UNIVERSIDAD NACIONAL DE AVELLANEDA

INGENIERÍA EN INFORMÁTICA

ARQUITECTURA DE SISTEMAS DE ELABORACIÓN DE DATOS 2

AÑO 2023 – 1° Cuatrimestre

TRABAJO DE LABORATORIO 1

Datos de los integrantes:

Apellido y Nombres

Fenger Leandro

Vidueiros Lopez Nicolas

Gandolfo Alejandro

Profesor: Matias Loiseau

Contenido

1 - Objetivo

El objetivo del presente informe es el de dar a conocer el diseño e implementación de un sistema embebido para controlar el sentido de giro de 4 (cuatro) motores D.C. según la información proveniente de un sensor de proximidad.
El sistema se deberá comportar de acuerdo con las siguientes condiciones:

● Si la distancia recibida por el sensor de proximidad es mayor a 15 cm, los 4 (cuatro) motores deben avanzar hacia adelante, es decir los 4 tendrán el mismo sentido de giro.

● Si la distancia recibida por el sensor de proximidad es menor o igual a 15 cm, 2(dos) de los motores debe girar hacia atrás y los 2(dos) restantes deberán ir hacia adelante, lo que permitirá que el dispositivo pueda girar sobre su eje.

2 – Introducción

En el presente trabajo se implementará un circuito electrónico computarizado diseñado para controlar el sentido de giro de cuatro motores de corriente continua. El circuito estará compuesto por componentes electrónicos interconectados entre sí mediante la plataforma principal EDU-CIAA-NXP, una computadora diseñada y fabricada en Argentina con el objetivo de fomentar el uso y desarrollo de la tecnología en el ámbito educativo.

El hardware utilizado para este proyecto incluye un Puente H, el cual es un circuito electrónico que se utiliza comúnmente para el control de motores de corriente continua, permitiendo su rotación en ambos sentidos (horario y antihorario). Este tipo de circuitos se utilizan ampliamente en robótica y en la conversión de potencia.

También se empleará un sensor ultrasónico, que es un dispositivo de proximidad que detecta objetos a diferentes distancias sin necesidad de contacto físico. El sensor emite ondas sonoras de alta frecuencia en el rango de ultrasonido, fuera del alcance de la percepción humana, y recibe las ondas que rebotan en un objeto. La información obtenida del tiempo que tarda el sonido en regresar al sensor se utiliza para calcular la distancia al objeto.

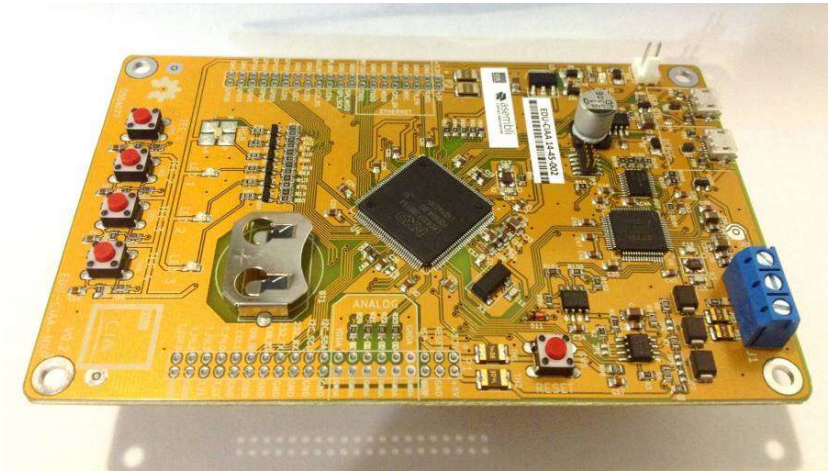
Además, se utilizará una protoboard para realizar las conexiones entre los distintos componentes y una fuente ATX genérica para alimentar la placa EDU-CIAA-NXP y los demás componentes.

En cuanto al software, se utilizará el editor de código Embedded IDE para programar la EDU-CIAA-NXP en lenguaje C y utilizando la biblioteca sAPI. La función de esta biblioteca es proporcionar una capa de abstracción del hardware que permita que los programas sean portables y que todas las placas sean programadas con las mismas funciones.

3 - Marco teórico

Para trabajar, hemos empleado, la placa EDU-CIAA, la cual es un sistema embebido (también conocido como “empotrado”, “incrustado” o “integrado”). Es un sistema de computación diseñado para realizar funciones específicas, y cuyos componentes se encuentran integrados en una placa base (en inglés. “motherboard”). El procesamiento central del sistema se lleva a cabo gracias a un microcontrolador, es decir, un microprocesador que incluye además interfaces de entrada/salida, así como una memoria de tamaño reducido en el mismo chip.
Estos sistemas pueden ser programados directamente en el lenguaje ensamblador del microcontrolador o microprocesador o utilizando otros lenguajes como C o C + + mediante compiladores específicos.
Son diseñados generalmente para su utilización en tareas que impliquen una computación en tiempo real, pero también destacan otros casos como son Arduino y Raspberry Pi, cuyo fin está más orientado al diseño y desarrollo de aplicaciones y prototipos con sistemas embebidos desde entornos gráficos.[5]
Las bibliotecas sAPI se usan en varias de las placas del proyecto CIAA. Su origen es de la tesis de Eric Pernia quien es el que actualiza el repositorio en su GitHub agregando y mejorando los archivos que él Embedded IDE descargan desde la plantilla sAPI. La idea de la sAPI es que se programen todas con las mismas funciones y que sirvan como capa de abstracción del hardware

La EDU-CIAA-NXP es una versión de bajo costo de la CIAA-NXP pensada para la enseñanza universitaria, terciaria y secundaria.



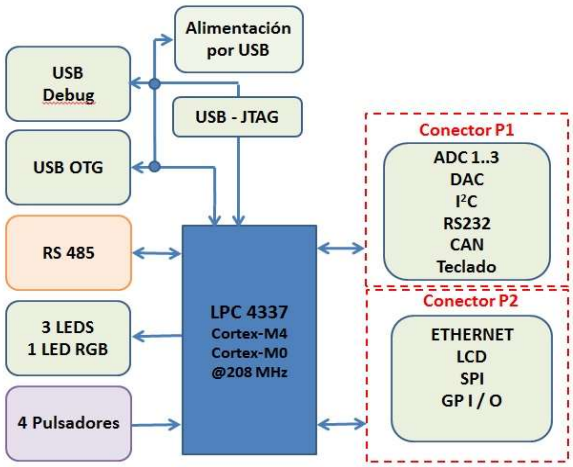
Bloques funcionales:

La EDU-CIAA está basada en la CIAA-NXP, por ser la primera versión de la CIAA que se encuentra disponible. Por lo tanto, su microcontrolador es también el LPC 4337 (dual core ARM Cortex-M4F y Cortex-M0).

Sin embargo, con el objetivo de abaratar costos y reducir su complejidad la EDU-CIAA incorpora sólo algunas de las funcionalidades de la CIAA.
A su vez, con el fin de permitir el desarrollo de algunas prácticas sencillas sin que sea necesario recurrir a hardware adicional, incluye además algunos recursos que no están presentes en la CIAA.

Diagrama en bloques de la plataforma:

En la siguiente figura se observa un diagrama en bloques de la EDU-CIAA basada en LPC 4337:



La EDU-CIAA cuenta con los siguientes módulos:

- 2 puertos micro-USB (uno para aplicaciones y debugging, otro para alimentación).
- 4 salidas digitales implementadas con leds RGB.
- 4 entradas digitales con pulsadores.
- 1 puerto de comunicaciones RS 485 con bornera.
- 2 conectores de expansión:

P1:

- 3 entradas analógicas (ADC0_1,2y3),
- 1 salida analógica (DAC0),
- 1 puerto I2C,
- 1 puerto asincrónico full dúplex (para RS-232).
- 1 puerto CAN,
- 1 conexión para un teclado de 3×4,

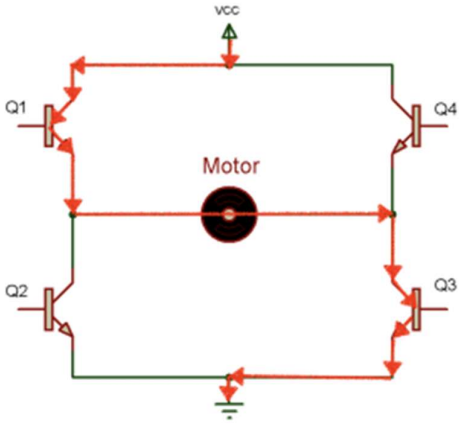
P2:

- 1 puerto Ethernet,
- 1 puerto SPI,
- 1 puerto para Display LCD con 4 bits de datos, Enable y RS.
- 9 pines genéricos de I/O.

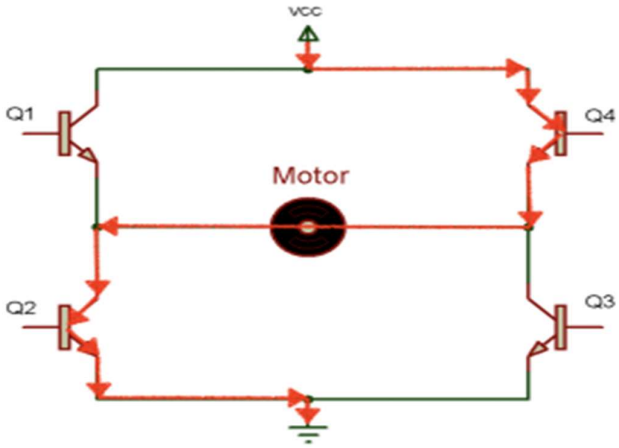
En este proyecto se ha utilizado el módulo L298N, el cual tiene la funcionalidad de comportarse como un puente H. Dicho módulo es un circuito electrónico que permite a un motor eléctrico de corriente continua girar en ambos sentidos (horario y antihorario).

Cabe destacar que un puente H no solo se utiliza para invertir el giro de un motor, sino que también puede ser empleado para frenarlo de manera brusca mediante un cortocircuito entre los bornes del motor. Incluso, se puede permitir que el motor frene bajo su propia inercia cuando se desconecta de la fuente de alimentación.

En las siguientes imágenes se podrá apreciar el sentido de giro de acuerdo con la dirección de circulación de la corriente en los motores que forman parte del circuito.



Corriente eléctrica en un sentido

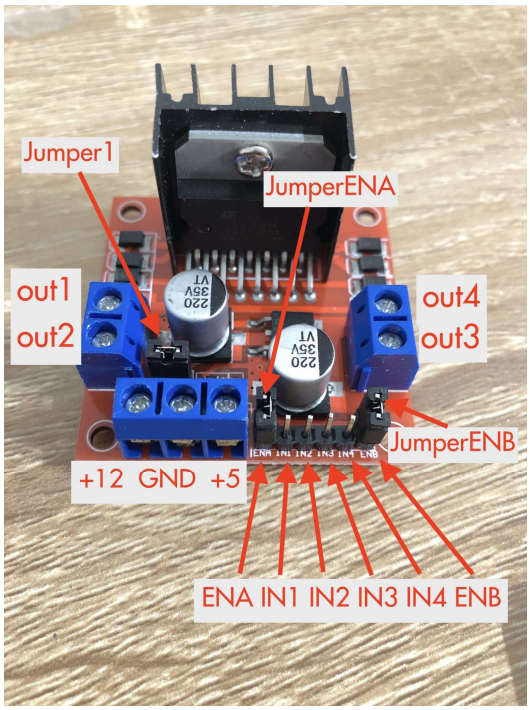


Corriente en sentido contrario

El módulo L298N, contiene terminales de conexión y jumpers (puentes de conexión), que de acuerdo con su disposición determinarán el comportamiento de este, como lo indica el recuadro a continuación:

Jumper	Terminal +12	Terminal +5
Conectado	Entrada de 6 a 12 VCD	Salida de 5 VDC
Desconectado	Entrada de 12 a 32 VCD	Entrada de 5 VDC

Relación entre Jumper y terminales



Conexiones L298N

Para el correcto funcionamiento de los motores, hay que tener en cuenta las direcciones de giro que otorgará el módulo L298N, para posteriormente hacer una correcta implementación de código que funcione de acuerdo con lo esperado. Básicamente, hay que considerar que cuando las entradas son distintas, se realiza un giro, cuando ambas son LOW se detienen. Solamente hay que asegurarse de nunca poner ambas en HIGH al mismo tiempo.

Dirección del giro	Adelante	Atrás	Detener
IN1 o IN3	HIGH	LOW	LOW
IN2 o IN4	LOW	HIGH	LOW

Direcciones de giro

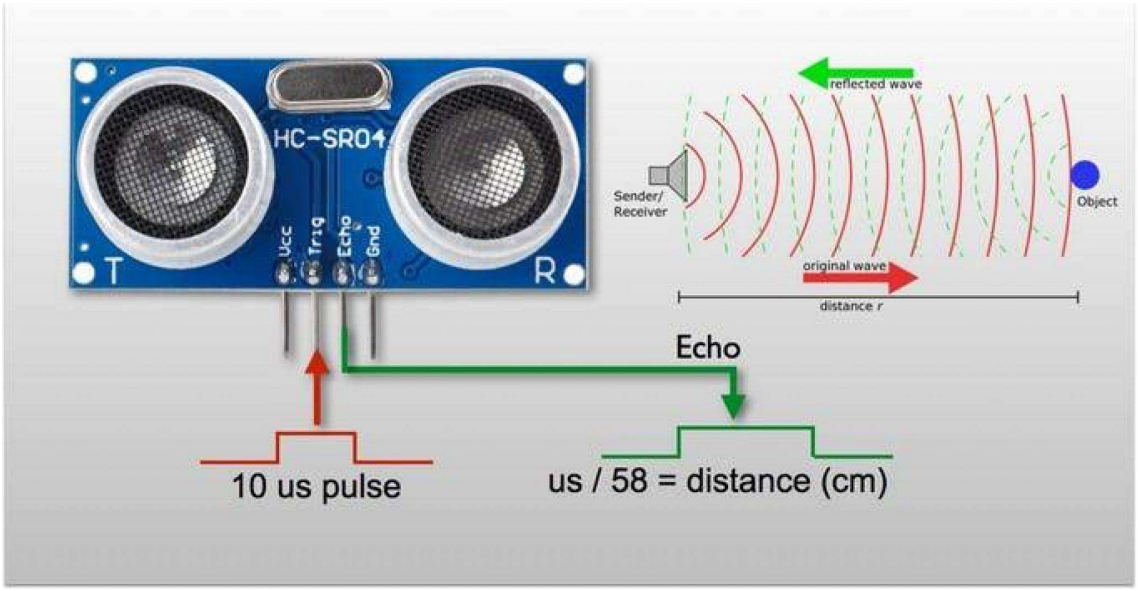
Para controlar el giro del dispositivo ante un obstáculo, hemos usado el sensor HC-SR04, el cual es un sensor de distancia de bajo costo que utiliza ultrasonido para determinar la distancia de un objeto en un rango de 2 a 450 cm. El sensor HC-SR04 es el más utilizado dentro de los sensores de tipo ultrasonido, principalmente por la cantidad de información y proyectos disponibles en la web.

El emisor piezoeléctrico emite 8 pulsos de frecuencia de ultrasonido(40KHz). Luego de recibir la orden en el pin TRIG, las ondas de sonido viajan en el aire y rebotan al encontrar un objeto, el sonido de rebote es detectado por el receptor piezoeléctrico, luego el pin ECHO cambia a alto (5V) por un tiempo igual al que demoró la onda desde que fue emitida hasta que fue detectada, el tiempo del pulso ECO es medido por el microcontrolador y así se puede calcular la distancia al objeto.

La distancia se puede calcular utilizando la siguiente fórmula:
Distancia(m) = {(Tiempo del pulso ECO) * (Velocidad del sonido=340m/s)}/2

ESPECIFICACIONES TÉCNICAS:

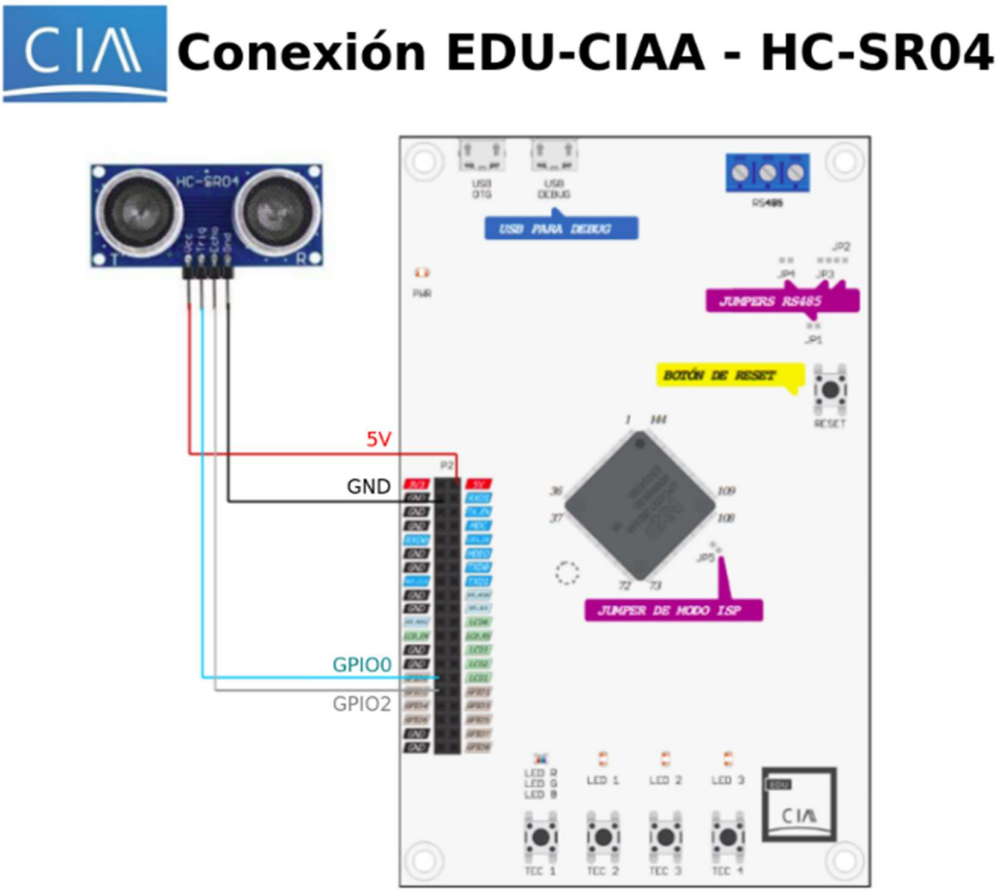
- Voltaje de Operación: 5V DC
- Corriente de reposo: < 2mA
- Corriente de trabajo: 15mA
- Rango de medición: 2cm a 450cm
- Precisión: +- 3mm
- Ángulo de apertura: 15°
- Frecuencia de ultrasonido: 40KHz
- Duración mínima del pulso de disparo TRIG (nivel TTL): 10 µS
- Duración del pulso ECO de salida (nivel TTL): 100-25000 µS
- Dimensiones: 45*20*15 mm



4 – Desarrollo

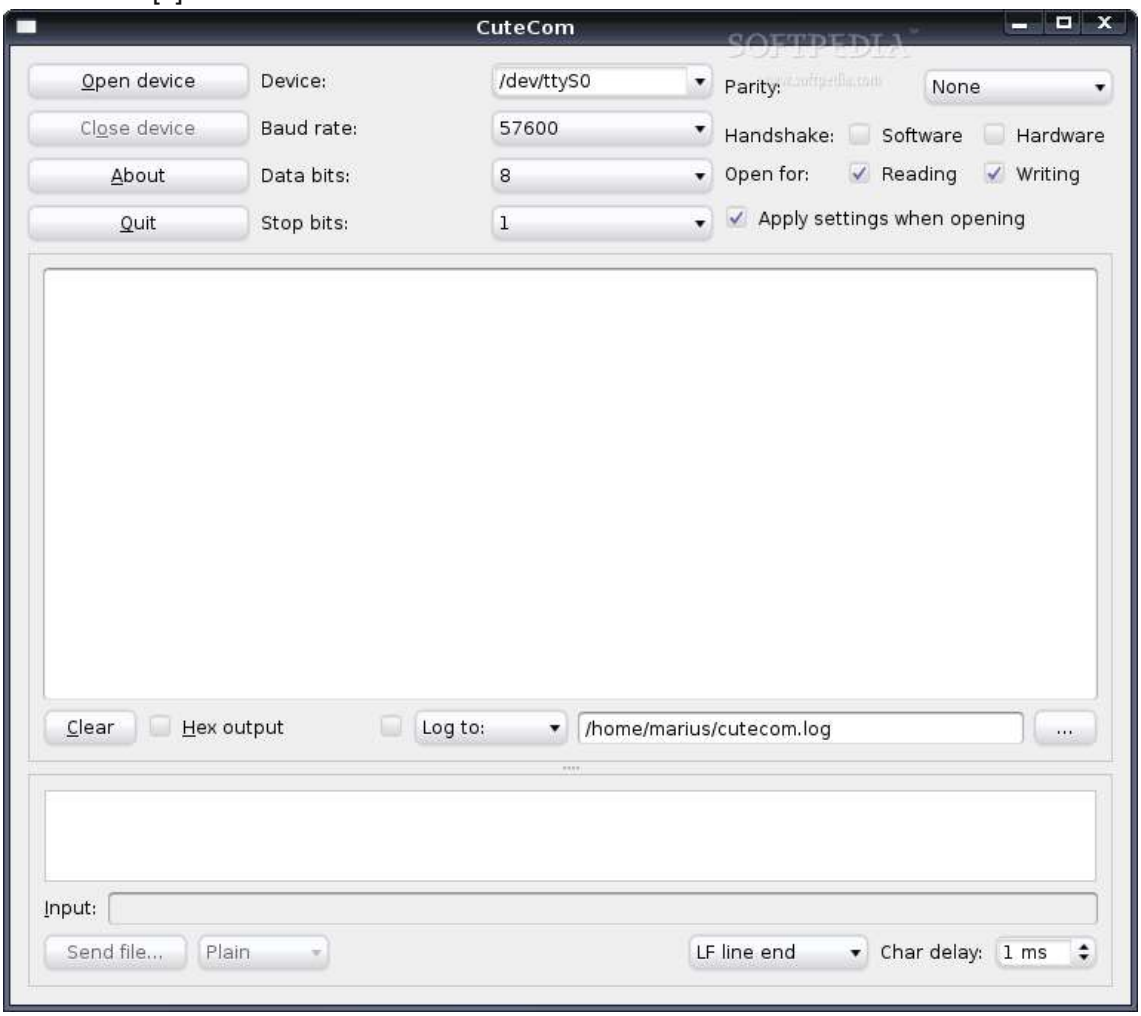
Para llevar adelante el objetivo del presente trabajo se comenzó por un trabajo de investigación, tanto de la funcionalidad de los componentes de hardware a utilizar como de las funcionalidades que brinda las sAPI.

Tras haber realizado este primer paso, comenzamos con las primeras conexiones, para ello adquirimos información que nos brinda el firmware de la CIAA en la plataforma GitHub, en donde, por un lado, nos brindó la configuración de las conexiones del sensor a la placa como también el código a implementar en el Embedded IDE para las lecturas de las distancias detectadas por el sensor [1].



Para poder monitorear las lecturas de las distancias medidas por el sensor HC-SR04 se utilizó CuteCom, él cual es un terminal gráfico. Está dirigido principalmente a hardware, desarrolladores u otras personas que necesitan un terminal para hablar con sus dispositivos.

Cuenta con una interfaz orientada a líneas en lugar de orientada a caracteres, soporte xmodem, ymodem, zmodem (requiere el paquete lrzsz) y entrada y salida hexadecimal, entre otras cosas.[2]



CuteCom

Posteriormente, pasamos a revisar las conexiones y funcionalidades del puente H, el cual nos permitirá controlar las direcciones de giro del motor CD, ya que, un motor de CD básicamente es una carga que, al alimentarlo con el voltaje necesario, realizará giros en una dirección. Todos los motores de CD tienen indicado una polaridad; es decir, una terminal positiva y una negativa. Si a un motor de CD le cambiamos la polaridad, estaremos modificando su sentido de giro.

Al realizar las conexiones a la placa, los pares de salidas (out1/out2 por un lado y out3/out4 por el otro) fueron conectados a cada motor.
El módulo contiene cuatro entradas señaladas como IN1, IN2, IN3, IN4, de las cuales nos valdremos para manejar los sentidos de giro de los motores CD.
Las entradas IN1 e IN2, son para controlar el primer par de motores CD, y las entradas IN3 e IN4 son para controlar el segundo par de motores.

Cabe mencionar que para no utilizar puertos de más de la EDU-CIAA como tampoco repetir código, lo que realizamos fue “puentear” la conexión física entre los Puentes H y los 2 pares de motores que disponemos.
Fue necesario asegurarnos de que los jumpers estén puenteados, este hace la labor de conectar con otro pin físicamente conectado a voltaje, por lo que, si el jumper de las entradas de habilitación está colocado, ambas entradas de habilitación estarán encendidas.

Entrada	Pin EDU-CIAA
IN1	GPIO1
IN2	GPIO4
IN3	GPIO5
IN4	GPIO7

En nuestro caso nos encontramos con la dificultad que 2 de las ruedas del circuito se encontraban defectuosas, por lo que tras haber realizado el analisis y seguir la lógica que el otro par de motores se debiera comportar como los 2 primeros, hicimos la desconexión de los 2 motores que proporcionaban el giro a las ruedas traseras de nuestro circuito y nos quedamos trabajando con únicamente 2 ruedas, manteniendo el mismo código el cual lo hace funcional y persistente.

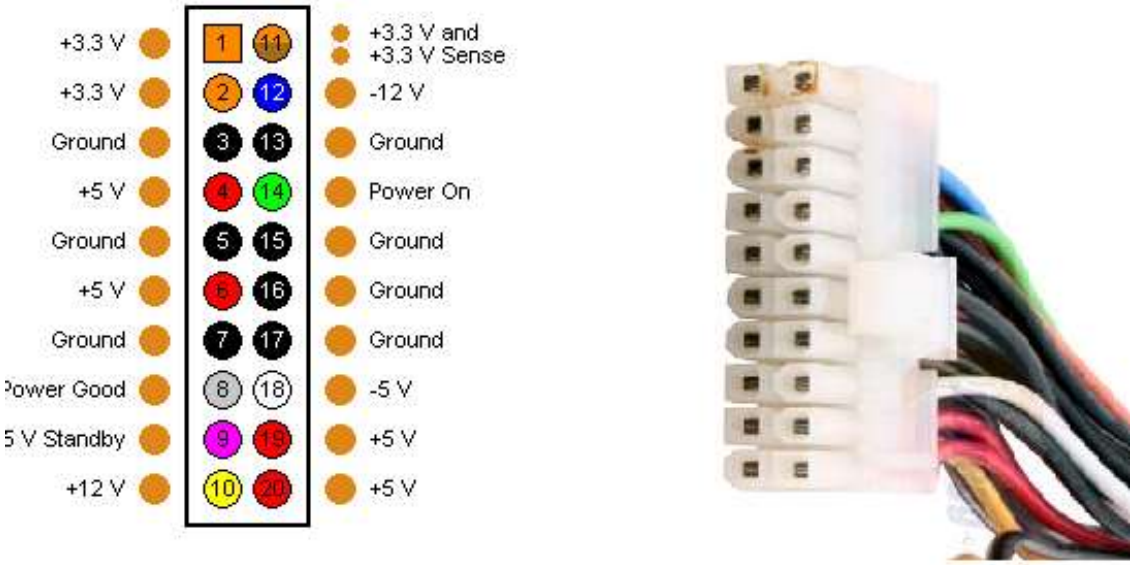
Una vez logradas las conexiones, analizamos la funcionalidad de las entradas IN para controlar el sentido de giro del motor desde nuestro código de desarrollo, HIGH lo consideramos *true* para enviarle un ‘1’ lógico y LOW lo consideramos *false* para enviarle un ‘0’ lógico a la placa. Por lo cual, las configuraciones quedaron de la siguiente manera:

```
if (distanceInCms > distanciaLimite){
// Avanzan las 2 ruedas si distanceInCms (cm) es mayor distanciaLimite (15)

        gpioWrite(GPIO4, false); //IN 2 -
        gpioWrite(GPIO1, true); //IN 1 - Rueda 1
        gpioWrite(GPIO5, true); //IN 3 -
        gpioWrite(GPIO7, false); //IN 4 - Rueda 2
    }
    else{

//Avanza 1 rueda y la otra retrocede
        gpioWrite(GPIO4, false); //IN 2
        gpioWrite(GPIO1, true); //IN 1 - Rueda 1
        gpioWrite(GPIO5, false); //IN 3
        gpioWrite(GPIO7, true); //IN 4 - Rueda 2
```

Cabe mencionar que dadas experiencias anteriores tras afirmar que los 5V provenientes del puerto USB de la PC no son suficientes, se nos sugirió la utilización de una fuente externa ATX.,
Lo primero que fue necesario hacer es puentear la fuente para que encienda luego de conectar la misma a 220V. Para ello se debe hacer un puente entre la conexión de color verde representada por el pin 14 (Power On) y cualquier pin de tierra, en nuestro caso el pin 13 que es un pin de GND.



Puente Fuente PC

Posteriormente, con ayuda de un tester nos aseguramos de que la salida de la fuente sea efectivamente de 5v para evitar posibles complicaciones al alimentar la placa. Para realizar las conexiones a las respectivas entradas input y GND utilizamos un protoboard, lo cual nos facilitó dicha conexión.

5 - Implementación y resultados

Al concluir con la conexión de la alimentación a la placa, realizamos las pruebas necesarias que nos permitieron comprobar que el motivo por el cual las ruedas del automóvil giraban a menor velocidad que la deseada estaba relacionado con la alimentación recibida.
El último detalle que no habíamos tenido en cuenta fue evitar los delays en el código, ya que la presencia de uno de ellos evitaba la rápida reacción del vehículo al detectar una distancia menor/mayor a 15 cm, ello es muy importante, debido a que si fuese un automóvil real con un delay de por medio se ocasiona un accidente.
Finalmente, el esquema de hardware y las implementaciones obtenidas han sido las siguientes:

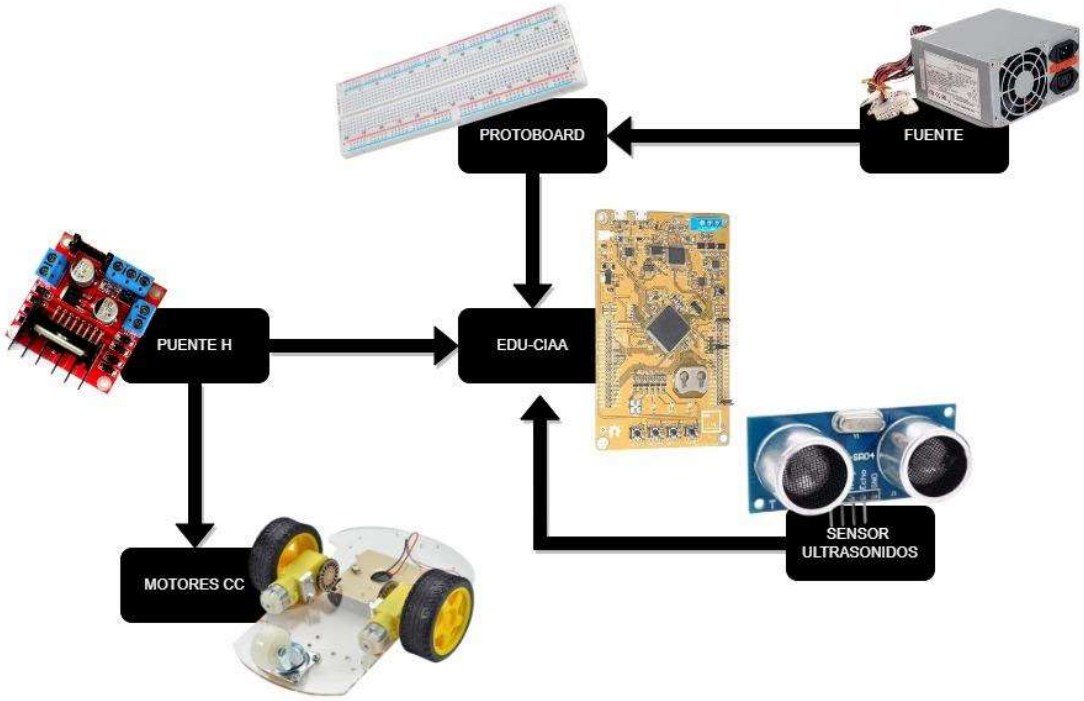
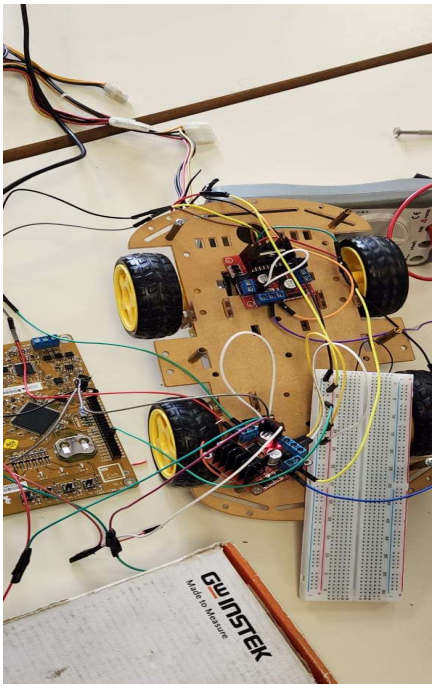
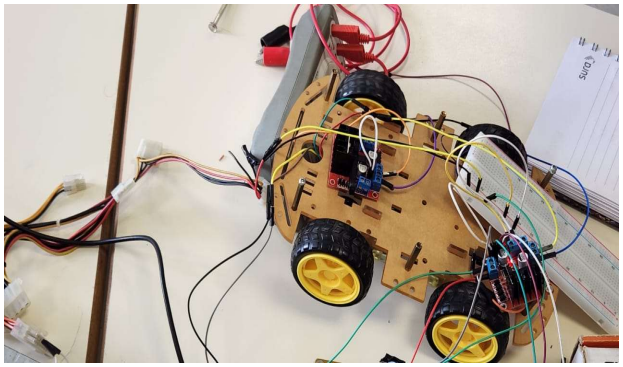
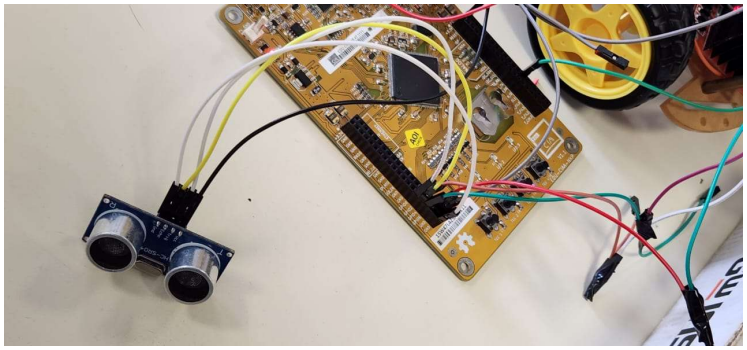


Diagrama en bloques de los componentes

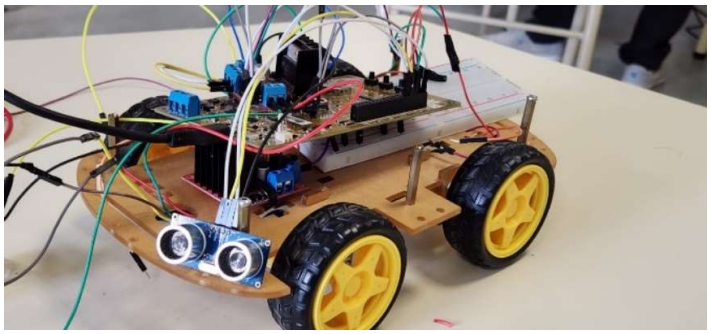




Interconexionado de los componentes



Conexión Sensor de Ultrasonido – EDU-CIAA



Circuito final montado

6 - Conclusiones

Finalmente, logramos concluir el trabajo de laboratorio y cumplir con el objetivo propuesto, el automóvil funcionó correctamente, al detectar una distancia menor a los 15 cm realiza un giro hacia la izquierda para evitar chocarse con un posible obstáculo, en condición contraria continúa avanzando. Han sido de gran utilidad las herramientas a las que hemos podido acceder relacionadas a la EDU-CIAA, desde la documentación de esta del lado del hardware hasta los modelos de desarrollo de la sAPI del lado del software.

Sumado a lo anterior podemos concluir que el poder llevar, todos los conceptos vistos en la teoría a la práctica y ver que se condicen es la mejor forma de comprenderlo para luego también poder explicarlo en detalle.

7- Posibles mejoras a desarrollar

Control de velocidad

Para tal caso, si deseamos variar la velocidad del motor, era necesario controlar la entrada de habilitación, removiendo el jumper correspondiente y conectando un cable a alguna salida de tipo PWM de EDU-CIAA [4].
Será necesario agregar en el código una salida llamada enableA para manipular la entrada ENA del módulo. Esta salida tiene que ser de tipo PWM para cambiar el ancho del pulso de la señal que activará al transistor mediante un *pwmWrite(salida, valor).* El valor que utiliza el comparador para determinar el ciclo útil no se expresa en tanto por ciento. En su lugar se expresa como un número entre 0 y 255 porque este es el valor máximo que puede alcanzar el contador (8 bits). De esta forma, un valor de 255 de ciclo útil realmente genera un ciclo de trabajo del 100%, mientras que con un valor de 127 se logra un ciclo de trabajo del 50%.

Implementación de un Servomotor

Un servomotor es un actuador rotativo que permite un control preciso en términos de posición angular, aceleración y velocidad, capacidades que un motor normal no tiene. En definitiva, utiliza un motor normal y lo combina con un sensor para la retroalimentación de posición. Por lo cual, nos permitirá realizar un paneo de 90 grados alrededor del automóvil para determinar el espacio libre de obstáculos. A continuación, podemos observar la conexión que debemos hacer a la placa [1]

Ejemplo con servomotor Tower Pro SG90

Especificaciones:

- Peso: 9g.
- Tamaño: 22,2 x 11,8 x 31 mm aprox.
- Stall torque: 1,8 kgf cm.
- Velocidad de operación: 0,1 s / 60 grados.
- Voltaje de operación: 4.8 V (~5V).
- Dead band width: 10µs.
- Rango de temperatura: 0ºC a 55ºC.

Conexión con EDU-CIAA-NXP:

El auto podría tener ubicado el servomotor en la parte delantera, este servo tendrá sujeto a él la placa de ultrasonidos, con lo que en primer lugar daremos la orden al servo de girar a la derecha, utilizando la función que nos brinda la sAPI servoWrite(), y tomar la distancia a la derecha, seguidamente daremos la orden al servo de girar a la izquierda para que mida la distancia libre que tenemos. Una vez recuperada la posición central y original del servo, el robot evalúa qué distancia es mayor sin obstáculos en función del resultado daremos la orden al robot para que gire en hacia un lado u otro.

- Pilas

Una posible mejora al proyecto sería colocarle una pila de 9v para que alimente a la placa y a todos los componentes. Con esto se lograría una mayor libertad de movimiento al dispositivo y no tener que estar conectado a la fuente con un cable demasiado largo, lo cual hace engorroso el desplazamiento del auto.

Buzzer

Un zumbador o mejor conocido como buzzer es un pequeño transductor capaz de convertir la energía eléctrica en sonido. Por lo cual, mediante el buzzer se emitirá una señal sonora para indicar que el automóvil se aproxima a un obstáculo.

8 - Bibliografía

[1] Eric Pernia(2019), hcsr04_ultrasonic_sensor, Recurso obtenido de https://github.com/ciaa/firmware_v3/tree/master/examples/c/sapi/hcsr04_ultrasonic_sensor consultado el día 22/09/2022.

[2] Alexander Neundor,Roman I Khimov(2019), Recurso obtenido de https://manpages.ubuntu.com/manpages/trusty/man1/cutecom.1.html consultado el día 22/09/2022.

[3] AbrahamG(2021), Recurso obtenido de https://www.automatizacionparatodos.com/puente-h-arduino/ consultado el día 22/09/2022.

[4] Eric Pernia(2016), Recurso obtenido de http://www.proyecto-ciaa.com.ar/devwiki/lib/exe/fetch.php?media=desarrollo:edu-ciaa:edu-ciaa-nxp_pinout_a4_v4r2_es.pdf consultado el día 22/09/2022.

[5] Sebastián Luchetti (2021), Recurso obtenido de https://tech.tribalyte.eu/blog-sistema-embebido-caracteristicas consultado el día 22/09/2022.

[6]Ingeniería Mecafenix(2017), Recurso obtenido de https://www.ingmecafenix.com/electronica/puente-h-control-motores/ consultado el día 22/09/2022.

[7] Naylamp Mechatronics SAC (2021), Recurso obtenido de https://naylampmechatronics.com/sensores-proximidad/10-sensor-ultrasonido-hc-sr04.html consultado el día 22/09/2022.

[8] efilomena (2017), Recurso obtenido de http://www.proyecto-ciaa.com.ar/devwiki/doku.php?id=desarrollo:edu-ciaa:edu-ciaa-nxp consultado el día 22/09/2022.

Apéndice

Código completo del dispositivo:

```
*============================================================================
 * Autor:
 * Licencia:
 * Fecha:
 *============================================================================*
/

// Inclusiones

#include "app.h"         // <= Su propia cabecera
#include "sapi.h"        // <= Biblioteca sAPI

/* con #define "defino" una variable llamada SalidaGPIO y le asigno la funcion
GPIO_OUTPUT, asi en la linea 30 - 33 si bien repito la variable
SalidaGPIO 4 veces, la funcion GPIO_OUTPUT la repito una vez */
#define SALIDAGPIO GPIO_OUTPUT

const int distanciaLimite =  15;
delay_t delayLed;
// FUNCION PRINCIPAL, PUNTO DE ENTRADA AL PROGRAMA LUEGO DE ENCENDIDO O RESET.
int main( void )
{
   // ---------- CONFIGURACIONES ------------------------------

   // Inicializar y configurar la plataforma
   boardConfig(); // Empezamos a configurar la placa
    uartConfig( UART_USB, 115200 ); // Inicializar periferico UART_USB
    delayConfig(&delayLed, 500);
    // Inicializar el sensor ultrasonico HC-SR04
    ultrasonicSensorConfig( ULTRASONIC_SENSOR_0, ULTRASONIC_SENSOR_ENABLE );

    //configurar PUENTE H */
    gpioConfig(GPIO1, SALIDAGPIO);
    gpioConfig(GPIO4, SALIDAGPIO);
    gpioConfig(GPIO5, SALIDAGPIO);
    gpioConfig(GPIO7, SALIDAGPIO);

    delay(100); // Retardo inicial de 100 ms

   // Variables enteras de 32 bits para guardar la distancia en pulgadas
   // y centimetros
    uint32_t distanceInCms;
```

```
   // ---------- REPETIR POR SIEMPRE --------------------------
   while( TRUE ) {
       if(delayRead(&delayLed)){
           gpioToggle(LEDR);
           }
    // Obtenemos la distancia actual medida por el sensor en centimetros
        distanceInCms = ultrasonicSensorGetDistance(ULTRASONIC_SENSOR_0, CM);

        /*gpioWrite(GPIO4, false); //IN 2
        gpioWrite(GPIO1, true); //IN 1
        gpioWrite(GPIO5, true); //IN 3
        gpioWrite(GPIO7, false); //IN 4 */

       if (distanceInCms > distanciaLimite){
           // Avanzan las 4 ruedas si distanceInCms (cm) es mayor
distanciaLimite (15)
           gpioWrite(GPIO4, false); //IN 2
           gpioWrite(GPIO1, true); //IN 1 - Rueda 1
           gpioWrite(GPIO5, true); //IN 3
           gpioWrite(GPIO7, false); //IN 4 - Rueda 2
       }
       else{
           // Avanzan dos ruedas y dos retroceden
           gpioWrite(GPIO4, false); //IN 2
           gpioWrite(GPIO1, true); //IN 1 - Rueda 1
           gpioWrite(GPIO5, false); //IN 3
           gpioWrite(GPIO7, true); //IN 4 - Rueda 2
       }

       // Reportamos las distancias medidas en centímetros y pulgadas
       printf( "Distance: %d cm\".\r\n", distanceInCms);

   }

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