**Fecha: miércoles 16 de octubre.**

Actividad: se realizó la primera compra de los componentes necesarios para la realización del proyecto. Los siguientes componentes fueron comprados en la tienda MicroJPM: driver para motor DC L298N, fotocelda, carro con 2 llantas traseras impulsadas por motores DC de 3V-6V y una llanta delantera de movimiento libre, regulador de voltaje LM7805, MPU-9250 genérico, resistencias de diversos valores, jumpers. Y los siguientes componentes fueron comprados en la tienda virtual ElectroTEC: registro de corrimiento 74LS164 y Node MCU.

**Fecha: miércoles 23 de octubre.**

Actividad: se descargaron los archivos necesarios para que el entorno de Arduino pueda programar el ESP8266 de la placa NodeMCU. Para lograr esto se debió de introducir el “link” de la dirección [library1] en la que se encuentra dicho paquete en el campo de *Additional Board Manager URLs* ubicado en la ventana de *Preferences* en el IDE de Arduino. Posteriormente, se debió de acceder a la pestaña de Tools, luego en Boards/Boards Manager, buscar el paquete respectivo de la placa NodeMCU e instalar dichos archivos. Finalmente, se configuraron ciertos elementos para que coincidieran con las características de la placa en específico utilizada. Por ejemplo: el baud rate de la comunicación en 115200 badios por segundo, tamaño de la memoria Flash en 4M (1M SPIFFS) y la frecuencia de CPU en 80 MHz.

Seguidamente, para poder compilar el código base proporcionado como parte del enunciado del proyecto, se debieron de descargar un par de librerías necesarias para el control del ESP8266 [library esp8266] y del MPU9250 [library mpu] de los repositorios oficiales para el manejo del entorno de Arduino. Estos archivos se debían de colocar dentro de la carpeta de librerías de Arduino. En caso de no realizar esta instalación correctamente, se retornaba un error debido a la ausencia de los archivos referenciados en el código.

**Fecha: viernes 25 de octubre.**

Actividad: Se consideró que, para seguir avanzando con el proyecto, primero se debería de entender el funcionamiento del software y el hardware y los principios sobre los cuales están estructurados. Para lograr esto, primero se investigó acerca de algunos fundamentos sobre los cuales está basado el funcionamiento del circuito, por ejemplo, la comunicación I2C entre el NodeMCU y el MPU9250 [i2c]. Además, se buscó información acerca de la implementación de servidores y su comunicación con el cliente mediante sockets [servidores].

Además, se analizó la estructura del código base de Arduino. Se investigó acerca de algunos métodos preconstruidos cuyo significado no se conocía; por ejemplo, funciones como el shiftOut() , millis(), map() y analogWrite(). La explicación de cómo implementar dichas funciones se obtuvo del sitio oficial de Arduino. [arduino]

**Fecha: sábado 26 de octubre.**

Actividad: Se continuó con la investigación de información relevante a la estructura del proyecto. En esta ocasión, se buscó la familiarización con el funcionamiento de los circuitos integrados y demás componentes de hardware. En primer lugar, se buscó la hoja de datos del L298 [datasheet], el cual corresponde al driver de los motores de corriente directa a utilizar. Fue importante reconocer la funcionalidad de cada uno de los pines del circuito integrado, los cuales se representan en la Figura X, especialmente reconocer los pines que representan las entradas y los habilitadores. Esto puesto a que dichos pines son los que se conectarán a algunas salidas del NodeMCU y del shift register.

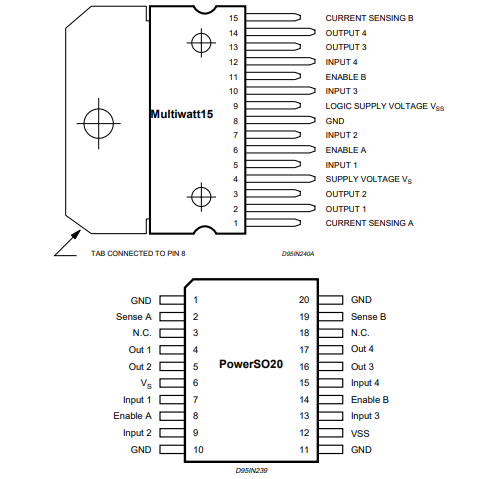


Figura X. Pines de conexión del L298.

A su vez, se buscaron las datasheets del shift register [datasheet shift] y el LM7805 [datasheet regulador], para igualmente reconocer cuáles son sus entradas y salidas. En el diagrama especificado en el enunciado del proyecto, se exponen las posiciones de los pines del NodeMCU tal cual son en la tarjeta real, por lo que no fue necesario verificar estas posiciones en su datasheet, sin embargo, esta también se buscó para futuras referencias [datasheet Node].

**Fecha: lunes 28 de octubre.**

Actividad: Una vez reconocidos los pines de todos los componentes a implementar, se procedió a probar cada componente por separado para verificar su funcionalidad. En primer lugar, se prueba el driver L298 utilizando un microcontrolador Arduino Uno. Se conectaron los *inputs* del integrado a las salidas digitales del Arduino, los *enables* a salidas digitales con pwm y los *outputs* se conectaron a los dos motores a utilizar. Se generó un código en Arduino que, a partir de datos enviados por el monitor serial, buscara controlar los motores en todas las direcciones posibles. Además, con las salidas de pwm, regular la velocidad. La prueba fue exitosa.

Seguidamente, se buscó probar el funcionamiento del *shift register* con el método *shiftOut()* de Arduino. Este método permite hacer el shift de un byte completo en el registro enviando bits individualmente al bit más significativo (MSB) o al menos significativo (LSB) de dicho registro con cada variación del reloj, de manera que con cada invocación del método, habrán 8 *toggles* del reloj para que se reemplace el byte completo. El método recibe cuatro parámetros: el pin de salida de la placa de dónde se envían los bits, el pin del reloj que controla el shift, el orden de los bits (primero el más o el menos significativo) y el byte que se introducirá al registro. Para esto se generó otro código en Arduino que simplemente enviara una serie de bytes al registro, mediante el método anterior, separados por un cierto delay, de manera que hubiera tiempo de medir con un multímetro las salidas del registro y verificar que se estuvieran dando los cambios deseados en cada bit.

Finalmente se probó el regulador de tensión de 5V. Para esto se armó un circuito resistivo simple y se conectó la entrada a una batería de 9V y luego se midió la tensión en una resistencia colocada entre la salida del integrado y la tierra del circuito, para verificar que efectivamente se entregaran 5V.

Problemas:

* Se empleó una protoboard antigua que, probablemente de tanto uso, tenía algunas conexiones internas dañadas por lo que se complicó el proceso de pruebas. En ocasiones el circuito funcionaba y en otras no. Se determinó que el problema era la condición de la *protoboard* ya que, al medir tensión en puntos importantes, si esta no correspondía a la que debería, bastaba con mover un poco el cable de ese nodo para que funcionara correctamente. Se decidió conseguir una *protoboard* diferente, en mejor estado.

**Fecha: martes 29 de octubre.**

Actividad: Se probó que el código de Arduino, proporcionado como parte del enunciado del proyecto, al ser cargado al NodeMCU generara un servidor y este pudiera establecer una conexión exitosa con el cliente, para cuya creación se utilizó el código de Python, también proporcionado desde un inicio.

Problemas:

* Al configurar la conexión a la red doméstica que se utilizó, se emplearon valores de IP erróneos. Se colocó la misma IP de la computadora en el código que genera el servidor, por lo que la comunicación con el NodeMCU no iba a ser lograda ya que la computadora intenta hacer una conexión consigo misma. Para que la conexión con el NodeMCU fuera exitosa, se cambió la dirección IP por otra, cuyo valor sea difícilmente usado por otro aparato (como el 200, por ejemplo). De esta forma sí se logró comunicar ambos dispositivos.

**Fecha: miércoles 30 de octubre**

Actividad: se procedió a armar al circuito en la *protoboard,* tal como se indica en la Figura X.

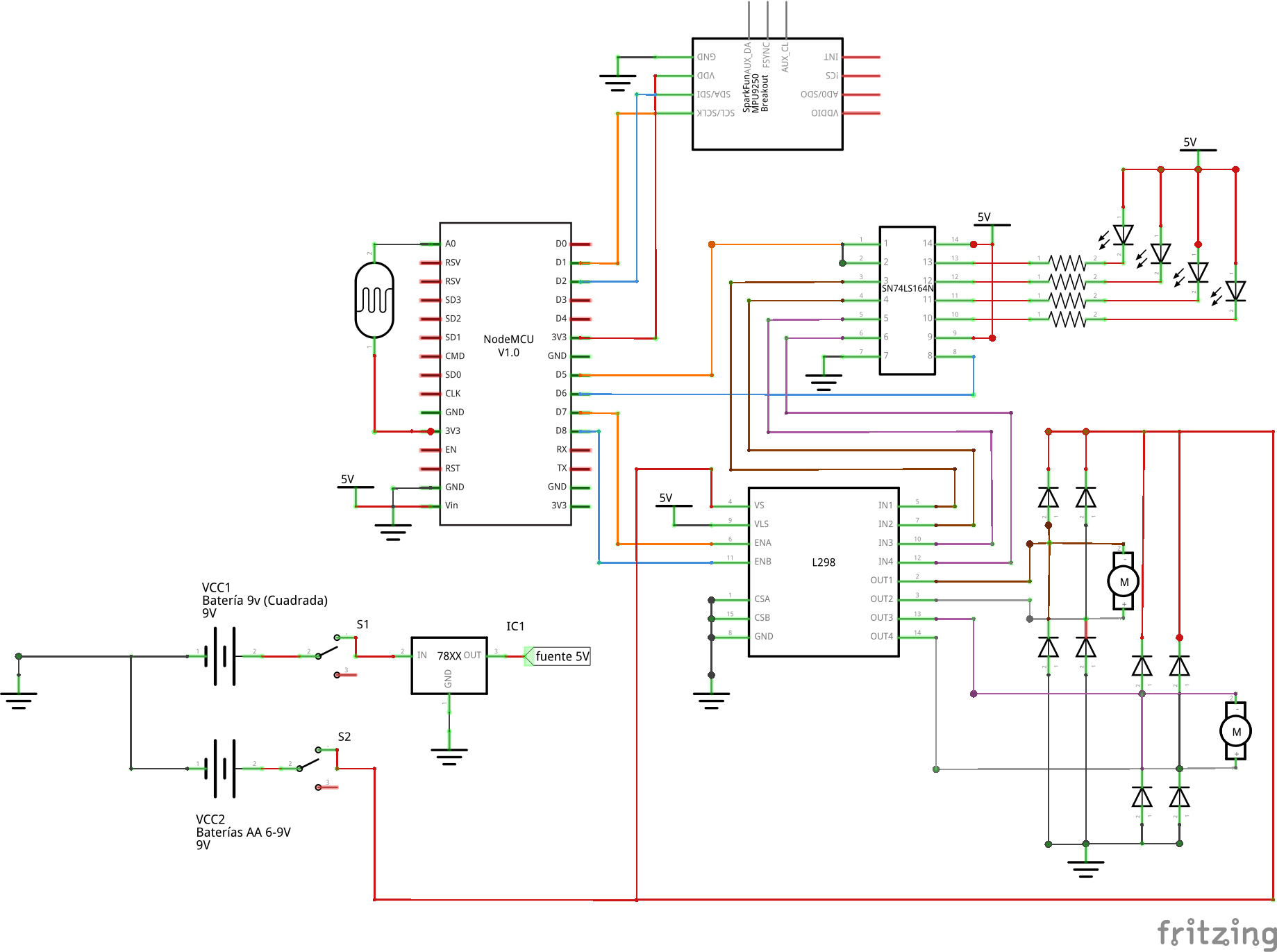


Figura X. Diagrama del circuito a implementar

Problemas:

* Luego de armar todo el circuito y alimentarlo, se notó que había una caída de tensión importante en el nodo que alimenta a la mayoría de componentes, el cual teóricamente debería ser de 5V. Se revisó que las conexiones estuvieran correctas y luego se decidió cambiar la batería de 9V que alimenta el regulador de tensión por una nueva, ya que la que se estaba utilizando ya había disminuido un poco su nivel de tensión ( a cerca de 6,5 V). Con la nueva batería, la tensión de salida del regulador sí cambió a los 5V deseados.
* El L298N no está diseñado para colocarse en protoboard puesto a que las patillas, por su distribución y estructura, no calzan dentro de orificios que forman los nodos de dicha placa. Se determinó que la mejor solución a este problema sería doblar las patillas, utilizando un alicate con puntas de pinza, con especial cuidado e intención para que pudieran calzar todas las patillas en diferentes nodos (dejando el surco central de la protoboard en medio de las 2 filas de patillas del circuito integrado).
* El NodeMCU tampoco está diseñado para colocarse en protoboard. En este caso, se decidió utilizar cables con terminales macho-hembra para lograr conectar, por un lado, a los pines de la placa con los pines hembra de los cables, y por el otro lado, los pines macho de los cables a los nodos de la protoboard deseados.
* Se utilizaron “jumpers” para realizar las conexiones entre nodos de la protoboard. Una vez que se terminó de construir el circuito del diagrama, se notó que la calidad visual del mismo era pobre; esto en el sentido de que, al estar tan cargado de “jumpers”, era difícil revisar las conexiones y verificar que estas se hayan hecho de forma correcta.
* Se tomó la decisión de rearmar el circuito, pero cortando cable de calibre 22 AWG de las dimensiones aptas para hacer cada una de las conexiones al ras de la *protoboard.*

Fuentes consultadas:

[arduino] https://www.arduino.cc/reference/en/language

https://www.luisllamas.es/arduino-i2c/

https://www.luisllamas.es/usar-arduino-con-los-imu-de-9dof-mpu-9150-y-mpu-9250/

https://programarfacil.com/podcast/nodemcu-tutorial-paso-a-paso/

https://programarfacil.com/esp8266/como-programar-nodemcu-ide-arduino/

https://www.instructables.com/id/The-74HC164-Shift-Register-and-your-Arduino/

<https://howtomechatronics.com/tutorials/arduino/arduino-dc-motor-control-tutorial-l298n-pwm-h-bridge/>

[servidores] <http://es.tldp.org/Universitarios/seminario-2-sockets.html>

[shiftOut] <https://www.arduino.cc/reference/tr/language/functions/advanced-io/shiftout/>

Datasheets:

[datasheet shift] https://www.esi.uclm.es/www/isanchez/apuntes/ci/74164.pdf

https://www.sparkfun.com/datasheets/Robotics/L298\_H\_Bridge.pdf

[datasheet regulador] http://ee-classes.usc.edu/ee459/library/datasheets/LM7805.pdf

[datasheet Node] https://www.esploradores.com/datasheet-nodemcu/

http://www.invensense.com/wp-content/uploads/2015/02/PS-MPU-9250A-01-v1.1.pdf

Librerías:

[library 1] https://arduino.esp8266.com/stable/package\_esp8266com\_index.json

[library esp8266] https://github.com/esp8266/Arduino

[library mpu] https://github.com/sparkfun/SparkFun\_MPU-9250-DMP\_Arduino\_Library