los buscados.

```

void actualizar_buzzer(){
    if(tipo_de_sonido==1){
        contador++;
        if(contador==2){
            estado_buzzer=!estado_buzzer;
            gpioWrite(GPIO6, estado_buzzer);
            contador=0;
        }
    }

    if(tipo_de_sonido==2){
        contador++;
        if(contador==5){
            estado_buzzer=!estado_buzzer;
            gpioWrite(GPIO6, estado_buzzer);
            contador=0;
        }
    }
}

```

6.5. Baterías de litio

En base a mediciones realizadas previamente en la etapa de protipado, encontramos conveniente utilizar dos baterías de litio, cuya capacidad máxima de carga individual fue de 4v, lo cual a partir de su conexión en serie nos permitió obtener los 8v requeridos para la alimentación de los L293D así como también alimentar la EDU-CIAA con 5v gracias a la fuente step-down.

6.6. Fuente step down

Una vez corregida la disposición de la fuente step-down, se procedió a calibrarla, desconectando el resto de los componentes. Midiendo la tensión de salida con un multímetro, se llevó la tensión de salida a 5v, lo necesario para alimentar la EDU-CIAA.

7. Conclusiones

7.1. Cumplimiento de los objetivos

El objetivo principal del proyecto era crear un vehículo a escala que pueda ser controlado de manera remota por el usuario, se pudo comprobar la viabilidad tanto en términos de hardware como de software de los objetivos planteados desde un principio. Permitiendo, de esta manera, el desarrollo total del proyecto en tiempo y forma.

A su vez, se dividió el proyecto en 4 partes principales:

1. Estructura física del vehículo

La estructura del vehículo fue realizada con la caraza de una lectora de CD (ver foto 9.10). La misma fue modificada para poder colocar los 4 motores, la EDU-CIAA junto con el poncho y las pilas de litio.

Para una mejor apariencia física, se optó por remover la tapa superior de la caraza. De esta manera, quedan a simple vista todos los componentes que conforman el vehículo.

La estructura demostró ser lo suficientemente rígida y ligera para soportar todos los componentes de hardware, y a su vez, permitir que los motores puedan transportarla sin la necesidad de una potencia excesiva.

2. Control de motores

El control de los motores del vehículo se realizó a través del integrado L293D, el cual cuenta con dos puente H completos. Estos están ubicados sobre el poncho, distribuidos de manera que el conexionado permita que con un integrado se controle el motor delantero y trasero de un lateral del vehículo.

Cada motor permite el movimiento en ambos sentidos de giro, de manera que el vehículo cumple con los objetivos de avanzar, rotar y retroceder.

3. Control remoto

El usuario puede controlar el vehículo de manera remota utilizando el módulo HM-10, el cual funciona mediante el protocolo bluetooth.

Se optó por utilizar una aplicación de android ya implementada “BLEJoystick”, disponible en Google Play Store, siendo esta muy simple y sencilla de utilizar. De esta manera, se cumplió con el objetivo de poder manejar el vehículo desde una distancia de hasta aproximadamente 20 metros.

4. Medición de sensores y respuesta en consecuencia

Con el fin de detectar objetos/obstáculos delante del vehículo, se cuenta con un sensor de proximidad ultrasónico HC-SR04, ubicado en el frente del poncho.

Luego de realizar algunas pruebas, se observó que la distancia para bloquear el avance del vehículo debía ser configurada con algunos centímetros extras de margen, debido al tiempo de respuesta del sensor. De esta manera, se fijó una distancia máxima de 30cm, con un ángulo de visión de 15°, cumpliendo con el objetivo de bloquear el avance del vehículo justo antes de impactar con cualquier objeto.

A su vez, se agregó un buzzer con el objetivo de indicar el evento de colisión mediante sonido, generando una alerta cada vez que se detecta un obstáculo antes de realizar el bloqueo total del vehículo.

No ha sido implementado el objetivo secundario del sensor de luminosidad LDR, debido a la demanda de tiempo de dicha tarea y a la complejidad del diseño del poncho.

7.2. Cumplimiento de Requerimientos

7.2.1. Requerimientos Funcionales

Requerimientos de Software

Control de Motores

1. El sistema debe accionar 4 motores individualmente, permitiendo girar en ambas direcciones.

Se cumplió con el objetivo parcialmente. Dado que el pin *enable* de los L293D es compartido por todos los motores, por lo que se obtienen las siguientes características:

- Todas las ruedas se accionan en simultaneo, es decir, giran todas o no gira ninguna.
 - La velocidad de todas las ruedas puede controlarse de igual manera.
 - Individualmente puede controlarse: dirección de giro y frenado rápido.
2. El sistema debe accionar motores en conjunto, permitiendo avanzar, retroceder y girar.

Se cumplió completamente con el objetivo, ya que se puede:

- Avanzar si toda las ruedas giran hacia adelante.
- Retroceder si todas las ruedas giran hacia atrás
- Girar para ambos lados, si un par lateral trata de avanzar y el otro par retroceder. Como podemos ver a continuación:

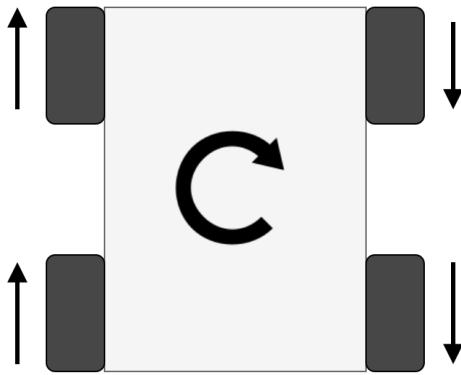


Figura 7.1: Dirección de giro

3. El sistema debe permitir el bloqueo de movimiento en una dirección (por ejemplo debido a un obstáculo detectado por el sensor de proximidad).

Se cumplió parcialmente con el objetivo, ya que se puede bloquear solo el movimiento hacia adelante. Esto puede verse en el código:

```
//bloquea el avance del vehiculo
void bloquear() {
    avance_bloqueado = TRUE;
}

...
void actualizar_desplazamiento() {

    switch (estado_auto) {

        //si el estado es adelante, todos los motores giran en un mismo
        sentido
        case ADELANTE:
            //El avance puede ser bloqueado por el detector de colisiones. En
            ese caso, se vuelve a actualizar el estado del vehiculo
            if (avance_bloqueado){
                estado_auto = FRENADO;
            }else{
                set_motor(M_DELANTERO_DER , M_MOV_ADELANTE ,
                          velocidad_avance);
                set_motor(M_DELANTERO_IZQ , M_MOV_ADELANTE ,
                          velocidad_avance);
                set_motor(M_TRASERO_DER , M_MOV_ADELANTE ,
                          velocidad_avance);
                set_motor(M_TRASERO_IZQ , M_MOV_ADELANTE ,
                          velocidad_avance);
            }
            break;

        ...
    }
}
```

4. OPT El sistema debe indicar mediante una serie de LEDs el estado de movimiento del vehículo.

No se cumplió con el objetivo opcional, ya que se buscó simplificar el diseño del poncho.

Control Remoto

1. El sistema debe establecer una conexión bluetooth con el sistema de control remoto.

Se cumplió con el objetivo, utilizando un módulo Bluetooth HM-10 provisto por la cátedra.

2. El sistema debe leer las indicaciones del control remoto y ejecutar las acciones correspondientes.

Se cumplió con el objetivo, utilizando como control remoto una aplicación de celular, cuyo funcionamiento ya fue detallado con anterioridad.

3. El sistema debe indicar mediante un LED el estado de la conexión bluetooth.

Se utilizó el LED que viene incluído en el módulo HM-10, que indica con una luz parpadeante si recibe alimentación, y una luz constante si se realizó la conexión.

OPT Medición Sensores

1. Ante ausencia de luz externa, se deben encender las luces del vehículo

No se cumplió con el objetivo, por demanda de tiempo y para simplificar el diseño del poncho.

2. Ante la presencia de obstáculos, se debe frenar el desplazamiento en esa dirección.

Se cumplió con el objetivo, utilizando un sensor de proximidad ultrasónica, HC-SR04. Al utilizar un único sensor, con ángulo de visión de 15°, el vehículo solo frena el desplazamiento hacia adelante.

Requerimientos de Hardware

Control de Motores

1. El vehículo debe utilizar puentes H para motores de corriente continua.

Se cumplió con el objetivo, utilizando dos integrados L293D, donde cada uno posee dos puentes H.

2. OPT El vehículo debe indicar con un LED para cada motor si este está en movimiento.

No se cumplió con el objetivo, para simplificar el diseño del poncho.

Baterías y cargador

1. El vehículo debe utilizar baterías de Litio para lograr la autonomía del vehículo
Se cumplió con el objetivo, utilizando dos baterías de Litio en serie obteniendo una tensión de alimentación de aproximadamente 8V.
2. El vehículo debe indicar mediante un LED cuando haya baja tensión
No se cumplió con el objetivo, no se pudo encontrar una manera sencilla de implementarlo.
3. El vehículo debe permitir cargar las baterías mediante un puerto USB.
No se cumplió con el objetivo, ya que cargar el dispositivo con todos los componentes conectados podría generar problemas eléctricos.

Componentes

1. Se debe contar con una llave de encendido/apagado y un LED que determine su estado.
Se cumplió con el objetivo, se cuenta con una llave y varios LEDs que indican estado, ya que hay LEDs de alimentación en el Poncho, en la EDU-CIAA y en el módulo Bluetooth.
2. OPT El sistema debe poseer un buzzer para indicar eventos o fallas
Se cumplió con el objetivo, el buzzer cumple dos funciones; notificar la presencia de obstáculos antes de bloquear el avance e indicar si el vehículo está retrocediendo.

7.2.2. Requerimientos No Funcionales

- Utilización de la placa de desarrollo EDU-CIAA.
- Estructura sólida y liviana, donde se colocan los motores y la EDU-CIAA.
- El conexiónado a los motores debe ser a través de borneras para poder cambiar la estructura.
- Programación en lenguaje C.
- Desarrollo de PCB en formato de poncho.
- Mecanismo de control simple.
- El sistema debe manejar respuestas en tiempo real.
- Fecha de finalización y entrega del proyecto el día 16/12.

Se cumplieron todos los requerimientos no funcionales en tiempo y forma.

7.2.3. Evaluación

El grado de cumplimiento de los objetivos, en general, fue altamente positivo. Los requerimientos funcionales que no se llevaron a cabo fueron los de menor prioridad, la mayoría relacionados al hardware, debido a que implicaban un costo de implementación muy alto para obtener una funcionalidad muy reducida. Por ejemplo, el caso de colocar un led que indique el estado de cada motor, lo cual costaría 4 pines GPIO, más toda su interconexión, aumentando aún más la complejidad del poncho.

El principal cambio realizado fue el uso de una placa doble faz, por dos motivos:

1. Obtener un plano de tierra para disipar el calor producido por los integrados L293D.
2. Facilitar la conexión a tierra de los componentes, junto con el uso de puentes.

Otro cambio fue el uso de una sola fuente step-down. En un principio la alimentación iba a consistir de dos pilas de litio en paralelo, las cuales entregarían aproximadamente 4v, y creímos que sería necesario utilizar dos conversores; un conversor step-up para alimentar los L293D con 8v, y un conversor step-down para alimentar la EDU-CIAA con 3.3v.

Una vez estudiada mejor la arquitectura de la EDU-CIAA, observamos que se alimentaba con 5v, y que internamente tiene un conversor a 3.3v para aquellos puertos o componentes que necesiten de dicha tensión. Por este motivo, cambiamos las pilas de paralelo a serie para obtener 8v que alimentan directamente a los L293D, y utilizamos una fuente step-down para alimentar la EDU-CIAA con 5v a través del conector Molex para alimentación externa.

Un error que cometimos fue implementar mal la huella del conversor step-down, la cual quedó completamente espejada, por lo que hubo que colocarla sobre pines, y la tierra soldarla sobre el plano en lugar de atravezando la placa.

7.3. Actividad de los integrantes

Tarea	Descripción	Tiempo
Compra de componentes	Se obtuvieron los componentes electrónicos principales, los puentes H (L293D) no se consiguieron individualmente por lo que se compró un shield de Arduino que posee 2. Todos los componentes fueron comprados en La Plata salvo las ruedas y motores. Componentes que requerían valores precisos como resistencias y capacitores se fueron comprando a medida que se concretaba el diseño.	3 Horas

Tarea	Descripción	Tiempo
Estudio BT	Mediante el uso de un módulo HM-10, el programa Hércules, la sAPI y un aplicación para Android, se estudió el funcionamiento de la comunicación Bluetooth. Se pudo concluir lo descripto en la sección de Software, Bluetooth.	8 Horas
Arquitectura de Software	Se diseño la estructura principal del programa, como se describió en la sección de software	4 Horas
Estudio L293D	Leyendo las datasheets del L293D y con recomendaciones de la cátedra, se llegó a un diseño para la PCB que cumplía con los requisitos eléctricos y térmicos. Además aquí también se incluye la investigación del módulo PWM de la EDU-CIAA, para poder controlar la potencia de los motores	8 horas
Prototipo L293D	Se utilizó un Arduino y el Shield mencionado para probar algunos motores y el funcionamiento de los puentes H	8 horas
Estudio de sensores	Se investigó principalmente el funcionamiento del sensor de proximidad HC-SR04	4 horas
Estudio de batería	Se vieron diferencias opciones para la implementación de las baterías, la caída de tensión en el L293D, tensión de las pilas de litio	2 horas
Desarrollo de Software prioritario	Se implementaron las partes más importantes del programa, el control de motores y la comunicación Bluetooth	8 horas
Estructura	Se dejó para el final para centrarnos en la investigación y desarrollo del poncho	4 horas
Prototipo de Estructura	El prototipo fue una precaria caja de zapatos, como se puede observar en las fotos	4 horas
Diseño de PCB	El diseño de la PCB fue evolucionando constantemente, la estructura y disposición de los componentes se mantuvo pero se fueron optimizando las pistas.	16 horas
Fabricación de PCB	La PCB se fabricó en el ATEI con ayuda de Alejandro	6 horas
Desarrollo de software secundario	No se cumplió con el cronograma original, y se desarrolló los últimos días	6 horas
Montaje	Se soldaron los componentes, se colocaron los motores, tornillos de montaje, y porta pilas	6 horas
Elección de baterías	Dado el tamaño solo cabían dos pilas, las necesarias para llegar 8v.	2 minutos
Calibración de Motores	Se reguló la velocidad y se verificó que el mapeo de software coincida con la implementación en hardware	30 minutos

Tarea	Descripción	Tiempo
Validación	Se verificó que el poncho y el software funciona correctamente y de acuerdo a lo especificado. Hubo que reprogramar diversos sectores, como el sensor de proximidad, que no actuaban acorde a lo especificado	12 horas

7.4. Presupuesto

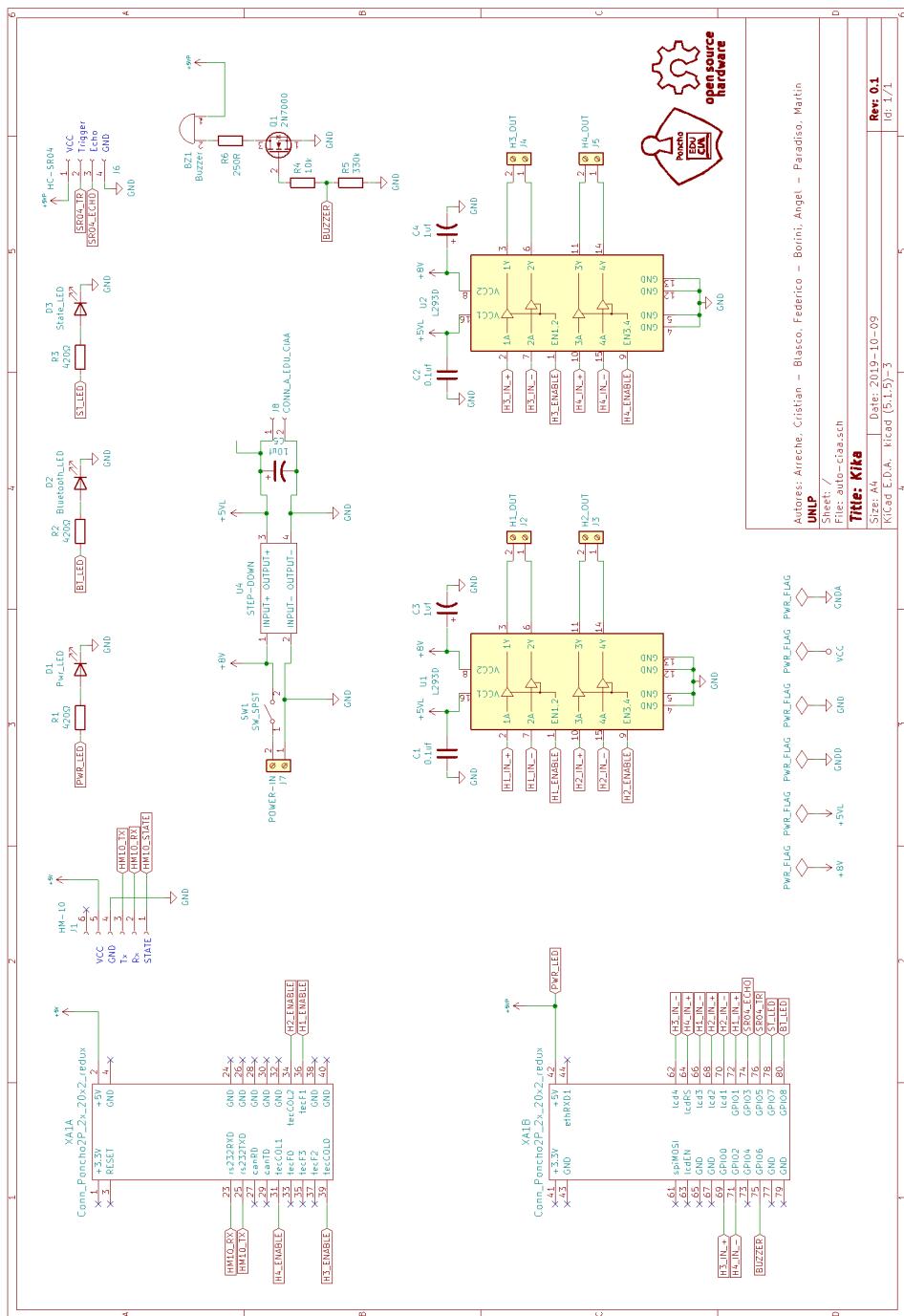
Producto	Precio
Shield motor driver arduino	\$ 350
Ruedas con caja reductora y motores	\$ 1200
Porta pilas de litio	\$ 150
Placa virgen 10x15 doble faz	\$ 180
Mini switch	\$ 45
HC-SR04	\$ 170
Tira de 2x40 pines macho	\$ 80
Step Down mini 360	\$ 150
Buzzer	\$ 30
Resistencias/capacitores/leds/varios	\$ 200
Transistor 2n7000	\$ 20
Hoja A4 papel fotográfico	\$ 20
Horas de ingeniería (400hs.)	\$ 72000
Total	\$ 74595

8. Bibliografía

1. Hoja de datos del microcontrolador LPC4337 utilizado por la EDU-CIAA-NXP.
Disponible en: https://www.nxp.com/docs/en/data-sheet/LPC435X_3X_2X_1X.pdf
2. Hoja de datos de L293D. Disponible en: <http://www.ti.com/lit/ds/symlink/l293.pdf>

9. Anexo

9.1. Esquemático Completo



9.2. Imágenes de Capas

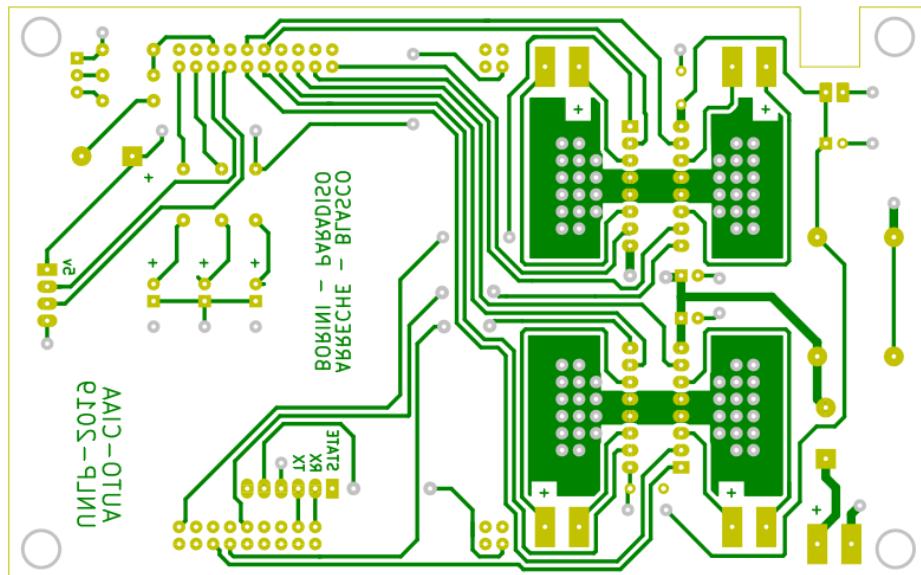


Figura 9.1: Capa de cobre inferior

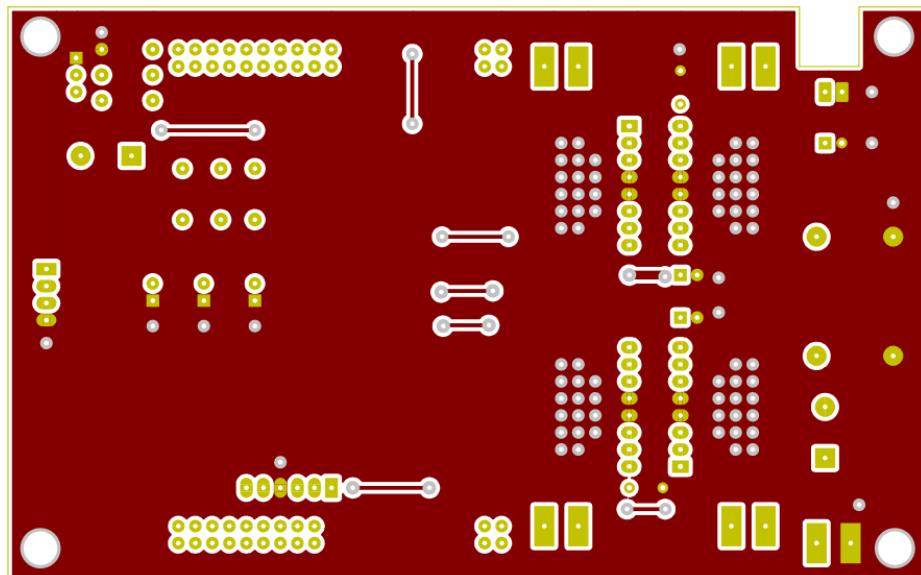
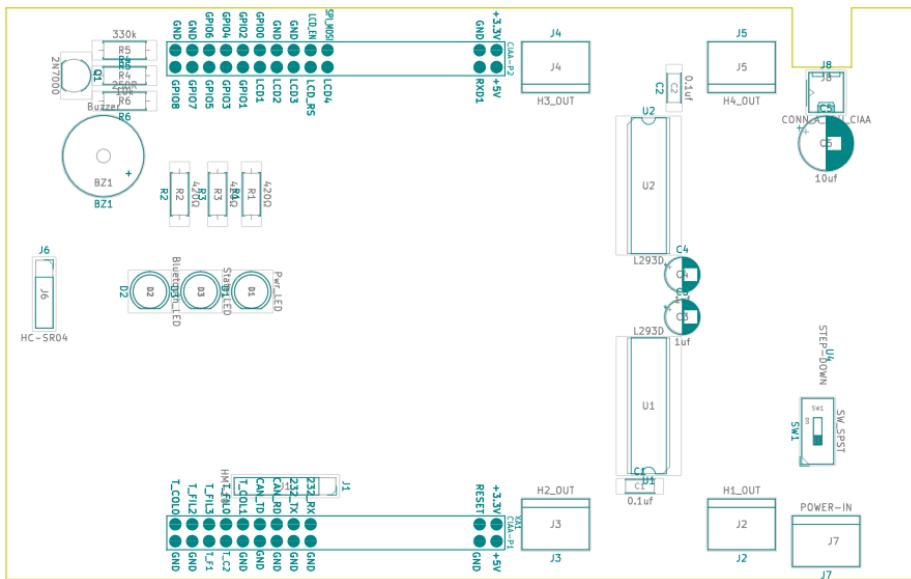


Figura 9.2: Capa de cobre superior (con plano de tierra)



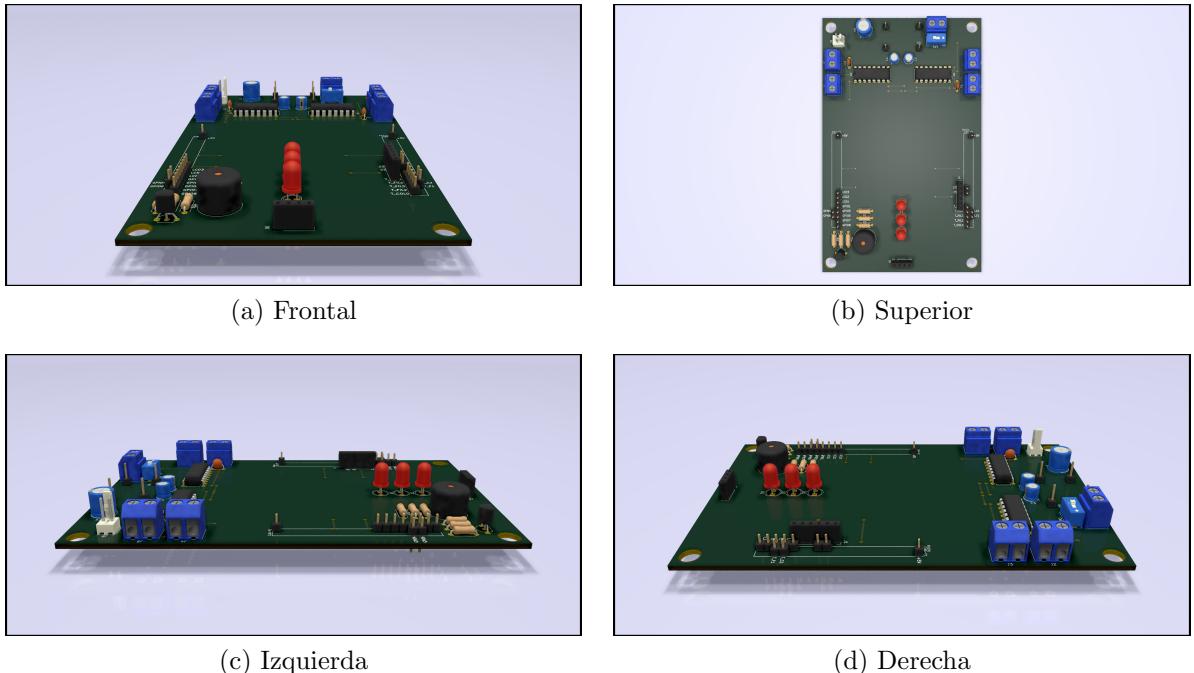


Figura 9.5: Renders generados en 3D con KiCAD

9.4. Información general

- HM-10 https://github.com/ciaa/firmware_v3/blob/master/examples/c/sapi/bluetooth/hm10_uart_bridge/EDU-CIAA-NXP%20y%20BLE%204.0%20HM10.pdf

9.5. Aplicaciones Externas

- BLEJoystick <https://play.google.com/store/apps/details?id=iyok.com.blejoystick>

9.6. Datasheets

- Transistor 2n7000 Disponible: <https://www.onsemi.com/pub/Collateral/2N7000-D.PDF>
- L293D Texas Disponible: <http://www.ti.com/lit/ds/symlink/l293.pdf>
- L293D STMicroelectronics Disponible: <https://www.st.com/resource/en/datasheet/l293d.pdf>
- HC-SR04 <http://raspoid.com/download/datasheet/HCSR04>

- HC-SR04 <https://cdn.sparkfun.com/datasheets/Sensors/Proximity/HCSR04.pdf>
- HC-SR04 <https://www.electroschematics.com/hc-sr04-datasheet/>

9.7. BOM

Referencia	Encapsulado	Huella
BZ1	-	Buzzer_Beeper:Buzzer_12x9.5RM7.6
C1	C. Cerámico	Capacitor_THT:C_Disc_D4.3mm_W1.9mm_P5.00mm
C2	C. Cerámico	Capacitor_THT:C_Disc_D4.3mm_W1.9mm_P5.00mm
C3	C. Electrolítico	Capacitor_THT:CP_Radial_D5.0mm_P2.50mm
C4	C. Electrolítico	Capacitor_THT:CP_Radial_D5.0mm_P2.50mm
C5	C. Electrolítico	Capacitor_THT:CP_Radial_D5.0mm_P2.50mm
D1	-	LED_THT:LED_D5.0mm
D2	-	LED_THT:LED_D5.0mm
D3	-	LED_THT:LED_D5.0mm
J1	-	Connector_PinSocket_2.54mm:PinSocket_1x06_P2.54mm_Vertical
J2	-	TerminalBlock:TerminalBlock_bornier-2_P5.08mm
J3	-	TerminalBlock:TerminalBlock_bornier-2_P5.08mm
J4	-	TerminalBlock:TerminalBlock_bornier-2_P5.08mm
J5	-	TerminalBlock:TerminalBlock_bornier-2_P5.08mm
J6	-	Connector_PinSocket_2.54mm:PinSocket_1x04_P2.54mm_Vertical
J7	-	TerminalBlock:TerminalBlock_bornier-2_P5.08mm
J8	KK254	Connector_Molex:Molex_KK-254_AE-6410-02A_1x02_P2.54mm_Vertical
Q1	TO-92	Package_TO_SOT_THT:TO-92_Inline_Wide
R1	DIN0207	Resistor_THT:R_Axial_DIN0207_L6.3mm_D2.5mm_P7.62mm_Horizontal
R2	DIN0207	Resistor_THT:R_Axial_DIN0207_L6.3mm_D2.5mm_P7.62mm_Horizontal
R3	DIN0207	Resistor_THT:R_Axial_DIN0207_L6.3mm_D2.5mm_P7.62mm_Horizontal
R4	DIN0207	Resistor_THT:R_Axial_DIN0207_L6.3mm_D2.5mm_P7.62mm_Horizontal
R5	DIN0207	Resistor_THT:R_Axial_DIN0207_L6.3mm_D2.5mm_P7.62mm_Horizontal
R6	DIN0207	Resistor_THT:R_Axial_DIN0207_L6.3mm_D2.5mm_P7.62mm_Horizontal

Referencia	Encapsulado	Huella	
SW1	-	Button_Switch_THT:SW_DIP_SPSTx01_de.9.78x4.72mm_W7.62mm_P2.54mm	Sli-
U1	DIP16	Package_DIP:DIP-16_W7.62mm	
U2	DIP16	Package_DIP:DIP-16_W7.62mm	

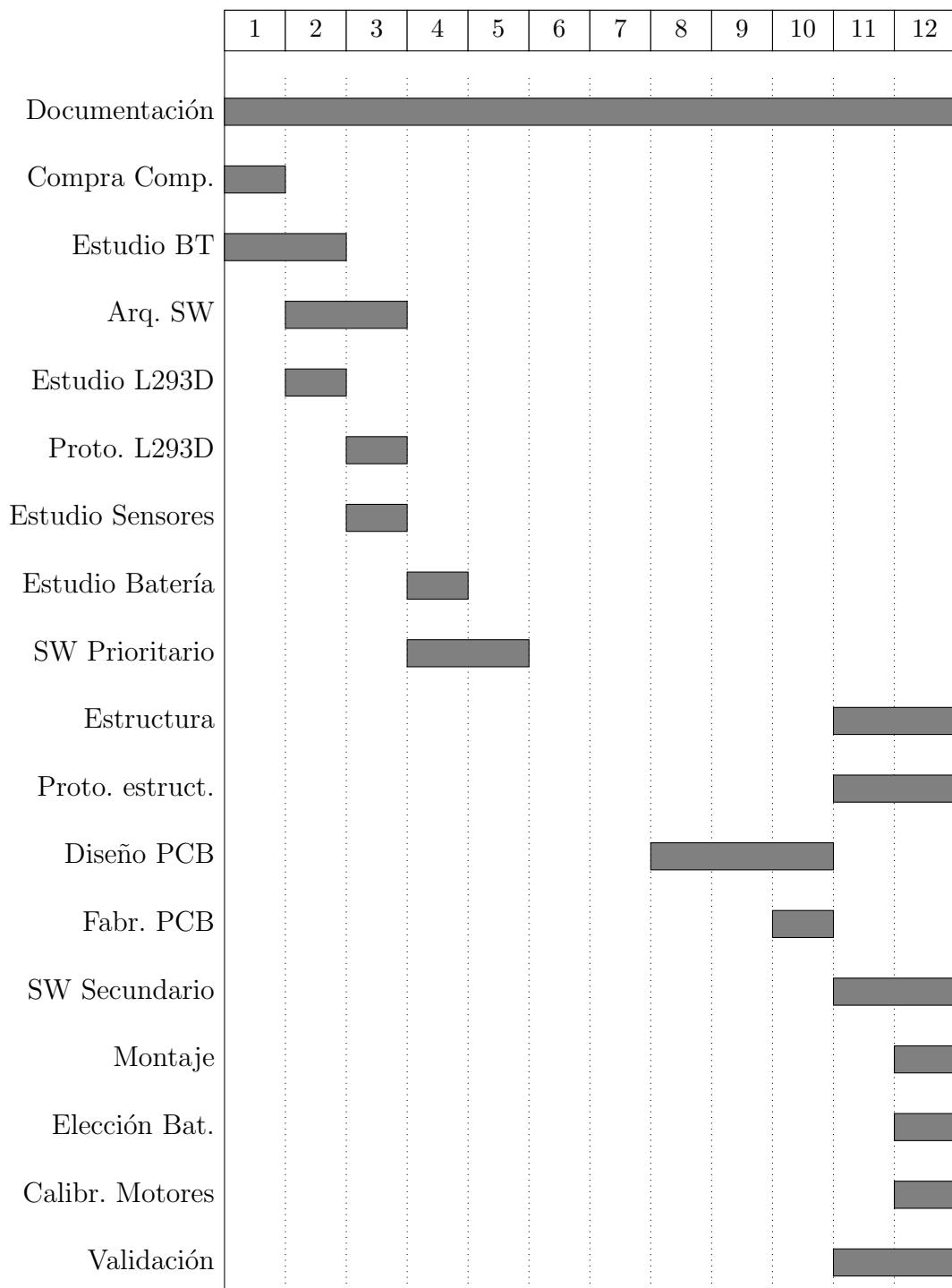
9.8. Imágenes de los Componentes

9.8.1. mini360



Figura 9.6: Fuente Step-Down

9.9. Cronograma



Las unidades de tiempo son semanas. Consideramos dos semanas de Validación como margen en caso de eventualidades que atrasen el proyecto.

9.10. División de Tareas del Grupo

Tarea a realizar	Alumno/s
Obtención de electrónica	Arreche, Blaseo, Borini, Paradiso
Estudio de protocolos Bluetooth	Arreche, Blaseo, Borini, Paradiso
Arquitectura de software	Arreche, Blaseo, Borini, Paradiso
Estudio de L293D	Arreche, Blaseo, Borini, Paradiso
Prototipo L293D	Arreche, Blaseo, Borini, Paradiso
Investigación de sensores	Arreche, Blaseo, Borini, Paradiso
Investigación de baterías y carga	Arreche, Blaseo, Borini, Paradiso
Desarrollo de software prioritario	Arreche, Blaseo, Borini, Paradiso
Diseño y fabricación de estructura	Arreche, Blaseo, Borini, Paradiso
Proto. de estructura con motores y L293D	Arreche, Blaseo, Borini, Paradiso
Diseño de la PCB	Arreche, Blaseo, Borini, Paradiso
Fabricación de PCB	Arreche, Blaseo, Borini, Paradiso
Desarrollo de SW secundario y sensores	Arreche, Blaseo, Borini, Paradiso
Montaje final	Arreche, Blaseo, Borini, Paradiso
Elección de baterías	Arreche, Blaseo, Borini, Paradiso
Calibración de motores	Arreche, Blaseo, Borini, Paradiso
Validación	Arreche, Blaseo, Borini, Paradiso
Documentación	Arreche, Blaseo, Borini, Paradiso

9.11. Manual de Usuario

1. Para poder utilizar el vehículo, se debe tener la aplicación BLE Joystick instalada en un teléfono Android.

<https://play.google.com/store/apps/details?id=iyok.com.blejoystick>

2. El vehículo contiene una llave para cortar la alimentación de baterías:

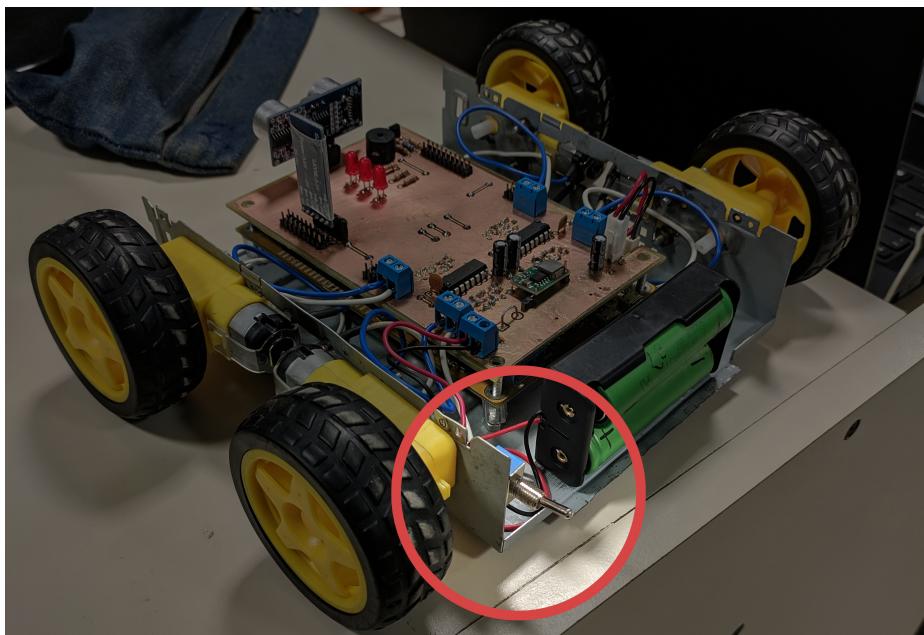


Figura 9.7: Llave de encendido

3. Conectar el teléfono al módulo Bluetooth del vehículo, una vez establecida la conexión, el LED del módulo HM-10 quedará fijo y dejará de titilar.

9.12. Imágenes del Proyecto

9.12.1. Desarrollo

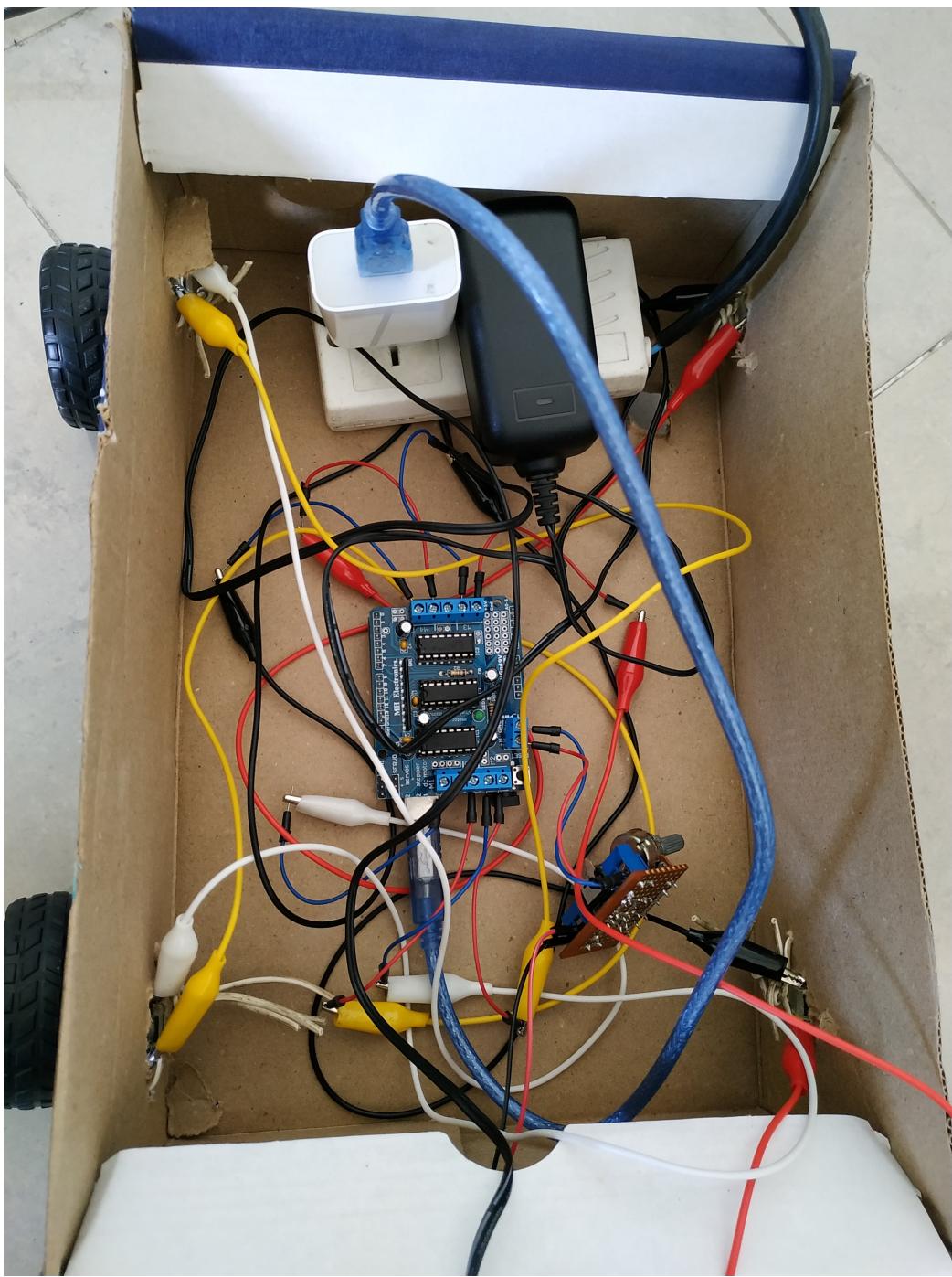


Figura 9.8: Prototipo utilizando Arduino

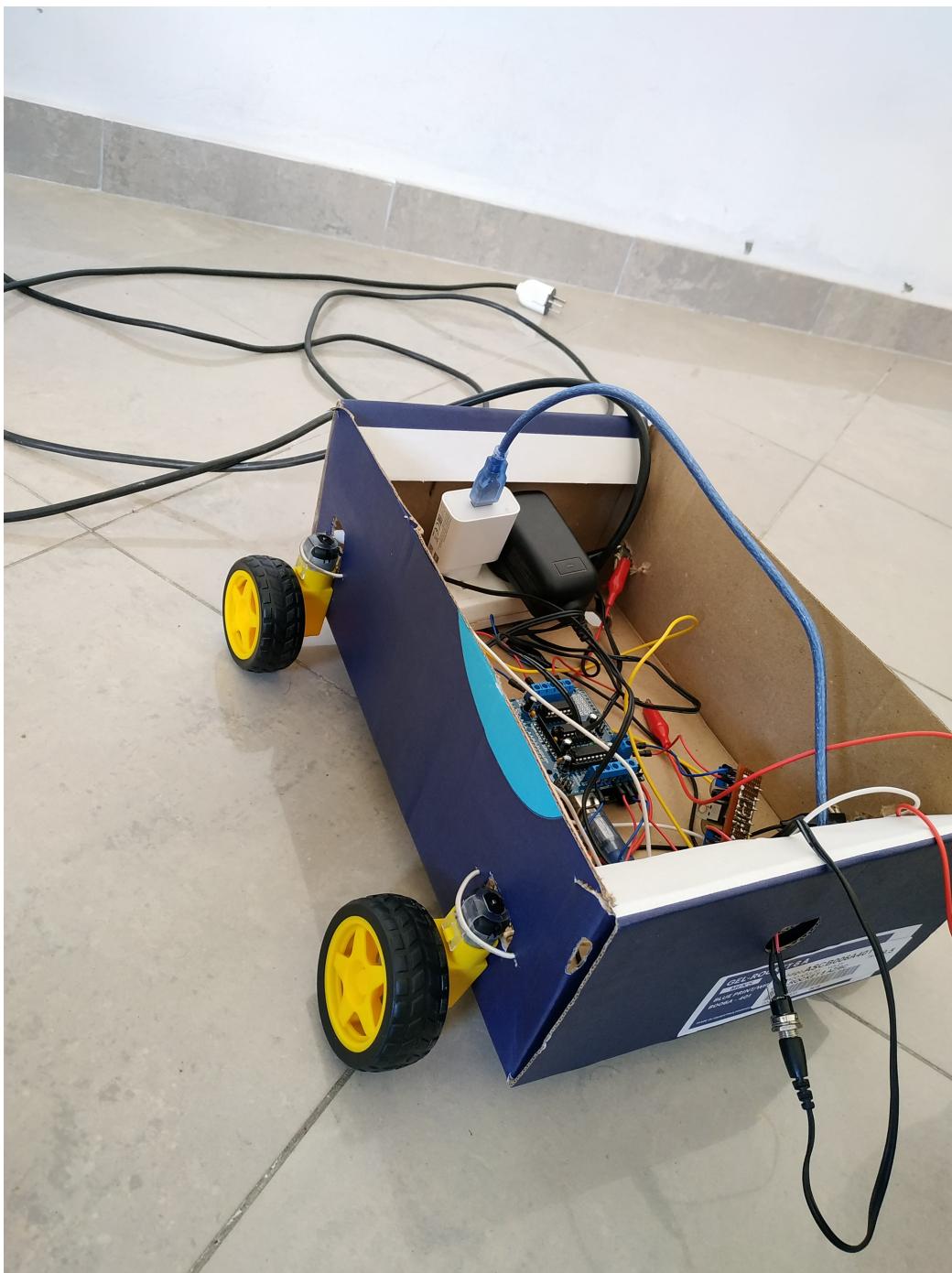


Figura 9.9: Prototipo utilizando Arduino



Figura 9.10: Fabricación de la estructura

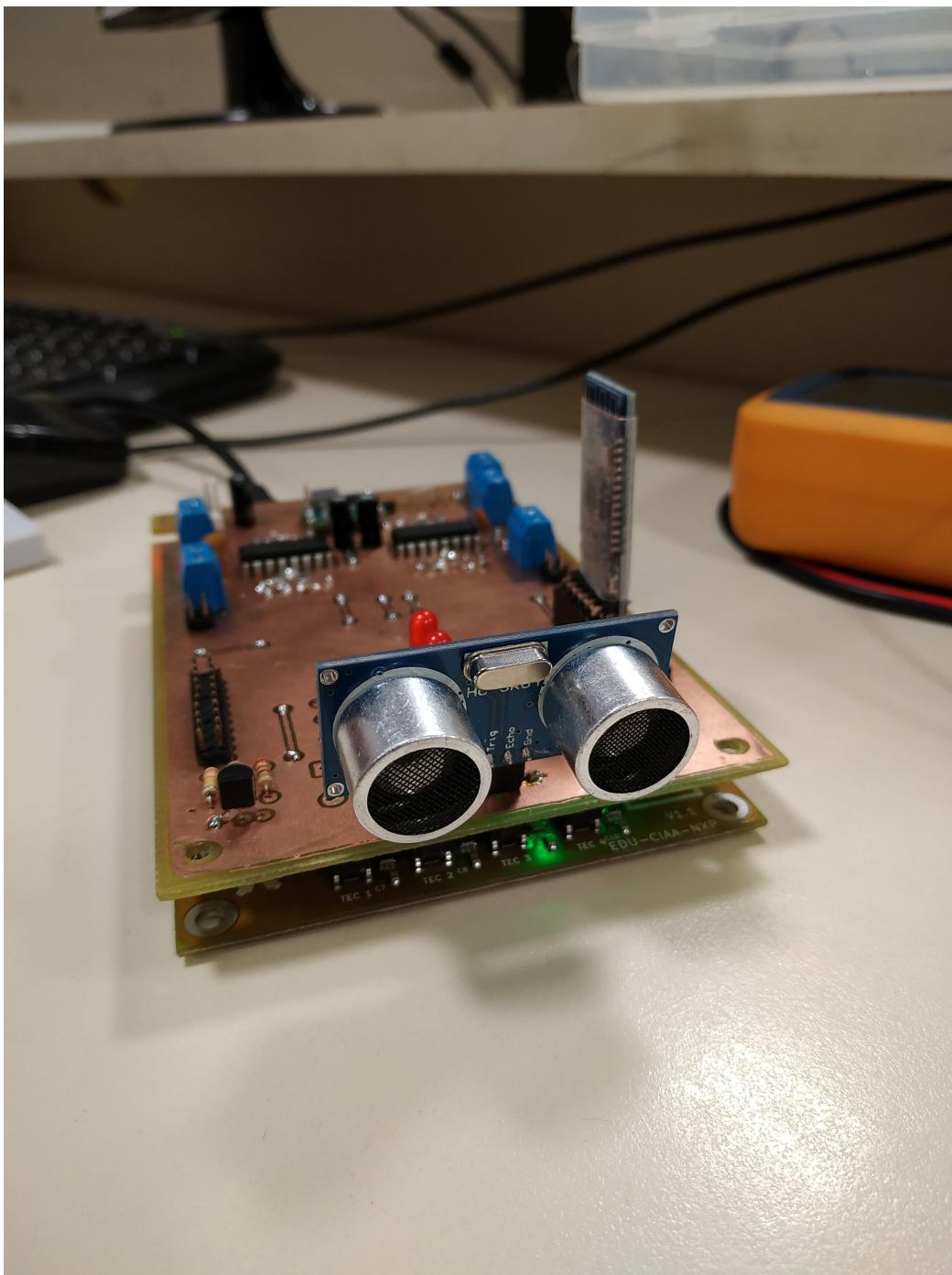
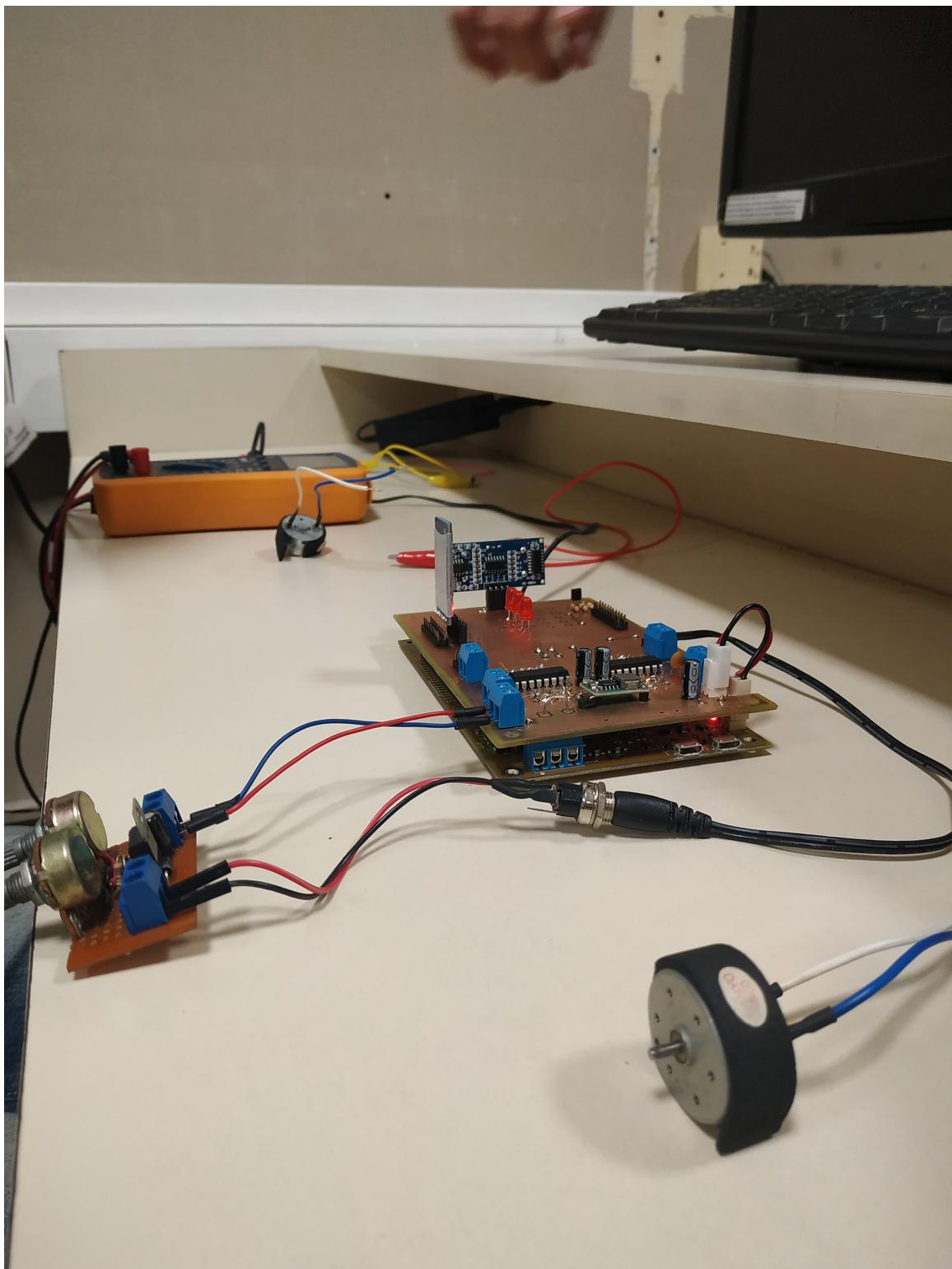
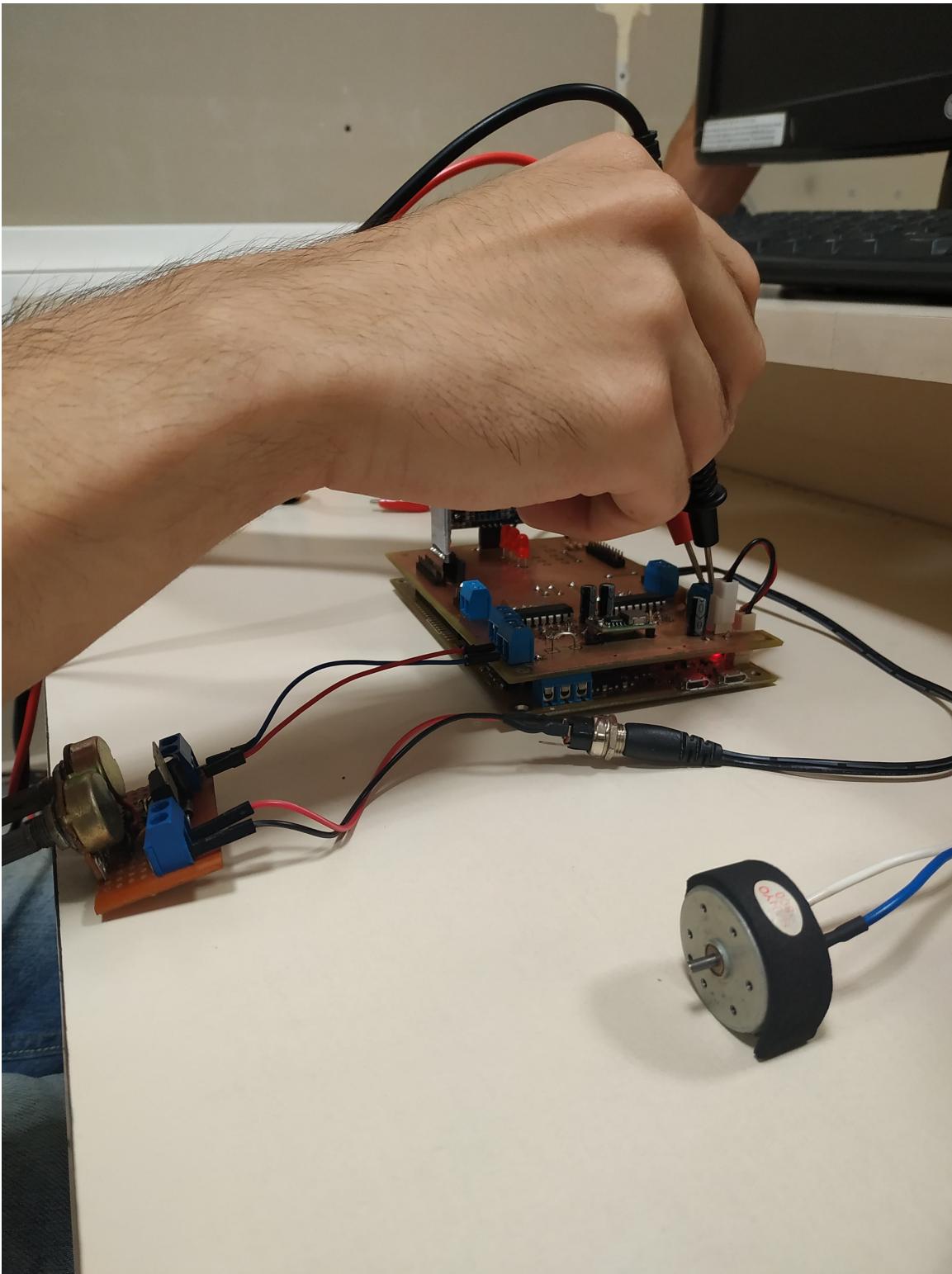
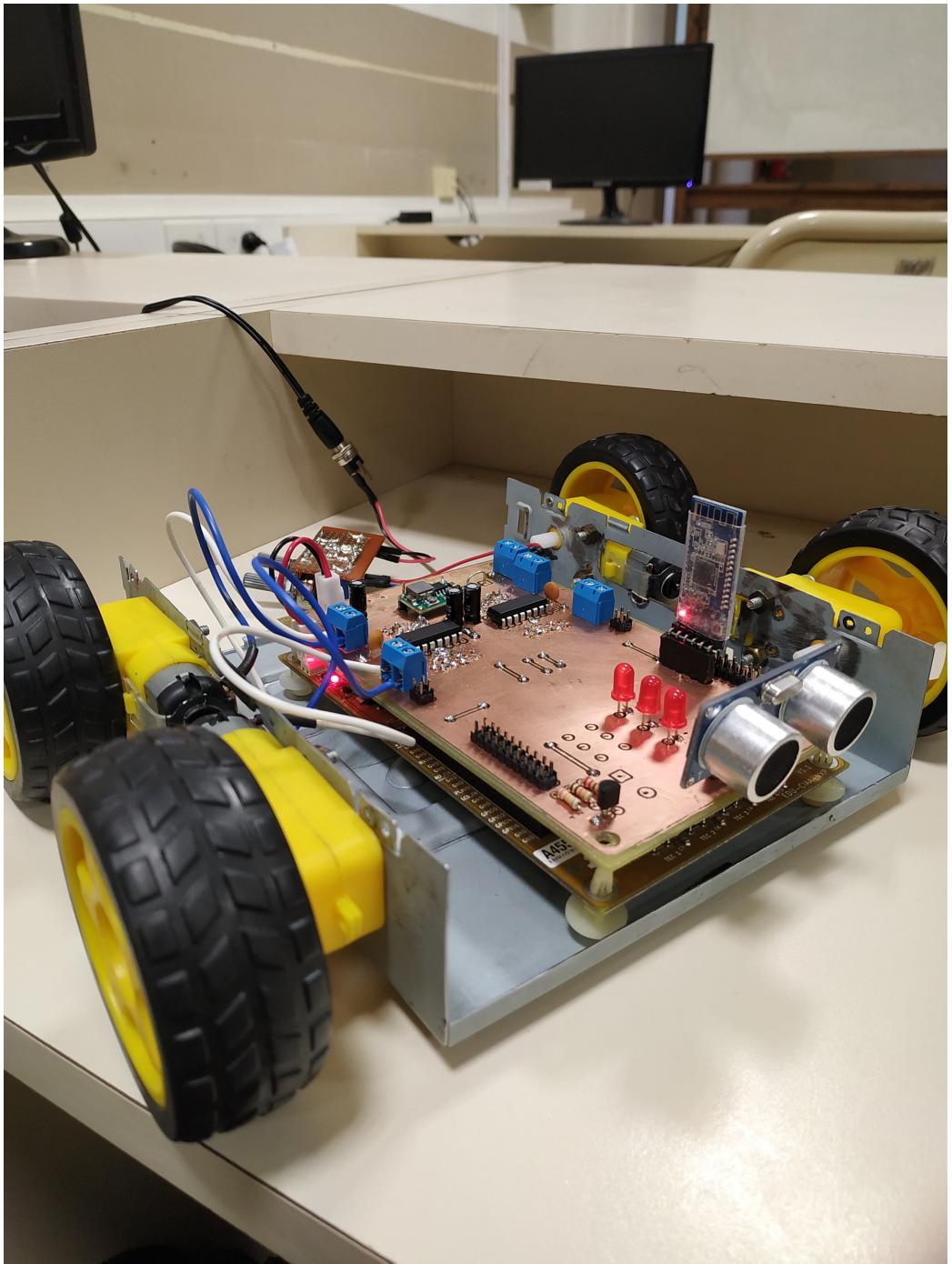


Figura 9.11: Pruebas de software







9.12.2. Proyecto terminado

