

Taller de proyecto II

Informe de Avance 2

Proyecto 5 - Auto en Línea Recta



8 de noviembre del 2018

Integrantes

- Ailigo Oriana – 1202/8
- Aguilar Sergio – 00765/7

1- Propuesta Original del Proyecto

Se dispone de un auto y las comunicaciones necesarias con la PC que son llevadas a cabo utilizando la red WiFi cuyo punto de acceso es el ESP8266, para mover el auto.

La problemática que se aborda en el proyecto es el control de dirección sin desviaciones por diferencias y fallas de funcionamiento en la reacción de los motores de CC.

NOTA: en el debug del sistema no se realizan cambios. Este auto debe ir derecho, ya que no lo hace por un problema en los motores. Por lo tanto, se deberá mantener el equilibrio del mismo.

El objetivo del proyecto:

- conseguir que el auto siga derecho, sin desviarse.
- mostrar en la interfaz web la desviación medida en RPM y lo que significa en distancias (metros o centímetros), para poder medir en la propia trayectoria del auto en el piso. (agregado en este informe)

Para ello se usarán los encoders así se podrá medir velocidad de giro, realizar comparaciones entre ambos y reducir el margen de diferencia de velocidad entre ambas ruedas para mantener la dirección del auto en línea recta. Aquí utilizaremos el algoritmo PID que se describiera en el transcurso del informe.

Esquema

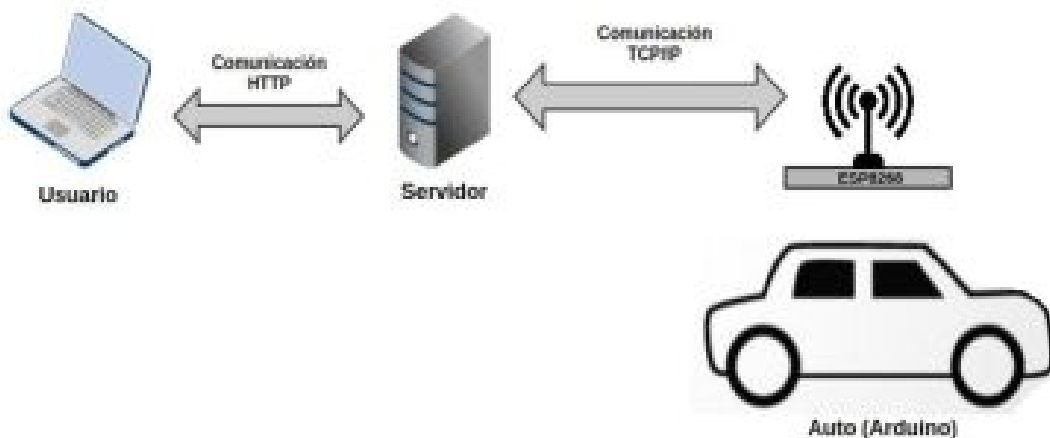


Figura 1: Esquema del proyecto

Desde el punto de vista del usuario, el mismo deberá acceder a una aplicación web donde encontrará los comandos necesarios para hacer funcionar el vehículo,
La mencionada aplicación será montada sobre un servidor HTTP el cual se comunicará con el vehículo mediante el protocolo TCP/IP.
Para llevar a cabo este objetivo se le incorporará al vehículo un módulo wifi ESP8266.

Se implementa en cada motor un **sistema de control PI de lazo cerrado** debido al diseño de "dirección diferencial" que se muestra en el medio de la siguiente imagen. Aquí, uno tiene que constantemente supervisar y actualizar las dos velocidades del motor para conducir en línea recta. Por lo tanto, se requiere una sincronización de las dos velocidades del motor.

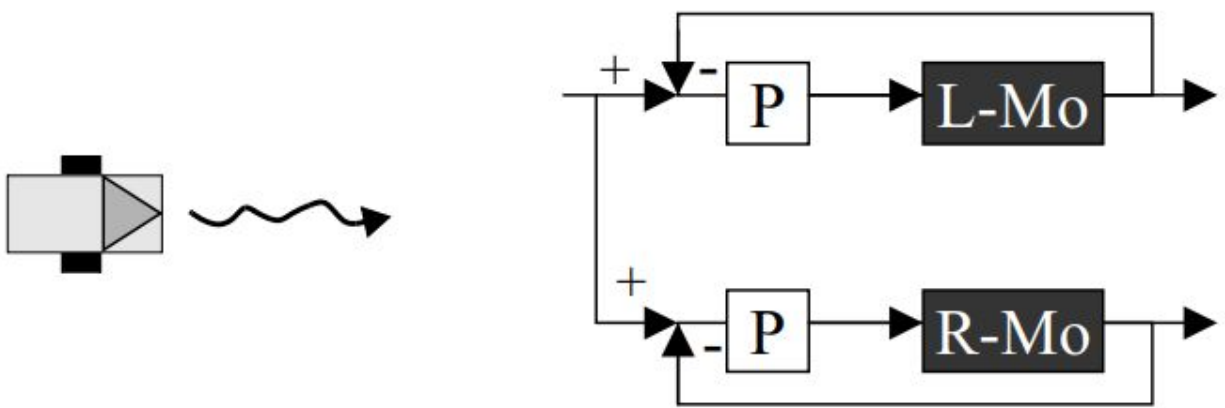


Figura 2:sistema de control PI de lazo cerrado

Hay dos bucles de control separados para el motor izquierdo y derecho, cada uno involucrando control de retroalimentación a través de un controlador P distinto. La velocidad de avance deseada es suministrado a ambos controladores. Desafortunadamente, este diseño no producirá perfecta línea recta de conducción. Aunque cada motor está controlado en sí mismo, no hay control de cualquier ligera discrepancia de velocidad entre los dos motores. Tal configuración provoca que el auto conduzca en una línea ondulada, pero no en línea recta.

2- Dispositivos

a) Disponibles:

1- Se realizaron pruebas con 1 de los motores conectados con el arduino UNO y mediante un sketch se muestra en código el funcionamiento del mismo. El cual funcionó con éxito. Por lo tanto, solo se copio la implementación del mismo para el otro motor.

En el anexo A se observa en más detalle la prueba del motor.

2- Por otro lado, se verificó individualmente el funcionamiento del módulo wifi mediante los comandos AT primero por consola (puerto serial). Después de tantos intentos, nos respondió con un "ok" en pantalla. Luego lo implementamos en código en el sketch para que nos mostrará en pantalla el nombre de la red y así poder acceder al mismo, colocando los caracteres correspondientes en consola.

En el anexo B se observa en más detalle la prueba del módulo wifi.

3- Se realizó una prueba individual con el encoder donde se veía que funcionaba mediante el giro de la rueda de forma manual.

En el anexo C se observa en más detalle la prueba del encoder.

4- Se implementaron librerías para disminuir la longitud de código y así poder usar las funciones que brindan las mismas: una para el encoder ("encoder"), para el control del PID y otra para los motores ("motor"). Además de implementar interrupciones por hardware las cuales funcionan si el encoder realizó una vuelta completa.

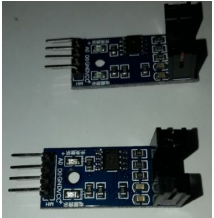
En el anexo D se observa en más detalle las 3 librerías.

5- Por último, se realizaron pruebas con el wifi junto con las librería del motor (L9110.h), utilizando una batería externa que alimenta el arduino, motores y el módulo wifi.

Para más información ver el anexo E.

b) Presupuesto

Dispositivos/Módulos/ Materiales	Descripción/Función	Precio
Chasis de 3 ruedas(incluye dos motores de cc) 	Chasis plástico del autito, con dos ruedas a motor de corriente continua y una libre.	\$640. Disponible en: https://articulo.mercadolibre.com .ar/MLA-686253561-kit-chasis-s mart-car-auto-robot-2-ruedas-ar duino-_J Fecha:21/10/18
Doble Puente H L9110 	Es el circuito electrónico que nos permite el control de los motores, de modo tal que puedan avanzar y retroceder.	\$90. Disponible en: https://articulo.mercadolibre.com .ar/MLA-748763522-doble-puent e-h-driver-l9110s-motor-dc-ardu ino-_JM Fecha:21/10/18
Módulo WiFi ESP8266 	Módulo que implementa el stack TCP/IP, el cual nos va a permitir la conexión con el servidor WEB	\$221. Disponible en: https://articulo.mercadolibre.com .ar/MLA-620320780-modulo-wifi -esp8266-serie-stack-tcp-ip-ante na-lwip-apsta-_J JM Fecha:21/10/18
Arduino UNO 	Kit de desarrollo con microcontrolador sobre el cual corre el software que controla al autito, los sensores y el módulo WiFi	\$279. Disponible en: https://articulo.mercadolibre.com .ar/MLA-609315098-arduino-uno -atmega328-compatible-usb-bas ado-en-ch340-nubneo-_JM JM Fecha:21/10/18
Power bank 	Batería externa con 5v de salida suficiente para alimentar todos los componentes. Además posee una Batería de Ion-Litio con 5200 mAh	\$3990. Disponible en: https://www.linio.cl/p/bateri-a-ext erna-celular-power-bank-5200m ah-negro-trmq1j Fecha:3/11/18

<p>Encoder LM393</p> 	<p>El sensor tiene un emisor y un receptor infrarrojo que detecta la ranura de la rueda. Se utilizará para medir la velocidad de las ruedas.</p>	<p>79 \$ cada uno. Disponible en: https://articulo.mercadolibre.com.ar/MLA-705949046-sensor-velocidad-giro-rueda-pulsos-tacometro-arduino-nubbeo-JM Fecha:29/10/18</p>
--	--	--

El presupuesto total es 5299 \$

3- Descripción y Documentación General del Proyecto

Descripción de Hardware y Conexiones

Conexionado doble puente H:

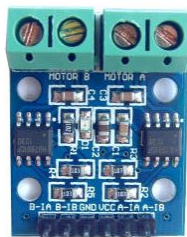
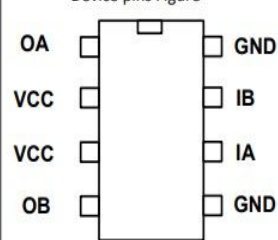


Puente H L 9110

Pin definitions:

No.	Symbol	Function
1	OA	A road output pin
2	VCC	Supply Voltage
3	VCC	Supply Voltage
4	OB	B output pin
5	GND	Ground
6	IA	A road input pin
7	IB	B input pin
8	GND	Ground

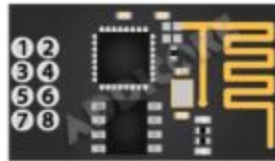
Device pins Figure



PIN PUENTE H	PIN DE CONEXIÓN Arduino UNO
B-IA	5
B-IB	4
GND	GND
VCC	VCC = 5v
A-IA	6
A-IB	7

Tabla 1: Pines del puente H L9110 en el Arduino UNO

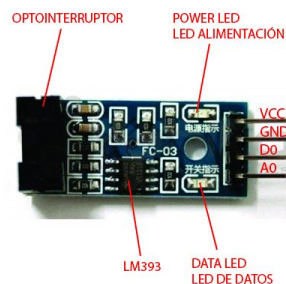
Conexionado ESP8266:



Número	Nombre Pin ESP8266	Pin de conexión(Dispositivo)
1	RX	Pin 0(Arduino uno)
2	GND	GND(Arduino uno)
3	CH	3.3V(Arduino uno)
4	GPIO2	Sin conexión
5	RESET	Sin conexión
6	GPIO1	Sin conexión
7	VCC	3.3V(Arduino uno)
8	TX	Pin 1(Arduino uno)

Tabla 2: Pines del ESP8266

Conexionado Sensor de Velocidad:



Pines del sensor de velocidad 1	Pines del arduino
VCC	5v
GND	GND
D0	2

Tabla 3: Pines del sensor de velocidad 1

Pines del sensor de velocidad 1	Pines del arduino
VCC	5v
GND	GND
D0	3

Tabla 4: Pines del sensor de velocidad 2

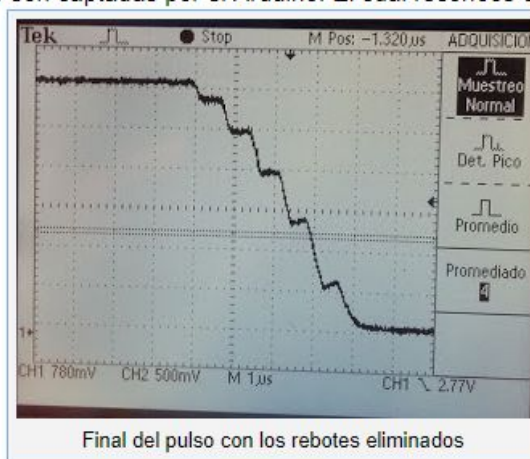
Se agregó en este nuevo informe de avance la siguiente nota:

NOTA : Para el uso de los encoder se implementaron capacitores.

Se tuvieron problemas a la hora de leer los pulsos digitales que genera el comparador LM-393. Uno de estos problemas es que el Arduino UNO lee más pulsos de los que se generan realmente (lee del orden de 4 veces más pulsos). Se realizaron búsquedas de información sobre este problema en internet y no se ha encontrado lo suficiente. Por otro lado, se notó que el sensor es muy sensible a las interferencias que se puede introducir en los pines VCC y GND. Por lo tanto, si alimentamos el sensor desde el propio Arduino a 3,3V o 5V, el regulador de tensión del propio Arduino introduce corrientes parásitas en el sensor. Produciendo que este no funcione correctamente.

Pruebas usando capacitores y encoders con tensión de 5 V:

En la fotografía siguiente se aprecia la rampa de bajada de un pulso del encoder con el condensador ya soldado. Podemos ver que los rebotes han desaparecido, pero aún se aprecia unos pequeños escalones en la señal. Estas pequeñas oscilaciones no son captadas por el Arduino. El cual reconoce el pulso correctamente.



Esta solución es muy buena, ya que no se envían al Arduino falsas señales y el programa no tiene que perder el tiempo verificando si la señal es buena o mala. De esta manera solo se activa la interrupción del Arduino cuando la señal es correcta.

En la siguiente imagen se aprecia el conexionado completo del proyecto (agregado para este nuevo informe).

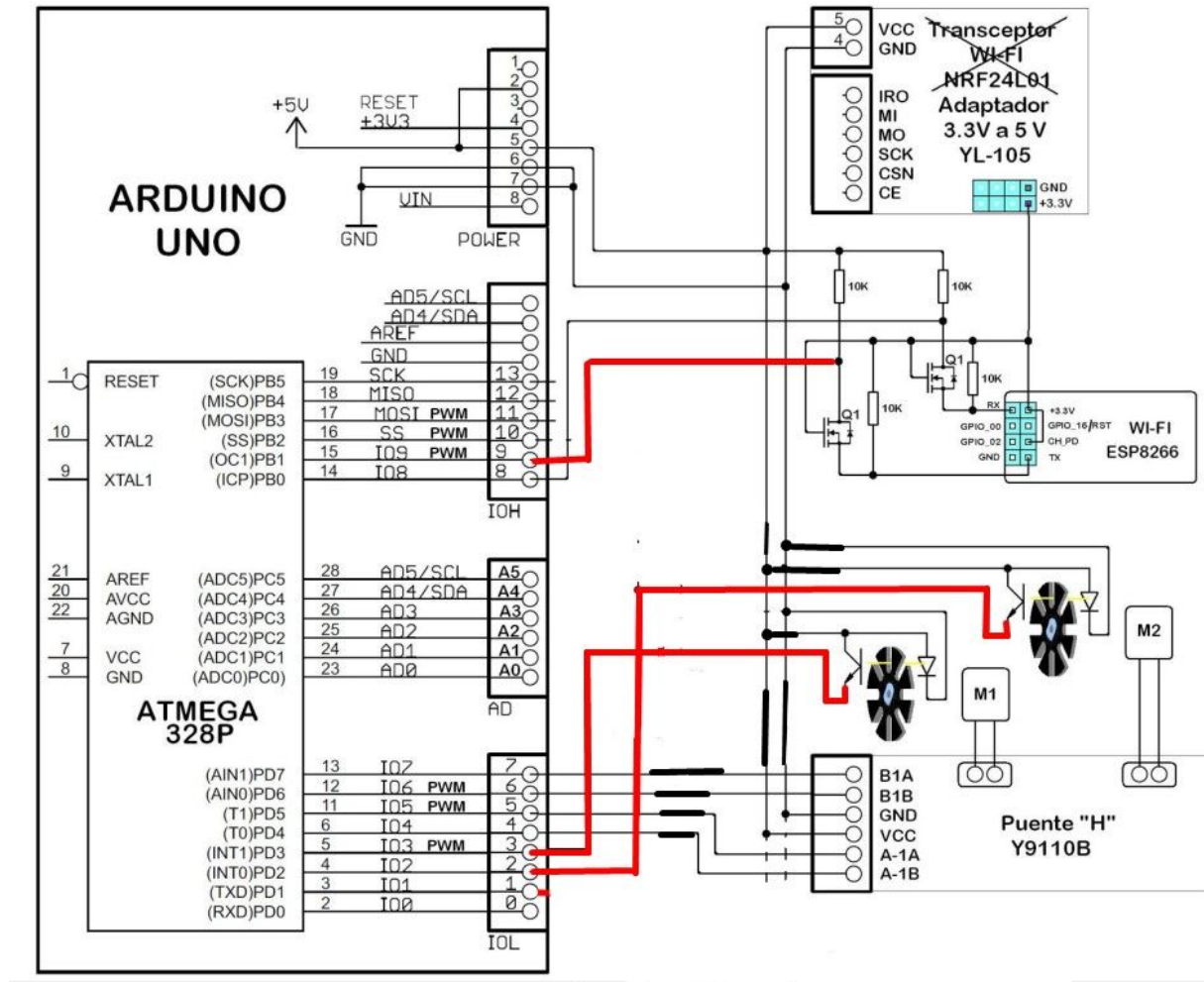


Figura 3: Esquema gráfico del proyecto

Con respecto al vehículo:

- Al encender el vehículo el mismo se encarga en primera instancia de la configuración de los pines necesarios para la utilización de los dos sensores de velocidad, del módulo wifi(ESP8266-01) y del puente H (L9110).

- Luego crea la red -mediante el módulo wifi- llamada autito Lineal la cual no posee ningún tipo de seguridad y puede ser accedida por cualquier dispositivo (máximo de 5). Además se monta un servidor TCP por el cual se recibirá “órdenes” (al ejecutarse, gracias a la librería PID_v1.h, se podrá visualizar el desvío producido por las ruedas) por parte de usuario. El mismo se encuentra en el puerto 666 de la ip 192.168.4.1, pues la dirección IP del vehículo no cambia.
- Para las tareas antes mencionadas se utilizan las siguientes funciones:
 - Calculo() La librería Encoder.
 - ESP8266 wifi() de la librería ESP8266.
 - wifi.Configuration() de la librería ESP8266.
 - L9110(int aia,int aib,int bia,int bib) de la librería L9110.
 - PID(double*, double*, double*, double, double, double, int) de la librería PID_v1. (se agregó en este nuevo informe). Para más información ver en el ANEXO D.

Una vez concluida la etapa de configuración, que aproximadamente tarda 3 segundos, el vehículo se encuentra en condiciones de funcionar.

En cuanto al funcionamiento, se diseñó el software sin la utilización de un sistema operativo, por ello se aprovechó la estructura LOOP propia de arduino para crear un lazo de control. El mismo se puede observar en la siguiente imagen. (se agregó para este nuevo informe)

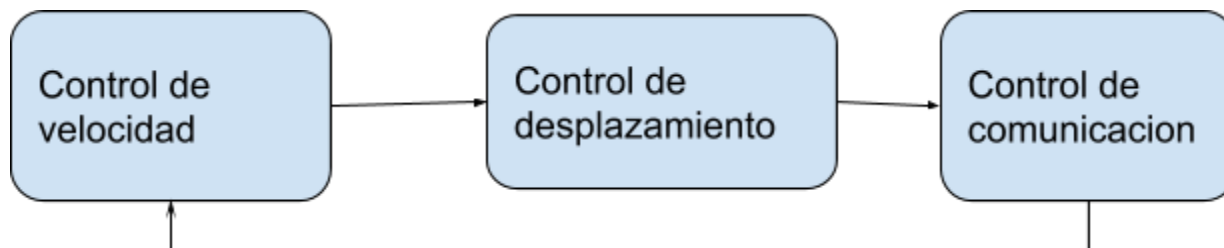


figura 4: Estructura del loop

- Control de velocidad: Se encarga de la realización de mediciones de velocidad y su respectivo almacenaje en variables globales del arduino.

Para las tareas del Control de distancia se utilizan las siguientes funciones:

- Encoder.calcular();
- Encoder.getRPM();
- Control de Desplazamiento: Se encarga de leer las variables de desplazamiento que determinan el mismo, ellas son 3:

- direction: 0 si la dirección no está definida, 1 si la dirección es hacia avanzar, 2 si la dirección es retroceder.
- speed: 0 velocidad nula (no se mueve), 1 velocidad intermedia, 2 velocidad alta.
- turn: 0 no se dobla, 1 se dobla a la izquierda, 2 se dobla a la derecha.

Es importante aclarar lo siguiente:

1. Para avanzar en línea recta se puede utilizar cualquiera de las 2 velocidades.
2. Para retroceder y doblar, la velocidad siempre es la 1.
3. Solo se puede doblar hacia adelante.

Una vez leídas e interpretadas las variables envía al “puente H” (L298N) los valores de PWM que correspondan para realizar la maniobra.

Para las tareas del Control de Desplazamiento se utilizan las siguientes funciones:

- “parar” de la librería L9110.
- “avanzar” de la librería L9110.
- “Izquierda” de la librería L9110.
- “Derecha” de la librería L9110.
- “atrás” de la librería L9110
- Control de Comunicación: En primera instancia verifica si ha recibido algún mensaje en el servidor TCP (ESP8266), en caso afirmativo procede a su lectura. Si el mensaje posee largo un byte, se interpreta que el mismo corresponde al pedido de ver velocidades. Si el mensaje posee tres byte de largo, se asume que el mismo corresponde a una “orden” de movimiento.

Los valores se almacenan en las variables de desplazamiento para que “Control de Desplazamiento” las ejecute.

En el caso de que se envíe la orden “212” que se correspondería con retroceder doblando a la derecha (no se encuentra permitido) el “control de Desplazamiento” no será capaz de interpretarla y el vehículo no se moverá

Para las tareas del Control de Comunicación se utilizan las siguientes funciones:

- “Receive_data” de la librería ESP8266.
- “data_send” de la librería ESP8266.
- Una vista para el control y visualización de la desviación medida en RPM: Utiliza queries de ajax para el control del autito y para la consulta y actualización de los valores. Dentro de los controles tenemos Marcha-, Frenar, Marcha+, Avanzar, Izquierda, Derecha y Retroceder. En la visualización de la desviación estará los RPM, y también expresado en distancia.

A- Alimentación del dispositivo/placa de desarrollo:

Su alimentación es a través del puerto serie. Para ello se incorporó una batería externa de 5V.

B- E/S de la placa de desarrollo con el exterior excepto PC:

La placa de desarrollo adquirirá información del exterior mediante los sensores de velocidad (encoders), gracias a ellos se podrá calcular la velocidad que posee cada rueda del auto. Más específicamente, cada un cierto tiempo dado por la interrupción provocada por un evento en el encoder(transición) este fue capturado a través de la función `millis()`, a definir, el microcontrolador obtiene el valor de los sensores de velocidad, detectando las ranuras de las ruedas y así detectar una vuelta completa de las mismas. Por otro lado, el microcontrolador también comanda el avance/retroceso, girar a izquierda o derecha, mediante el control de los motores de corriente continua.

El control de aceleración/desaceleración de los motores se realiza mediante el uso de un puente H, previamente presupuestado.

C- Comunicaciones de la placa de desarrollo con la PC:

Para la comunicación con la PC se montará al arduino un módulo Wifi ESP8266 por el cual se podrá mover al auto, la comunicación se hará mediante el protocolo TCP/IP. Cabe aclarar que la comunicación en sentido PC-vehículo consiste en el envío de los comandos de movimiento, y en el sentido inverso, vehículo-PC consiste en el envío de mediciones de los sensores de velocidad.

D- Sistema/interfaz web:

El sistema web se basará en un servidor HTTP el cual brindará al usuario del sistema una interfaz gráfica por la cual podrá hacer que el vehículo se mueva. Para ser más precisos con la aplicación, la misma contendrá las siguientes funcionalidades:

-Acción de avance.

-Selección de marcha: Se elegirá el sentido de la marcha y la velocidad en caso que sea hacia delante.

-Otra funcionalidad que se agregó para este informe de avance 2 es un botón para mostrar en la interfaz web la desviación medida en RPM y lo que significa en distancias (metros o centímetros), para poder medir en la propia trayectoria del auto en el piso.

E- Infraestructura de software propuesta para la PC:

Página que está cargada con la interfaz donde se realiza la comunicación. No está implementada.

Resumen de los protocolos de comunicación utilizados al momento:

*Protocolo TCP/IP: Realiza la comunicación Vehículo-PC.

El **Protocolo de Control de Transmisión (TCP)** permite a dos anfitriones (pc y módulo esp8266) establecer una conexión e intercambiar datos. El TCP garantiza la entrega de datos, es decir, que los datos no se pierdan durante la transmisión y también garantiza que los paquetes sean entregados en el mismo orden en el cual fueron enviados.

El **Protocolo de Internet (IP)** utiliza direcciones que son series de cuatro números octetos (byte) con un formato de punto decimal, por ejemplo: 69.5.163.

4- Guía de Instalación del Ambiente de Desarrollo y Documentación

a) Ambiente de desarrollo

El **software de desarrollo** que utilizamos es IDE .

Instalación del Arduino IDE:

Se accede a la solapa “software” del sitio oficial de Arduino.
(<https://www.arduino.cc/en/Main/Software>). Luego se selecciona el SO operativo de nuestra computadora y se procede con la descarga. Finalizada la descarga se debe instalar de la manera que corresponde con cada SO, en el caso de linux debemos descomprimir el archivo y mediante el intérprete de comandos ejecutar “./install.sh”

Lenguajes utilizados:

Las librerías en arduino se desarrollan en C++, para lo cual no es necesario la instalación de ningún paquete.

Drivers necesarios:

Para el motor la librería ControlMotor

Para el encoder utilizamos función encoder()
(implementada en el archivo autito.ino)

Para el módulo wifi utilizamos la librería **ESP8266**

Instalación de las librerías necesarias:

Las librerías utilizadas por el vehículo se adjuntan al presente informe en formato “zip”, para su instalación se debe descomprimir todas ellas, luego copiar cada carpeta (ControlMotor, ESP8266) dentro del directorio de instalación de arduino dentro de la carpeta “libraries”.

Instalación de las sistema en el arduino:

El programa a cargar en el arduino, se adjunta al presente informe como “autito.ino”.

Para su instalación debemos clicar en él y una vez abierto el IDE de arduino clicar en subir. Al cabo de unos segundos el programa se encuentra descargado.

Copia/Instalación de Código:

Las funciones que controlan cada hardware utilizado en el vehículo fueron detalladas con anterioridad, sin embargo para afianzar la idea de la estructura de archivos se incorpora la siguiente imagen:

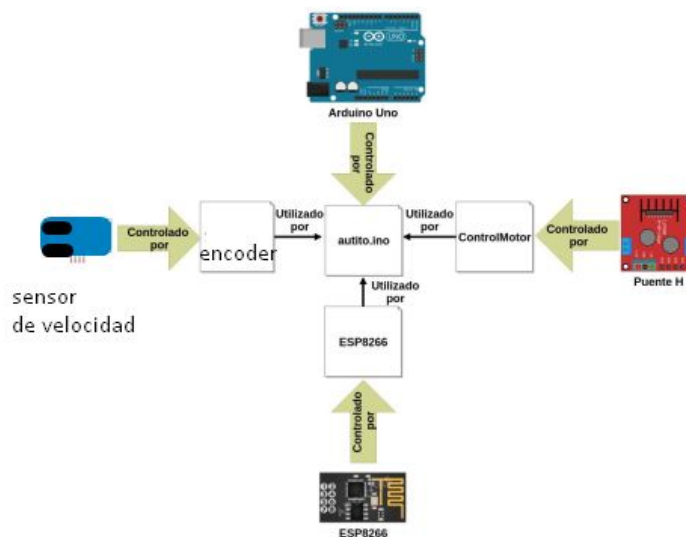


Figura 5: Relación de componentes

b) Documentación de Código

Fecha: 25/09/18

A modo de resumen, dentro del hardware utilizado encontramos:

- El módulo WiFi ESP8266, encargado de levantar la red “SistemaAntichoque” y un servidor TCP/IP, contra el cual se conecta y comunica el subsistema web.
- El módulo Doble Puente H, encargado del manejo de los motores.
- Dos módulos sensores de velocidad LM393, encargados de medir la velocidad del vehículo.
- Una placa de desarrollo, arduino UNO, encargada de manejar todos los módulos anteriormente detallados, a través de las librerías ESP8266, ControlMotor.

1- Esta parte del código maneja los dos motores:

void derecha(int velocidad): Gira a la derecha el vehículo con velocidad “velocidad”.

void izquierda(int velocidad): Gira a la izquierda el vehículo con velocidad “velocidad”.

void adelante(int velocidad): Permite avanzar el autito con una cierta velocidad.

void atras (int velocidad): Permite retroceder el autito con una cierta velocidad.

void parar(int velocidad): Permite frenar/detener el autito con una cierta velocidad.

```
void adelante(int speed)
{
    analogWrite(AIA, speed);
    analogWrite(AIB, 0);
    analogWrite(BIA, speed);
    analogWrite(BIB, 0);
}
```

```
void atras(int speed)
{
    analogWrite(AIA, 0);
    analogWrite(AIB, speed);
    analogWrite(BIA, 0);
    analogWrite(BIB, speed);
}
```

```
void izquierda(int speed)
{
    analogWrite(AIA, speed);
    analogWrite(AIB, 0);
    analogWrite(BIA, 0);
    analogWrite(BIB, speed);
}
```

```
void derecha(int speed)
{
    analogWrite(AIA, 0);
    analogWrite(AIB, speed);
    analogWrite(BIA, speed);
    analogWrite(BIB, 0);
}
```

```
void parar()
{
    analogWrite(AIA, 0);
    analogWrite(AIB, 0);
    analogWrite(BIA, 0);
    analogWrite(BIB, 0);
}
```


Fecha: 28/09/18

2- Esta parte del código maneja los encoders:

```

/*****
*****
*   Función counter_1
*   Descripción: cuenta los pulsos buenos
*   Parametros : ninguno
*   Valor de Retorno: ninguno
*****
*****
*/
void counter_1(){
  pulsos [0] ++; } // Suma el pulso bueno que entra.

/*****
*****
*   Función counter_2
*   Descripción: cuenta los pulsos buenos
*   Parametros : ninguno
*   Valor de Retorno: ninguno
*****
*****
*/
void counter_2(){
  pulsos [1] ++; } // Suma el pulso bueno que entra.

```

```

void loop(){
  if (millis() - timeold >= 1000){ //Se actualiza cada segundo, esto será igual a
    for (int i = 0 ; i < 2 ; i++){
      noInterrupts(); //No procesar interrupciones durante los cálculos.(Desconec
      rpm [i]= 60 * pulsos [i] / pulsos_por_vuelta * 1000 / (millis() - timeold)
      //Ojo con la fórmula de arriba, la variable rpm tiene que ser tipo float po
      velocidad [i]= rpm[i]/proporcion * 3.1416 * diametro_rueda * 60 / 1000000;
    }
    Serial.print(" ");
    Serial.print(millis()/1000); Serial.print(" | "); // Escribelo a puer
    Serial.print(pulsos[0],0); Serial.print(" | ");
    Serial.print(pulsos[1],0); Serial.print(" | ");
    Serial.print(rpm[0],0); Serial.print(" | ");
    Serial.print(rpm[1],0); Serial.print(" | ");
    Serial.print(rpm[0]/proporcion,1); Serial.print(" | ");
    Serial.print(rpm[1]/proporcion,1); Serial.print(" | ");
    Serial.print(velocidad[0],2); Serial.print(" | ");
    Serial.println(velocidad[1],2);
    pulsos [0]= 0; // Inicializamos los pulsos.
    pulsos [1]= 0; // Inicializamos los pulsos.
    timeold = millis(); // Almacenamos el tiempo actual.
    interrupts(); //Reinicie el procesamiento de interrupción (Reiniciamos la int
  }
}

```

Explicación:

mediante 2 interrupciones se llama a las funciones counter_x con x= 1 y 2 , los cuales incrementan en 1 y solo se consideran para el cálculo una vez que transcurran 1 seg (delta de tiempo) y se realiza la cuenta de las revoluciones para estas cuentas se debe tener en cuenta los siguientes factores:

- **delta de tiempo** → millis() - timeold.
- **pulsos contados hasta el momento en que delta de tiempo es mayor a 1.**
- **pulsos por vuelta** → CTE = número de ranuras del disco de encoder.
- **proporción** → CTE = es la relación de reducción del motorreductor (48:1).

Lenguaje: C++.

Fecha: 01/10/18

3- Esta parte del código maneja el modulo wifi:

```
void setup()
{
  // velocidad a elegir 19200    115200

  Serial.begin(115200); // la velocidad a elegir para el puerto es 115200
  ESP8266.begin(115200); // la velocidad a elegir para el modulo tambien debe ser de 115200
  Serial.println("Iniciando...");
  Serial.println("\nPrueba de Funcionamiento ESP8266:");

  comandoESP("AT+RST"); // resetear el modulo
  delay(3000);

  comandoESP("AT+CWMODE=2"); // configuro como access point
  comandoESP("AT+CIPMUX=1"); // configuro para multiples conexiones
  comandoESP("AT+CIPSERVER=1,80"); // encender el servidor en el puerto 80
  comandoESP("AT+CIFSR"); // nos da la direccion ip
  // (115200)
}
```

Explicación: Primero se configuran las velocidades tanto para el puerto como para el módulo wifi. Luego se resetea el módulo, a continuación se configura lo siguiente como:

- **acces point.**
- **múltiples conexiones a red.**
- **encender el servidor en el puerto 80.**
- **dar la dirección IP**

A Continuación se muestra lo que se ve en consola:

```

Iniciando...

Prueba de Funcionamiento ESP8266:
AR$)U$D$H$j$H$
ets Jan 8 2013,rst c`use:2, boot mode:(3,6)AT+CWM=
AT+CIPLUX=1

OK

AT+CIPSERVER=1,80

OK
AT+CIFSR
) $IFSR:APIP,#192.168.4.1#
+CIFSR:0:

```

Nota: falta realizar la interfaz de la página. Pero la conexión de la misma ya está realizado.

```

void loop()
{
  //Serial.println('.');
  if(ESP8266.available()) // comprueba si el esp está enviando un mensaje
  {
    Serial.println("CONECTADO ");
    if(ESP8266.findUntil("+IPD,",".")) // si el modulo encuentra al final "+IPD,","."
    {
      delay(100); // espere a que se llene el búfer serial (lea todos los datos seriales)
      // obtener el ID de conexión para que podamos desconectar
      connectionId = ESP8266.read()-48; // restar 48 porque la función read () devuelve
      // el valor decimal ASCII y 0 (el primer número decimal) comienza en 48

      Serial.println("leyendo pagina...");
      // en su tipo de navegador debe escribir http://192.168.4.1/cmd=izqx

      if(ESP8266.find("cmd=")) // avance el cursor hasta "cmd="
      {
        tipo = ESP8266.readStringUntil('x');// retorna la cadena completa leída del
        //buffer serie (por ejemplo cmd=derx), hasta que se detecte el carácter x.
        //es decir, que al colocar la x al final del comando que querramos realizar
        //ahi sabe que tiene que captar dicho comando (der,izq,on).

        Serial.println(tipo);
        // comienzo del arranque de las ruedas, adelante, derecha, izquierda, atras, parar.
        if(tipo == "on") // si es adelante

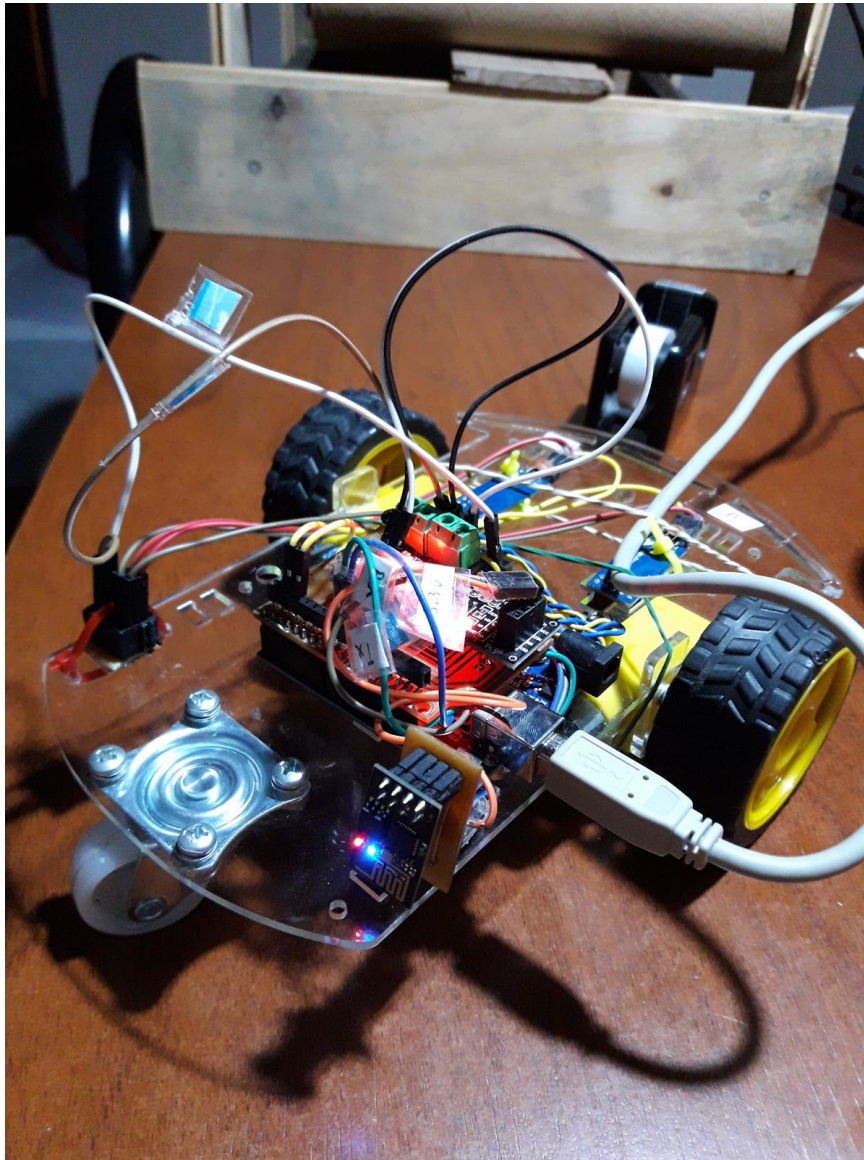
```

Explicación: Aquí se comprueba si el módulo wifi está enviando un mensaje. Si es así es porque está conectado. Luego se espera a que se llene el buffer serial, se obtiene la id de

conexión para que podamos conectar. Cuando se escribe la dirección <http://192.168.4.1/cmd=izqx> en la web, se podrá realizar la acción mover rueda izquierda. Lenguaje: C++.

5- Documentación en Formato Gráfico

Estado actual de auto:



Anexo

Anexo A

13/9/2018

Se comenzó con la investigación de motores de CC y sistemas de control de lazo cerrado, y se documentó referencias respecto al mismo. El artículo leído explica cómo usar el módulo de controlador de motor de doble canal HG7881 (L9110).

El HG7881 (L9110) es un chip de controlador de motor compacto que admite un rango de voltaje de 2.5-12 V a 800 mA de corriente continua. Estos chips tienen diodos de abrazadera de salida incorporados para proteger su electrónica sensible del micro controlador. Son adecuados para proyectos de robots muy pequeños.

Cada chip HG7881 (L9110) puede conducir un solo motor de CC con dos entradas de control digitales. Una entrada se usa para seleccionar la dirección del motor mientras que la otra se usa para controlar la velocidad del motor. La velocidad se controla utilizando la modulación de ancho de pulso PWM. Los conductores de motor suelen tener lo que se llama una tabla de verdad que determina el efecto de sus entradas. La tabla de verdad para un solo chip HG7881 (L9110) es la siguiente:

HG7881 (L9110) tabla de verdad del motor

entrada		salida		
IA	I B	O A	I B	estado del motor
L(bajo)	L	L	L	apagado
H(alto)	L	H	L	Adelante
L	H	L	H	Marcha atrás
H	H	H	H	apagado

Tenga en cuenta que la dirección real de "adelante" e "atrás" depende de cómo se montan y cablean los motores. Siempre puede cambiar la dirección de un motor invirtiendo su cableado.

El módulo de controlador de motor de doble canal HG7881 (L9110) utiliza dos de estos chips de controlador de motor. Cada chip del controlador está destinado a conducir un motor, por lo que tener dos significa que este módulo puede controlar dos motores de forma independiente. Cada canal de motor usa la misma tabla de verdad que la anterior. Cada juego de terminales de tornillo se utiliza para conectar un motor. Consulte la tabla a continuación para conocer las conexiones de los cabezales de los pines.

HG7881 (L9110) Conector de módulo de controlador de motor de doble canal

Pin	Descripcion
B-IA	Motor B entrada A (IA)
B-IB	Motor B salida B (IB)
GN D	Ground
VCC	Tensión de funcionamiento 2.5-12V
A-IA	Motor A entrada A (IA)
A-IB	Motor A entrada (IB)

Recomendamos usar la entrada 1A para controlar la velocidad de cada motor y la entrada 1B para controlar la dirección.

Conexiones:

Salida digital Arduino D10 a la entrada del controlador del motor B-IA.
 Salida digital Arduino D11 a la entrada del controlador del motor B-IB.
 Motor controlador VCC a voltaje de operación 5V.
 Motor conductor GND a tierra común.
 Motor del conductor MOTOR B atornille los terminales a un motor pequeño

El siguiente boceto de Arduino muestra cómo controlar un solo motor:

```
// conexiones cableadas
#define HG7881_B_IA 5 // D10 -> Motor B Entrada A -> MOTOR B +
```

```

#define HG7881_B_IB 4 // D11 -> Motor B Entrada B -> MOTOR B -

// conexiones funcionales
#define MOTOR_B_PWM HG7881_B_IA // Velocidad PWM del motor B
#define MOTOR_B_DIR HG7881_B_IB // Motor B Dirección

// los valores reales para "rápido" y "lento" dependen del motor
#define PWM_SLOW 50 // ciclo de trabajo arbitrario de baja velocidad PWM
#define PWM_FAST 200 // ciclo de trabajo arbitrario de velocidad rápida PWM
#define DIR_DELAY 1000 // breve retraso para cambios bruscos de motor

void setup()
{
    Serial.begin( 9600 );
    pinMode( MOTOR_B_DIR, OUTPUT );
    pinMode( MOTOR_B_PWM, OUTPUT );
    digitalWrite( MOTOR_B_DIR, LOW );
    digitalWrite( MOTOR_B_PWM, LOW );
}

void loop()
{
    boolean isValidInput;
    // dibujar un menú en el puerto serie
    Serial.println( "-----" );
    Serial.println( "MENU:" );
    Serial.println( "1) Avance rápido" );
    Serial.println( "2) Adelante" );
    Serial.println( "3) parar suave (detenerse despacio)" );
    Serial.println( "4) marcha atras" );
    Serial.println( "5) marcha atras rapido" );
    Serial.println( "6) Parada de golpe (freno)" );
    Serial.println( "-----" );
    do
    {
        byte c;
        // obtener el siguiente carácter del puerto serie
        Serial.print ( "?");
    }
}

```



```

    while (! Serial.available ())
        ; // LAZO...
c = Serial.read ();
    // ejecuta la opción de menú según el personaje recibido
switch( c )
{
    case '1': // 1) Avance rápido
        Serial.println ("Avance rápido ...");
        // siempre detiene los motores brevemente antes de cambios abruptos
        digitalWrite (MOTOR_B_DIR, LOW);
        digitalWrite (MOTOR_B_PWM, LOW);
        delay (DIR_DELAY);
        // establece la velocidad y dirección del motor
        digitalWrite (MOTOR_B_DIR, HIGH); // direccion = adelante
        analogWrite (MOTOR_B_PWM, 255-PWM_FAST); // velocidad PWM = rápido
        isValidInput = true;
        break;

    case '2': // 2) adelante
        Serial.println( "adelante..." );
        // siempre detenga los motores brevemente antes de cambios abruptos

        digitalWrite( MOTOR_B_DIR, LOW );
        digitalWrite( MOTOR_B_PWM, LOW );
        delay( DIR_DELAY );
        // establecer la velocidad y dirección del motor
        digitalWrite( MOTOR_B_DIR, HIGH ); // direccion = adelante
        analogWrite( MOTOR_B_PWM, 255-PWM_SLOW ); // velocidad del PWM =
lento
        isValidInput = true;
        break;

    case '3': // 3) Parada suave (preferida)
        Serial.println( "Tope suave (costa) ..." );
        digitalWrite( MOTOR_B_DIR, LOW );
        digitalWrite( MOTOR_B_PWM, LOW );
        isValidInput = true;
        break;

```

```

case '4': // 4) Reversa
    Serial.println( "Avance rápido..." );
    // siempre detenga los motores brevemente antes de cambios abruptos
    digitalWrite( MOTOR_B_DIR, LOW );
    digitalWrite( MOTOR_B_PWM, LOW );
    delay( DIR_DELAY );
    //establecer la velocidad y dirección del motor
    digitalWrite( MOTOR_B_DIR, LOW ); // direccion = reversa
    analogWrite( MOTOR_B_PWM, PWM_SLOW ); // velocidad del PWM = lento
    isValidInput = true;
    break;

case '5': // 5) reverso rapido
    Serial.println( "adelante rapido..." );
    // siempre detenga los motores brevemente antes de cambios abruptos

    digitalWrite( MOTOR_B_DIR, LOW );
    digitalWrite( MOTOR_B_PWM, LOW );
    delay( DIR_DELAY );
    // establecer la velocidad y dirección del motor
    digitalWrite( MOTOR_B_DIR, LOW ); // direccion = reversa
    analogWrite( MOTOR_B_PWM, PWM_FAST ); // velocidad del PWM = rapido
    isValidInput = true;
    break;

case '6': // 6) Parada de golpe (uso con precaución)
    Serial.println( "parada de golpe (frenar)..." );
    digitalWrite( MOTOR_B_DIR, HIGH );
    digitalWrite( MOTOR_B_PWM, HIGH );
    isValidInput = true;
    break;

default:
    // caracter invalido! vea el menu otra vez!
    isValidInput = false;
    break;

```

```
    }  
    } while( isValidInput == true );  
  
    // repite el main loop y dibuja el menu...  
}  
/*EOF*/
```

HG7881_Motor_Driver_Example - Arduino sketch

Es importante tener en cuenta que el módulo de controlador de motor de doble canal HG7881 (L9110) es muy simple. El voltaje utilizado para conducir el chip es el mismo voltaje utilizado para conducir los motores. Esto significa que si está utilizando un chip de 5V, debe esperar conducir motores de 5V. Elija un controlador de motor de mayor rendimiento para obtener más opciones de potencia y control.

Enlaces de referencias para tener en cuenta respecto a los motores:

Video de control de velocidad PWM ANÁLISIS y CONTROL:
<https://www.youtube.com/watch?v=BMncYdbufng>

Video de control de motores:
<https://www.youtube.com/watch?v=mVo8DfEsRno>

Anexo B

22/9/2018

USO DEL ESP8266 CON COMANDOS AT

La comunicación con el ESP01 con el firmware por defecto se realiza a través de comandos AT, que son comandos de texto enviados por Serial.

Podemos enviar estos comandos por un conversor USB-TTL (FT232, CH340G o CP2102) o, en nuestro caso, usando Arduino y Software serial como adaptador.

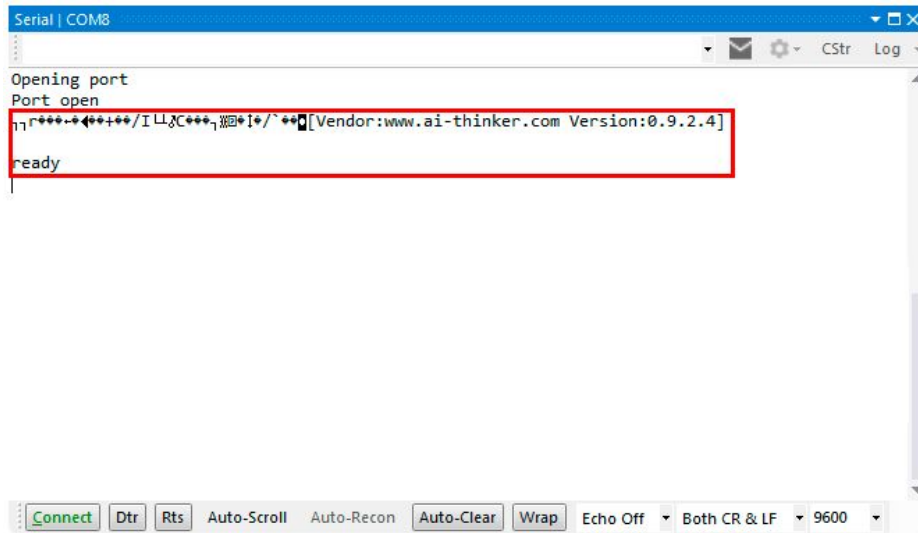
PRIMERA PRUEBA

Vamos a hacer la primera prueba de conexión con el ESP01. Para ello conectamos el ESP01 a Arduino como hemos visto en el apartado anterior. Dejamos conectado Arduino al ordenador por USB.

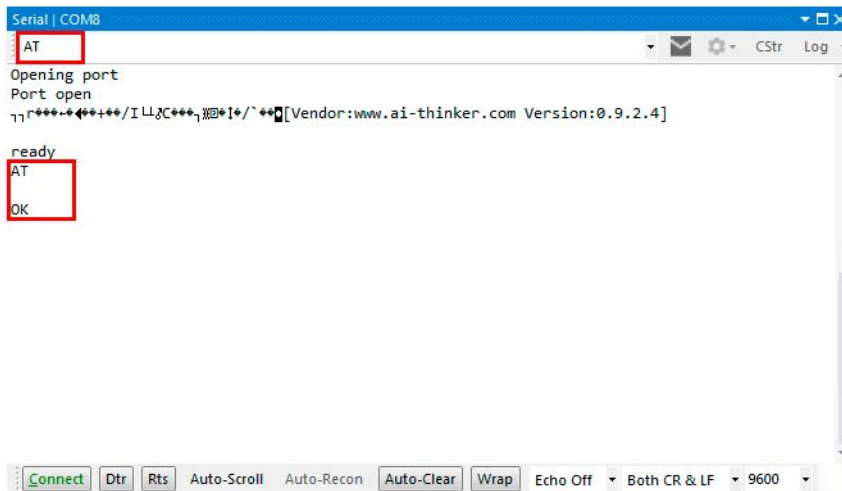
A continuación cargamos el siguiente Sketch en Arduino, que seguramente alguno reconocerá como el programa Serial Loop. Este sketch únicamente actúa “puente” entre el puerto serie hardware conectado con el PC, y el puerto serie Soft conectado al ESP01.

```
1 // La velocidad depende del modelo de ESP-01
2 // siendo habituales 9600 y 115200
3 const int baudRate = 9600;
4
5 #include "SoftwareSerial.h"
6 SoftwareSerial softSerial(2, 3); // RX, TX
7
8 void setup()
9 {
10     Serial.begin(baudRate);
11     softSerial.begin(baudRate);
12 }
13
14 void loop()
15 // enviar los datos de la consola serial al ESP-01,
16 // y mostrar lo enviado por el ESP-01 a nuestra consola
17 {
18     if (softSerial.available())
19     {
20         Serial.print((char)softSerial.read());
21     }
22     if (Serial.available())
23     {
24         softSerial.print((char)Serial.read());
25     }
26 }
```

Una vez cargado el Sketch, encendemos (o reiniciamos) el ESP01. En el Monitor Serie el ESP01 responde con una serie de caracteres que dependen del fabricante y modelo, y finalmente “Ready”, indicando que el módulo está listo.



Si ahora escribimos AT, el módulo responderá con “OK”, indicando de nuevo que el módulo está listo para su uso.



Si no se ve el mensaje inicial finalizado en “Ready” y en su lugar aparecen “caracteres raros”, se debe cambiar la velocidad del puerto Serie en el Sketch Serial Loop, y en el Monitor Serie.

Nota: Tener en cuenta que se mandan comandos a la terminal, y que la comunicación del arduino se realiza mediante los pines 0 (RX) y 1 (TX).

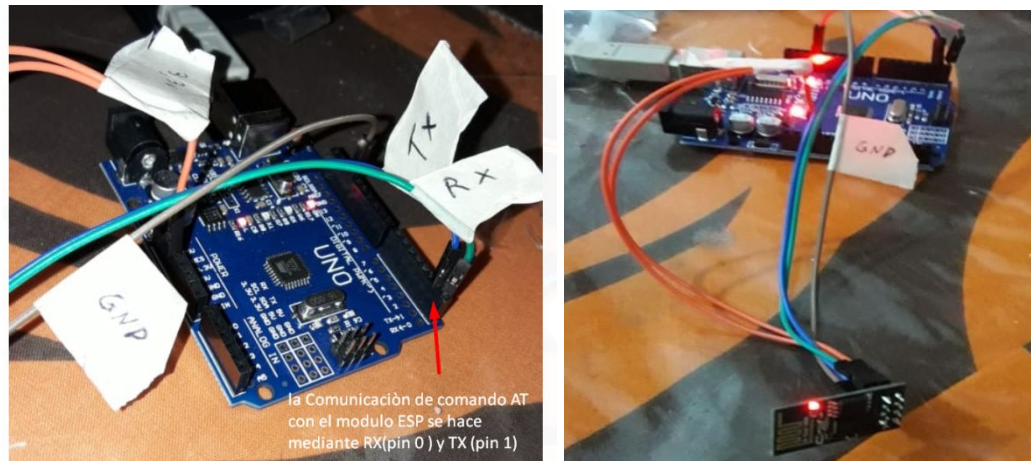


Figura 3: Conexionado físico Módulo ESP8266 y Arduino Uno

Anexo C

13/9/2018

Encoder

Probamos el funcionamiento del encoder viendo el siguiente video:

<https://www.youtube.com/watch?v=t9Zy8UvJXsl>

```

encoder$
int encoderA = 2;      // pin de conexión del encoder
int encoderPos = 0;    // contador de pasos
int encoderALast = LOW; // variable para estado anterior del encoder
int n = LOW;           // lectura del encoder

void setup() {
  pinMode (encoderA, INPUT); // pin del arduino se declara como entrada
  Serial.begin (9600);       // comunicación serial
}

void loop() {
  n = digitalRead(encoderA); // lectura del pin donde e
  if ((encoderALast == LOW && (n == HIGH)) { // cuando la lectura del e
    encoderPos++; // incremento de posición
    Serial.println (encoderPos); // se muestra la posición
    Serial.println(encoderPos*36); // se muestra la posición
    Serial.println((encoderPos*0.2*3.141592)*(6.5/2)); // se muestra la
  }
  encoderALast = n; // ultima lectura del enco
}

```

COM3

2
72
4.08
108
6.13
4
144
8.17

Figura 1: Primera prueba del encoder**7/10/2018**

Se realizaron pruebas de medición de rpm para una rueda del autito. Para esto se implementó el manejo de interrupciones de hardware mediante un encoder que dispara eventos. Cada vez que se activa un evento se realiza un cálculo de frecuencia relativo a cada cuanto una rueda realiza una vuelta completa. El código para el cálculo es el siguiente:

```
void Encoder() {
```

```
    encoderNow = digitalRead(encoderB);  
    if((encoderLast == LOW)&&(encoderNow == HIGH)) {  
        contador++;  
        if(contador == 6){  
            contador=0;  
            frecuencia = (1 * 1000) / (double) deltaTiempoInterrupciones; // frecuencia de las interrupciones  
en Hz  
            tiempoInterrupcionAnterior = tiempoInterrupcionActual;  
        }  
    }  
    encoderLast = encoderNow;  
}
```

Para establecer el valor de contador (variable de control de una vuelta) se pensó en lo siguiente:

“como el disco de encoder posee 6 ranuras se estima que una vuelta de la rueda del autito equivale a 6 transiciones entre estados LOW y HIGH “

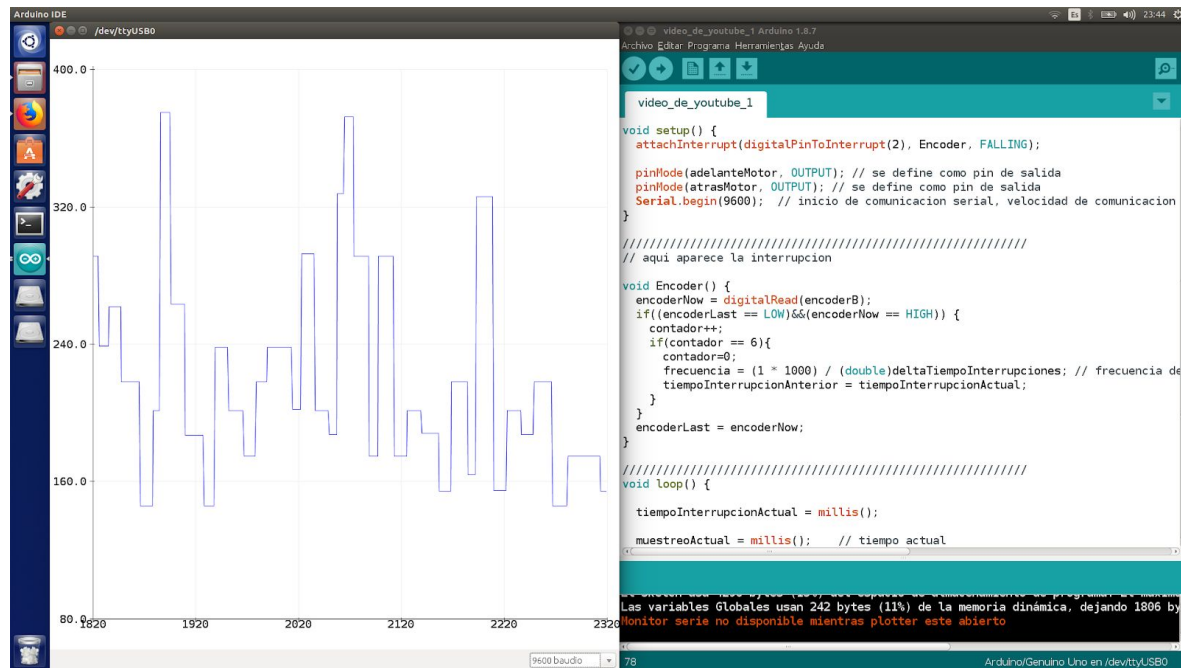


Figura 7: medición de velocidad de una rueda del autito

21/10/18

Con respecto a la velocidad de cada rueda, sin considerar el roce con el suelo, se logró llegar a un valor en común para ambas. En la siguiente figura se ve el valor al que converge la rotación de los dos motores. Este valor será elegido como valor deseado para el algoritmo del PID para el control de la velocidad de los motores.

Segundos	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28
	RPM A-->	100	FWM A-->	201																								
3	RPM A-->	100	FWM A-->	201																								
4	RPM A-->	90	FWM A-->	201																								
5	RPM A-->	100	FWM A-->	201																								
6	RPM A-->	100	FWM A-->	201																								
7	RPM A-->	90	FWM A-->	201																								
8	RPM A-->	100	FWM A-->	201																								
9	RPM A-->	100	FWM A-->	201																								
10	RPM A-->	100	FWM A-->	201																								
11	RPM A-->	90	FWM A-->	201																								
12	RPM A-->	100	FWM A-->	201																								
13	RPM A-->	100	FWM A-->	201																								
14	RPM A-->	90	FWM A-->	201																								
15	RPM A-->	100	FWM A-->	201																								
16	RPM A-->	90	FWM A-->	201																								
17	RPM A-->	100	FWM A-->	201																								
18	RPM A-->	90	FWM A-->	201																								
19	RPM A-->	110	FWM A-->	201																								
20	RPM A-->	100	FWM A-->	201																								
21	RPM A-->	100	FWM A-->	201																								
22	RPM A-->	90	FWM A-->	201																								
23	RPM A-->	100	FWM A-->	201																								
24	RPM A-->	100	FWM A-->	201																								
25	RPM A-->	110	FWM A-->	201																								
26	RPM A-->	100	FWM A-->	201																								
27	RPM A-->	100	FWM A-->	201																								
28	RPM A-->	100	FWM A-->	201																								

Esta oscilando

se estabiliza

Figura 8: Valor de RPM deseado.

Anexo D

24/10/2018

Se implementó un librería para la medición de la velocidad llamada Encoder.h, con las siguientes funciones:

```
class Encoder {
private:
    int encoder_pin [2];
    float rpm [2]; // Revoluciones por minuto calculadas.
    volatile float pulsos [2]; // Número de pulsos leídos por el Arduino en un segundo
    unsigned long timeold; // Tiempo
    unsigned int pulsos_vuelta; // Número de pulsos por vuelta del motor, por canal = 6, depende del tipo de encoder!!
public:
    Encoder(int,int);//asigna variables con valores default.
    void counter_1();// incrementar pulsos en 1.
    void counter_2();// incrementar pulsos en 1.
    void calculo();// calcular rpm.
    void reset_Counters();// poner en 0 los pulsos.

    float get_Rpm(int i);// obtener rpm del encoder iesimo.
    unsigned long get_Timeold();// obtener timeold.
    unsigned long get_Pulsos(int i);// obtener cantidad de pulsos contados hasta el momento.
};
```

Función que cuenta el número de pulsos este es llamado como rutina de servicio para atender una interrupción:

```
/*
*****
*****
*   Función counter
*   Descripción: cuenta los pulsos buenos
*   Parametros : ninguno
*   Valor de Retorno: ninguno
*****
*****
*/
void Encoder::counter_1(){
    pulsos [0] ++; } // Suma el pulso bueno que entra.

void Encoder::counter_2(){
    pulsos [1] ++; } // Suma el pulso bueno que entra.
```

Función que realiza el cálculo de RPM para cada rueda:

```

/*****
*****
* Función calculo
* Descripción: calcula para cada rueda los rpm correspondientes
* Parametros : ninguno
* Valor de Retorno: ninguno
*****
*****
*/
void Encoder::calculo(){
    for (int i = 0 ; i < 2 ; i++){
        noInterrupts();

        /*No procesar interrupciones durante los cálculos.
        (Desconectamos la interrupción para que no actúe en esta parte del programa.)*/

        rpm [i]= 60 * pulsos [i] / pulsos_vuelta * 1000 / (millis() - timeold) ;

        /* Tenga en cuenta que esto sería 60 * 1000 / (milis () - timeold) * pulsos si la interrupción.
        sucedió una vez por revolución.(Calculamos las revoluciones por minuto)
        Ojo con la fórmula de arriba,
        la variable rpm tiene que ser tipo float porque salen decimales en medio de la operación.*/

    }
}

```

con el siguiente algoritmo podemos ver la velocidad y el PWM aplicado a los motores A y B

```

int pmwa = 255;
int pmwb = 255;

void loop() {
    motor.adelante(pmwa, pmwb, 1);

    if (millis() - encoder.get_Timeold() >= 1000) { //Se actualiza cada segundo, esto será igual
        encoder.calculo();
        Serial.print("Segundos Transcurridos --> ");
        Serial.print(millis() / 1000);      Serial.print("    | RPM A --> "); // Escribelo a la salida
        Serial.print(encoder.get_Rpm(0), 0); Serial.print("    | RPM B --> "); //a
        Serial.print(encoder.get_Rpm(1), 0); Serial.print("    | PWM A --> "); //b
        Serial.print(pmwa);                  Serial.print("    | PWM B --> ");
        Serial.print(pmwb);                  Serial.print("    | ");
        Serial.println(" ");
        encoder.reset_Counters();
        interrupts(); //Reinicie el procesamiento de interrupción (Reiniciamos la interrupción)
    }
}

```

Segundos Transcurridos --> 29	RPM A --> 180	RPM B --> 140	PWM A --> 255	PWM B --> 255	
Segundos Transcurridos --> 30	RPM A --> 170	RPM B --> 140	PWM A --> 255	PWM B --> 255	
Segundos Transcurridos --> 31	RPM A --> 170	RPM B --> 140	PWM A --> 255	PWM B --> 255	
Segundos Transcurridos --> 32	RPM A --> 190	RPM B --> 130	PWM A --> 255	PWM B --> 255	
Segundos Transcurridos --> 33	RPM A --> 170	RPM B --> 140	PWM A --> 255	PWM B --> 255	
Segundos Transcurridos --> 34	RPM A --> 170	RPM B --> 150	PWM A --> 255	PWM B --> 255	
Segundos Transcurridos --> 35	RPM A --> 180	RPM B --> 140	PWM A --> 255	PWM B --> 255	
Segundos Transcurridos --> 36	RPM A --> 170	RPM B --> 140	PWM A --> 255	PWM B --> 255	
Segundos Transcurridos --> 37	RPM A --> 180	RPM B --> 140	PWM A --> 255	PWM B --> 255	
Segundos Transcurridos --> 38	RPM A --> 170	RPM B --> 140	PWM A --> 255	PWM B --> 255	
Segundos Transcurridos --> 39	RPM A --> 170	RPM B --> 140	PWM A --> 255	PWM B --> 255	
Segundos Transcurridos --> 40	RPM A --> 180	RPM B --> 140	PWM A --> 255	PWM B --> 255	
Segundos Transcurridos --> 41	RPM A --> 180	RPM B --> 130	PWM A --> 255	PWM B --> 255	
Segundos Transcurridos --> 42	RPM A --> 170	RPM B --> 140	PWM A --> 255	PWM B --> 255	
Segundos Transcurridos --> 43	RPM A --> 180	RPM B --> 140	PWM A --> 255	PWM B --> 255	
Segundos Transcurridos --> 44	RPM A --> 170	RPM B --> 140	PWM A --> 255	PWM B --> 255	
Segundos Transcurridos --> 45	RPM A --> 180	RPM B --> 130	PWM A --> 255	PWM B --> 255	
Segundos Transcurridos --> 46	RPM A --> 170	RPM B --> 140	PWM A --> 255	PWM B --> 255	
Segundos Transcurridos --> 47	RPM A --> 170	RPM B --> 140	PWM A --> 255	PWM B --> 255	
Segundos Transcurridos --> 48	RPM A --> 180	RPM B --> 130	PWM A --> 255	PWM B --> 255	
Segundos Transcurridos --> 49	RPM A --> 180	RPM B --> 140	PWM A --> 255	PWM B --> 255	
Segundos Transcurridos --> 50	RPM A --> 180	RPM B --> 140	PWM A --> 255	PWM B --> 255	
Segundos Transcurridos --> 51	RPM A --> 170	RPM B --> 140	PWM A --> 255	PWM B --> 255	
Segundos Transcurridos --> 52	RPM A --> 190	RPM B --> 130	PWM A --> 255	PWM B --> 255	
Segundos Transcurridos --> 53	RPM A --> 180	RPM B --> 140	PWM A --> 255	PWM B --> 255	
Segundos Transcurridos --> 54	RPM A --> 170	RPM B --> 150	PWM A --> 255	PWM B --> 255	
Segundos Transcurridos --> 55	RPM A --> 170	RPM B --> 130	PWM A --> 255	PWM B --> 255	
Segundos Transcurridos --> 56	RPM A --> 180	RPM B --> 140	PWM A --> 255	PWM B --> 255	
Segundos Transcurridos --> 57	RPM A --> 180	RPM B --> 140	PWM A --> 255	PWM B --> 255	
Segundos Transcurridos --> 58	RPM A --> 170	RPM B --> 130	PWM A --> 255	PWM B --> 255	
Segundos Transcurridos --> 59	RPM A --> 170	RPM B --> 140	PWM A --> 255	PWM B --> 255	
Segundos Transcurridos --> 60	RPM A --> 170	RPM B --> 150	PWM A --> 255	PWM B --> 255	

Figura 8: variación de RPM a partir de los 60 segundos y los 30 segundos

27/10/2018

Se implementó un librería para el control de motores llamada L9110.h, con las siguientes funciones:

```
class L9110 {
private:
    int _aia;
    int _aib;
    int _bia;
    int _bib;
public:
```



```

        L9110(int aia,int aib,int bia,int bib);
        void adelante(int i,int d,int t);
        void atras(int i,int d,int t);
        void izquierda(int i,int d,int t);
        void derecha(int i, int d,int t);
        void frenar(int t);
};

```

Las siguientes son funciones para el desplazamiento del carrito:

```

void L9110::adelante(int i,int d,int t){//desplazamiento hacia adelante
    digitalWrite(_aib,LOW);
    digitalWrite(_bib,LOW);
    analogWrite(_aia,i);
    analogWrite(_bia,d);
    delay(t);
}

void L9110::atras(int i,int d,int t){//desplazamiento hacia atras
    digitalWrite(_aib,LOW);
    digitalWrite(_bib,LOW);
    analogWrite(_aia,i);
    analogWrite(_bia,d);
    delay(t);
}

void L9110::izquierda(int i,int d,int t){//desplazamiento hacia izquier
    analogWrite(_aia,i);
    digitalWrite(_aib,LOW);
    digitalWrite(_bia,LOW);
    analogWrite(_bib,d);
    delay(t);
}

void L9110::derecha(int i,int d,int t){//desplazamiento hacia derecha
    digitalWrite(_aia,0);

```

```

    analogWrite(_bia,d);
    analogWrite(_aib,i);
    digitalWrite(_bib,0);
    delay(t);
}

void L9110::frenar(int t){//para total del desplazamiento
    analogWrite(_aia,0);
    analogWrite(_bia,0);
    digitalWrite(_aib,0);
    digitalWrite(_bib,0);
    delay(t);
}

```

la funcionalidad adelante() fue usada en la implementación del código anterior
 En cada función se pueden apreciar 3 parámetros a excepción de frenar que solo tiene tiempo, para los demás los primero 2 parámetros representan el pwm a aplicar al motor 1 y 2 respectivamente.

24/10/2018

Se Utilizó librería PID_V1 para conseguir que el auto ande recto con las siguientes funciones:

```

PID(double*, double*, double*, double, double, double, int);          //constructor. vincula el
PID a la entrada, la salida y el punto de ajuste. Los parámetros iniciales de ajuste también se
establecen aquí

void SetMode(int Mode);// * establece PID en Manual (0) o Automático (no 0)

bool Compute(); // realiza el cálculo PID. debería llamarse cada ciclo de tiempo () ciclos. ON /
OFF y la frecuencia de cálculo se pueden configurar usando Set Mode, Set SampleTime
respectivamente

void SetOutputLimits(double, double); //sujeta la salida a un rango específico. 0-255 por
defecto, pero es probable que el usuario quiera cambiar esto dependiendo de la aplicación

```


COMUNICACIÓN CON EL PID:

En el constructor se establece como parámetros Input, Output y Setpoint las cuales serían las variables de control para el funcionamiento de los motores.

- La acción de control aplicada se obtiene a través de del segundo parámetro del constructor
- La señal de entrada se envía por referencia a través del primer parámetro
- El valor de ajuste deseado se envía a través del tercer parámetro

Anexo E

El siguiente código representa el movimiento del auto, usando el módulo wifi, la batería y comandos de teclado:

```
#include <SoftwareSerial.h>
#include <L9110.h>

//----- Variables
String tipo = "Hola \n\r";
int connectionId;
int Intnumero;
int Intaccion;
char inicio;

//-----Creando Instancias de Objetos
SoftwareSerial ESP8266(9,8); /* RX:9, TX:8 OJO revisar conexiones y confirmar si esta bien*/
L9110 motor(4,5,6,7); /*Revisar el orden de los pines por que las funciones responden bien si los pines estan bien configurados*/

void setup()
{ // velocidad a elegir 19200 115200
```

```

Serial.begin(115200); // la velocidad a elegir para el puerto es 115200
ESP8266.begin(115200); // la velocidad a elegir para el modulo tambien debe ser de 115200
Serial.println("Iniciando...");
Serial.println("\nPrueba de Funcionamiento ESP8266:");

comandoESP("AT+RST"); // resetear el modulo
delay(3000);

comandoESP("AT+CWMODE=2"); // configuro como access point
comandoESP("AT+CIPMUX=1"); // configuro para multiples conexiones
comandoESP("AT+CIPSERVER=1,80"); // encender el servido en el puerto 80
comandoESP("AT+CIFSR"); // nos da la direccion ip
delay(6000);
}

void loop()
{
  //Serial.println('.');
  if(ESP8266.available()) // comprueba si el esp está enviando un mensaje
  {
    Serial.println("CONECTADO ");
    if(ESP8266.findUntil("+IPD",".")) // si el modulo encuentra al final "+IPD","."
    {
      delay(100); // espere a que se llene el búfer serial (lea todos los datos seriales)
      // obtener el ID de conexión para que podamos desconectar
      connectionId = ESP8266.read()-48; // restar 48 porque la función read () devuelve
      // el valor decimal ASCII y 0 (el primer número decimal) comienza en 48

      Serial.println("leyendo pagina...");
      // en su tipo de navegador debe escribir http://192.168.4.1/cmd=izqx

      if(ESP8266.find("cmd=")) // avance el cursor hasta "cmd="
      {
        tipo = ESP8266.readStringUntil('x');// retorna la cadena completa leída del buffer serie
        (por ejemplo cmd=derx), hasta que se detecte el carácter x.
      }
    }
  }
}

```

//es decir, que al colocar la x al final del comando que querramos realizar ahi sabe que tiene que captar dicho comando (der,izq,on).

```

Serial.println(tipo);
// comienzo del arranque de las ruedas, adelante, derecha, izquierda, atras, parar.
if(tipo == "on") // si es adelante
{
  Serial.println("adelante");
  motor.adelante(240,240,1000);
}

if(tipo == "der") // si es derecha
{
  Serial.println("derecha");
  motor.derecha(240,240,1000);
}

if(tipo == "izq") // si es izquierda
{
  Serial.println("izquierda");
  motor.izquierda(240,240,1000);
}

if(tipo == "atr") // si es atras
{
  Serial.println("atras");
  motor.atras(240,240,1000);
}

if(tipo == "off") // si es parar
{
  Serial.println("parar");
  motor.frenar(2000);
}
}

```

//int pinNumber = (ESP8266.read()-48)*10; // obtengo el primer número, es decir, si es el pin 13 entonces el primer número es 1, luego multiplique para obtener 10

//pinNumber += (ESP8266.read()-48); //obtengo el segundo número, es decir, si el número pin es 13, entonces el segundo número es 3, entonces lo agrego como el primer número

//digitalWrite(pinNumber, !digitalRead(pinNumber)); // toggle pin

}

String closeCommand = "AT+CIPCLOSE="; //cerrar coneccion

closeCommand+=connectionId; // añadir conexión ID

comandoESP(closeCommand); //dejo coneccion

}

}

void comandoESP(String cmd)

{

ESP8266.println(cmd);

if(ESP8266.available()) // comprueba si el esp está enviando un mensaje

delay(1000);

Serial.println(ESP8266.readStringUntil(14)); // si el modulo encuentra al final 14

delay(1000);

}

