Servos: Arduino-ESP8266 (NodeMCU)

Fernando G. Tinetti, Julián Delekta

Reporte Técnico TR-RT-01-2023
III-LIDI, Fac. de Informática, UNLP
CIC, Prov. de Buenos Aires
Argentina
e-mail: fernando@info.unlp.edu.ar
Junio 2023

Resumen. Se explican en este documento algunos detalles de los controles de motores servo, su relación con señales PWM y algunos problemas o diferencias de funcionamiento encontrado entre placas Arduino y NodeMCU con ESP8266 (y que podría extenderse a placas que utilizan ESP32, por ejemplo). También se incluyen algunos detalles mecánicos observados al desarmar un servo.

1.- Introducción

Tanto para Arduino como para ESP8266 la biblioteca más básica para manejar servos mapea ("traduce") ángulos, en general entre 0º y 180º, a valores de PWM (Pulse Width Modulation) específicos para motores servo [1] [2]. Para varios tipos de hardware, incluyendo Arduino y ESP8266 se tiene el *sketch* ejemplo sweep.ino, que mueve el servo desde 0º a 180º y desde 180º a 0º de manera continua, con pasos relativamente pequeños de movimientos y tiempos de espera (*delays*) intermedios. No se dedicará espacio en este documento a los llamados motores servo "de rotación continua" sino a los que tienen una amplitud de movimiento menor a 360º.

Algunas bibliotecas implican programar con mayor nivel de detalle que la básica de Arduino e implementada por varias otras placas de desarrollo (son de menor nivel de abstracción). Más específicamente, las bibliotecas de menor nivel de abstracción dejan a cargo del programador el manejo de la señal de PWM para los motores servo. Estas bibliotecas normalmente basan en placas o *breakout boards* [15] específicas para el control de servos. Entre estas bibliotecas y placas, quizás la más conocida sea la biblioteca de Adafruit para su placa PCA9685 [5]. Aunque no necesariamente el programador tiene que generar por sí mismo la propia señal de PWM (como lo hace usualmente un sistema de tiempo real), sí tiene que definir parámetros tales como la frecuencia en la cual se genera el PWM y los límites de PWM asociados a los límites de movimiento del servo.

La relación entre PWM y el funcionamiento de un servo es conceptualmente sencilla, aunque hay mucha información errónea o que puede confundir muy fácilmente. Entre la mejor documentación disponible en la web se puede mencionar [16], dado que:

- Identifica y explica claramente la diferencia entre al ancho de pulso y la frecuencia, donde el ancho del pulso máximo es muy inferior al período de la señal.
- Hace una identificación explícita a la amplitud (tensión/volts) de la señal de PWM y no sólo al ancho de pulso. Aunque lo documentado no necesariamente es correcto en todos los casos (y así lo explican), la sola mención/explicación de esta diferenciación dada en [16] da una idea de que efectivamente conocen en detalle lo que están explicando.

Es importante notar que PWM en general no necesariamente es el PWM que "reconocen" los servomotores, dado que PWM es un concepto general, que luego los controles de los motores servo implementan de manera específica y casi idéntica para todos los modelos. Las diferencias más comunes de motores servo se refieren a

- Torque.
- Amplitud de movimientos (la mayoría son de 0º a 180º, aunque los hay de 0º a 270º).
- Material de engranajes (plástico o metal).
- Ejes (1 ó 2).
- Consumo.
- Velocidad de movimiento.

Donde normalmente los de mayor torque también son de mayor costo y consumo y suelen tener engranajes metálicos.

No se dedicará espacio en este documento a los múltiples sitios donde se explican conceptos de manera errónea o confusa para explicar el PWM de los servos. Solo como ejemplo: en varias explicaciones se puede entender que el ancho de pulso máximo para un motor servo es igual al período de la señal (como se puede entender conceptualmente la definición de PWM en general) y no hay registro de ningún control de motor servo que así lo implemente (entre los más comunes, al menos). Estas diferencias de tiempos son de alguna manera "aprovechadas" por otras formas de codificación de señales muy similares a PWM, como PPM (Pulse Position Modulation), sobre todo en el ámbito de las comunicaciones de control de drones [11] para "concatenar" la codificación de varias señales en un mismo ciclo. Sin embargo, la explicación de PPM está fuera del alcance de este documento.

2.- Biblioteca Básica: IDE y Placa Arduino

Utilizando la biblioteca servo en Arduino, se puede ver el funcionamiento de los motores servo LD-2015, que se utilizan en algunos robots humanoides como el de la Fig. 1: un robot humanoide de 17 grados de libertad, con 15 servos LD-2015 y 2 servos LD-2701. Ya en términos de funcionamiento controlado desde un microcontrolador, la Fig. 2 muestra la posición del servo cuando se le indica una posición de 5º desde una placa Arduino, vía la biblioteca estándar de servos (servo.h). Específicamente, se utiliza uno de los pines identificados para PWM y la función write(),

que genera la señal PWM correspondiente al valor que se le pasa como parámetro entre 0° y 180°, que son tomados como grados de posición del servo [3].

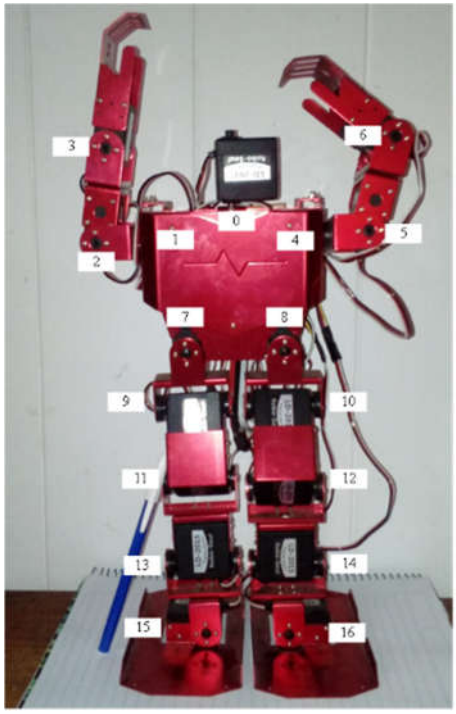


Figura 1: Robot Humanoide de 17 Grados de Libertad.

Figura 2: Servo 5º - Arduino.

La señal PWM generada desde Arduino para la posición de 5º del servo se muestra en la Fig. 3, donde:

- La frecuencia es de 50Hz (1 valor cada 20ms, considerado usual para servos).
- La tensión de la señal es de 5V (valor de las señales digitales en Arduino).
- Sin entrar en detalles de precisión, el ancho de pulso parece estar por debajo de 1ms.

La Fig. 4 muestra la posición del servo cuando se le indica una posición de 175º vía la biblioteca de Arduino. Sin comentar sobre la exactitud o el rango y las posiciones de 5º - 175º, se puede ver que el servo se mueve entre los extremos de las posiciones posibles y no necesariamente se llega a tener un rango completo de 180º, aunque es cercano.

Figura 3: PWM 5º - Arduino.

Figura 4: Servo 175º - Arduino.

La señal PWM generada desde Arduino para la posición de 175º del servo se muestra en la Fig. 5, donde:

- La frecuencia se mantiene en 50Hz (1 valor cada 20ms, considerado usual para servos), como es de esperar, dado que ya fue identificado en la Fig. 2.
- La tensión de la señal es de 5V (valor de las señales digitales en Arduino), también como es de esperar en Arduino e identificado en la Fig. 2.
- Sin entrar en detalles de precisión, el ancho de pulso parece ser de poco más de 2ms.

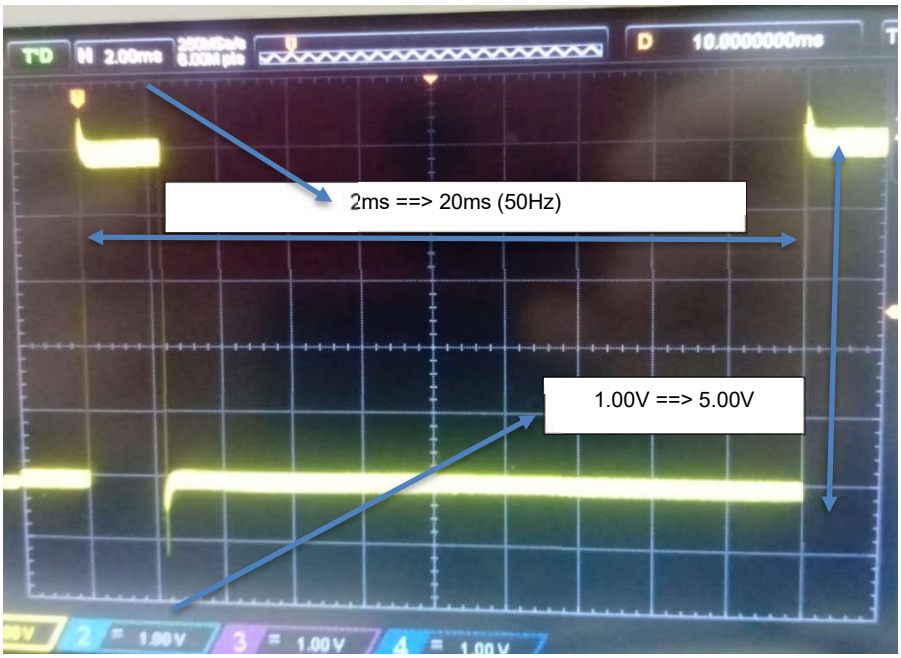


Figura 5: PWM 175º - Arduino.

Es posible que utilizando la función writeMicroseconds() de la biblioteca servo se pueda lograr una amplitud de movimientos más cercana a 0º - 180º. De este modo, con un "ajuste fino" en cierto modo se termina directamente manejando el ciclo de trabajo del servo (la propia señal PWM), pero para la explicación en este documento no es necesario. Además, el objetivo es mantener el mayor nivel de abstracción posible para el uso de la biblioteca: posición en grados del servo.

3.- Biblioteca Básica: IDE Arduino y Placa con ESP8266 NodeMCU

Utilizando la misma biblioteca e IDE del caso anterior, pero cambiando el hardware, hay que tener en cuenta los detalles específicos que corresponden para la generación de la señal PWM. En particular, se tiene en cuenta que el ESP8266EX es el SoC (System on Chip) [6] que implementa PWM por hardware en 4 de sus pines de entrada/salida de propósito general (o GPIO, del inglés General Purpose Input/Output) [7]. Sin embargo, algunos sitios indican de manera explícita que el ESP8266 no tiene hardware para PWM [8], no queda claro si es información desactualizada o directamente errónea desde el primer SoC ESP8266. Es importante notar que el ESP8266EX es la base de muchas implementaciones de placas de desarrollo [17].

Es interesante notar que las placas NodeMCU [14] permiten utilizar cualquiera de sus pines para PWM y todo indicaría que se aprovecha el hardware si es posible y en los demás casos se

implementa por hardware. Sin embargo, y aunque están documentados los pines con PWM "por hardware" y la referencia a un sistema operativo de tiempo real [7], no se indica/documenta explícitamente la diferencia para el usuario. Es decir: cuando en un sketch se utiliza la biblioteca servo queda abierta la posibilidad de que en todos los casos la señal de PWM termine siendo implementada completamente por software. La Fig. 6 muestra una placa NodeMCU y en particular la identificación de los pines/GPIO asociados al PWM por hardware en el ESP8266 (cuatro pines, identificados con ~), uno de los cuales se utilizará el los experimentos que se muestran en este documento a continuación.

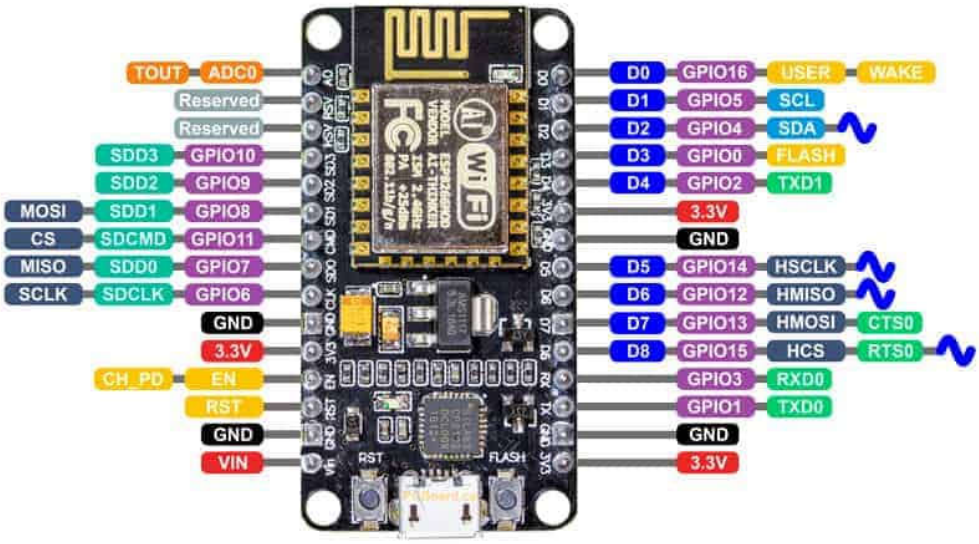
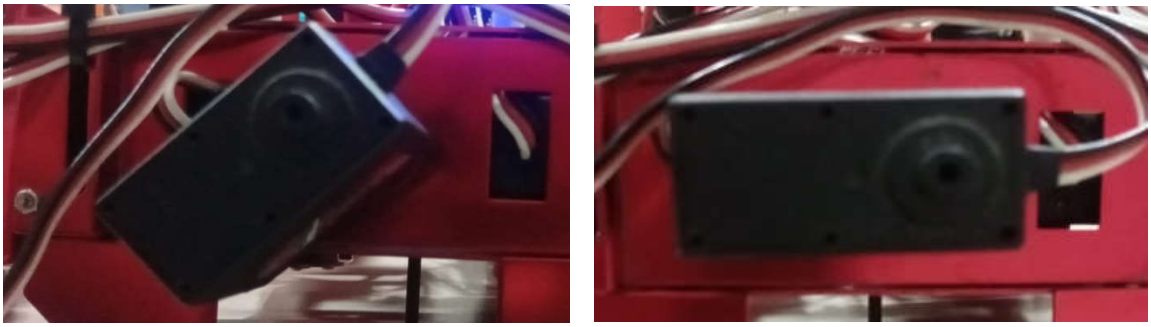


Figura 6: Pines de la Placa NodeMCU.

Ya en términos de funcionamiento controlado desde una placa NodeMCU, la Fig. 7a) muestra la posición del servo cuando se le indica una posición de 5º vía la biblioteca estándar de servos (servo.h). Específicamente, se utiliza uno de los pines identificados para PWM y la función write(), que genera la señal PWM correspondiente al valor que se le pasa como parámetro entre 0° y 180°, que son tomados como grados de posición del servo [3]. Para mejorar la comparación entre placas, se incluye la imagen de la misma posición (5°) indicada desde Arduino en la Fig. 7b). Claramente, aunque las posiciones tienen las mismas orientaciones, los ángulos son diferentes.

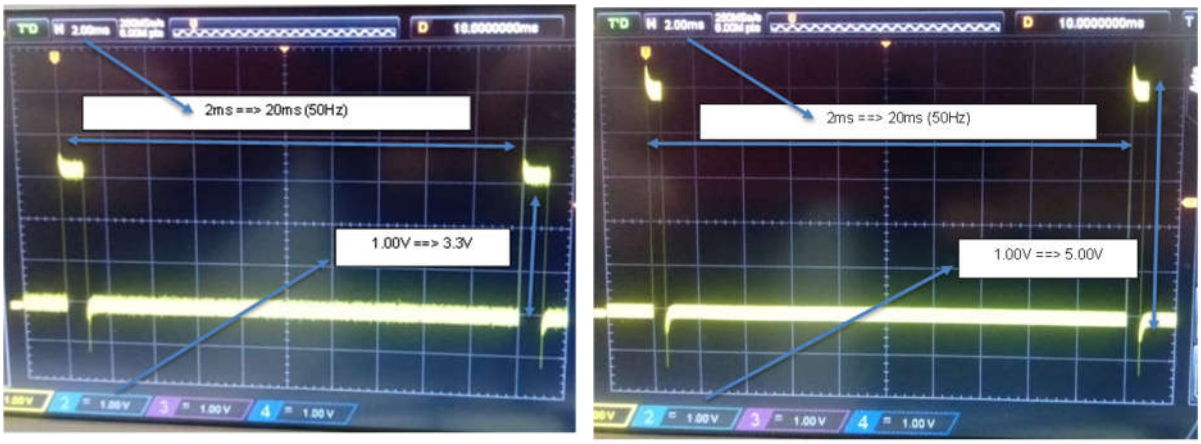
La señal PWM generada desde NodeMCU para la posición de 5º del servo se muestra en la Fig. 8a) y como en el caso anterior, se incluye la imagen de la señal para la misma posición con Arduino en la Fig. 8b). Comparando las señales PWM generadas desde NodeMCU (ESP8266) y Arduino, las dos diferencias más notables son el ancho de la señal y la amplitud/tensión (3.3V vs. 5V). Lo que se mantiene de manera bastante similar (o igual, de hecho) es la frecuencia: 50Hz, un pulso (o "dato") cada 20ms. Es posible que al ser el ancho de la señal generado por la NodeMCU mayor que el generado por Arduino se pueda justificar que el ángulo es mayor al que genera la señal de Arduino, pero esto debería confirmarse con el otro "extremo", en los ángulos cercanos a 180º. La

otra posibilidad es que la diferencia de tensión (NodeMCU genera señales de 3.3V y Arduino 5V) sea la que determine un movimiento de menor ángulo del servo.



a) NodeMCU b) Arduino

Figura 7: Servo 5º - IDE Arduino: NodeMCU vs. Arduino.



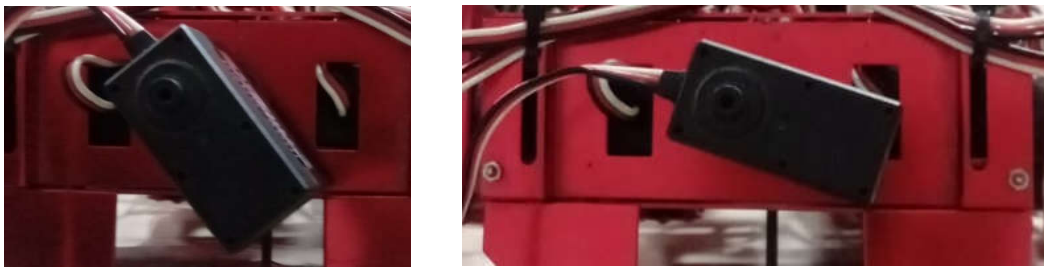
a) NodeMCU b) Arduino

Figura 8: PWM 5º - IDE Arduino: NodeMCU vs. Arduino.

La Fig. 9a) muestra la posición del servo cuando se le indica una posición de 175º vía la biblioteca estándar de servos (servo.h). Específicamente, se utiliza uno de los pines identificados para PWM y la función write(), que genera la señal PWM correspondiente al valor que se le pasa como parámetro entre 0° y 180°, que son tomados como grados de posición del servo [3]. Para mejorar la comparación entre placas, se incluye la imagen de la misma posición (175°) indicada desde Arduino en la Fig. 9b). Una vez más, aunque las posiciones tienen las mismas orientaciones, los ángulos son notablemente diferentes.

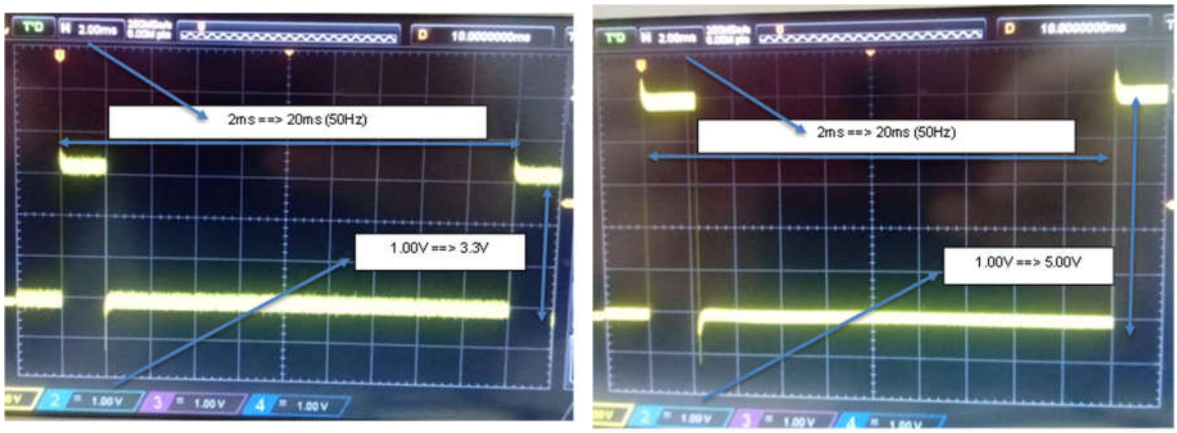
La señal PWM generada desde NodeMCU para la posición de 175º del servo se muestra en la Fig. 10a) y como en el caso anterior, se incluye la imagen de la señal para la misma posición con Arduino en la Fig. 10b). Como para el caso de la posición en 5º, comparando las señales PWM generadas desde NodeMCU (ESP8266) y Arduino, las dos diferencias más notables son el ancho

de la señal y la amplitud/tensión (3.3V vs. 5V). Lo que se mantiene de manera bastante similar (o igual) es la frecuencia: 50Hz, un pulso cada 20ms. Es posible en este caso que al ser el ancho de pulso generado por la NodeMCU menor que el generado por Arduino se pueda justificar que el ángulo es menor al que genera la señal de Arduino y también es posible que la diferencia de amplitud de la señal tenga algún efecto. A priori, al ser la posición proporcional al ancho de la señal, se podría directamente justificar la diferencia sin mencionar la diferencia de tensión de la señal PWM. De alguna manera, esto podría coincidir con lo indicado en el sitio [16]. Lo que se mantiene sin explicación ni documentación es por qué para la misma posición en grados indicada desde NodeMCU se genera un ancho diferente de la señal PWM comparándola con la de Arduino para los mismos valores.



a) NodeMCU b) Arduino

Figura 9: Servo 5º - IDE Arduino: NodeMCU vs. Arduino.



a) NodeMCU b) Arduino

Figura 10: PWM 175º - IDE Arduino: NodeMCU vs. Arduino.

4.- Resumen de Comparación Arduino-NodeMCU (ESP8266)

Si bien en la sección anterior se hizo una comparación caso por caso, la comparación en cuando a funcionalidad de la misma biblioteca del IDE de Arduino para la placa Arduino y la placa

NodeMCU utilizada se puede resumir en:

- La orientación de las posiciones es la misma, es decir las orientaciones de posiciones de menor y mayor grado son las mismas. Si se tuviera un brazo montado sobre el eje del servo, el posicionamiento en grados orientaría el brazo en la misma dirección tanto utilizando NodeMCU como utilizando Arduino.
- Las posiciones finales de los ejes del servo son diferentes. Específicamente el rango de posiciones de 0º a 175º utilizando NodeMCU es bastante menor al que se obtiene con Arduino. Es de notar que lo único diferente es la placa utilizada, tanto el IDE como la biblioteca como la alimentación son idénticos en todos los experimentos.
- La diferencia en posiciones finales para los rangos de posiciones de 0º a º180º puede considerarse proporcional y correspondiente a las señales PWM generadas desde cada placa. No se hicieron mediciones al detalle para justificar esta afirmación de manera exhaustiva/objetiva, pero el análisis visual podría al menos considerarse orientado a esa conclusión.
- No está claro por qué la señal PWM generada desde NodeMCU (ESP8266) es diferente de la generada desde Arduino. Es de esperar que los motores difieran entre sí, pero no la amplitud de la señal PWM para una misma posición en grados.
- No está claro si la diferencia de tensión Arduino-NodeMCU (ESP8266) en la señal de PWM tiene impacto o no en el control de posición del servo.
- Considerando que Arduino obtiene un rango de posiciones bastante más cercano al esperado para los valores de 5º a 175º, no está claro por qué la placa NodeMCU genera señales PWM bastante diferentes a las que genera Arduino.
- No se mostraron en este documento, pero la posición correspondiente a los 90º en ambas placas en casi idéntica.

La Fig. 11 muestra esquemáticamente un resumen de la comparación de posiciones del eje del servo controlado por ambas placas.

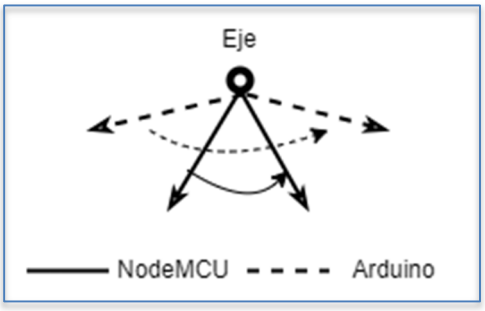


Figura 11: Comparación de Posiciones de Servo: Arduino vs. NodeMCU.

En todos los casos se mantuvo la idea de manejar el servo con el mayor nivel de abstracción posible desde el código fuente, es decir simplemente usando la función de posicionamiento en grados del eje del servo, vía la función de biblioteca write(). Las relativamente grandes diferencias de

amplitud de movimientos indican que no es un método confiable en general y es posible que como mínimo sea necesaria una calibración en el código fuente para cada servo y cada placa que se decida utilizar. Esto genera que para evitar esta labor que en cierto modo es peligrosa o al menos puede generar errores y consume tiempo, se recurra a placas de manejo de servos, sobre todo para los casos como el del robot de la Fig. 1. Cuando se requiere el manejo de múltiples servos "simultáneamente" muchas placas de desarrollo como Arduino o NodeMCU ni siquiera tienen la cantidad de pines disponibles pero, aunque los tuvieran, tampoco sería bueno dedicar tanto esfuerzo a replicar la generación *confiable* de señales de PWM para todos los servos (con la consiguiente multiplicación de requerimientos de tiempo real).

Las placas de control de múltiples servos no necesariamente evitan tareas como la calibración, dado que depende en gran medida del tipo de servo a controlar y eso no se puede tener en cuenta a priori en cada una de esas placas, pero:

- Simplifica toda la tarea/programación de control de todos los servos a utilizar, reduciéndola a programar/calibrar lo necesario en una única placa de control de múltiples servos.
- Descargar automáticamente a los microcontroladores del procesamiento de generación de señales de PWM, que directamente llevará a cabo la placa elegida.

5.- Placa de Control de Servos: NodeMCU+PCA9685

Si bien no necesariamente hay muchas placas para control de múltiples servos, existen empresas que proveen placas (módulos o *breakouts*) con las que se pueden manejar, por ejemplo, 16, 24, o 32 motores [9] [10] [12] [4], la mayoría con conexión serie o I^2C a una placa de desarrollo o directamente a un microcontrolador. Estas placas proveen dos ventajas muy útiles para las aplicaciones con múltiples servos:

1. Con muy pocos pines de comunicación (por ejemplo, tanto conexiones vía UART como I^2C involucran básicamente 2 pines) se manejan 32 servos, por ejemplo.
2. La generación simultánea de todas las señales PWM, para las cuales los microcontroladores no tienen que intervenir en ningún aspecto.

Dependiendo de las placas, algunas proveen bibliotecas relativamente complejas, con funciones donde se parametrizan no solamente movimientos individuales de servos sino conjuntos de movimientos y tiempo que se debe emplear para cada movimiento o para el conjunto completo de movimientos.

En particular, la placa conocida como PCA9685 puede considerarse especial. Sin lugar a dudas, es a la que hacen referencia la mayoría de los sitios web donde se explica el manejo de múltiples servos con Arduino o microcontroladores en general. Sin embargo, también en la mayoría de los casos no se utiliza la que podríamos considerar como PCA9685 *original*, que es de la empresa Adafruit. La gran mayoría en realidad son clones o copias de la *original*, que tienen el mismo

funcionamiento y hasta las mismas dimensiones, aunque cambian los colores utilizados en algunas partes físicas de cada placa (usualmente: colores de pines de conexión para los servos) [13]. También en general, las placas que se podrían considerar clones ya tienen incluido un capacitor que se puede considerar relativamente grande y que la original no incluye (solo tiene el espacio para incluirlo en los casos para los cuales el desarrollador considere necesarios). Quizás las dos razones más importantes para el uso de clones de PCA9685 son:

1. El muy bajo costo relativo de los clones, y mucho más aún comparado con las placas de otras empresas que proveen placas de control de múltiples servos.
2. La relativamente alta confiabilidad al menos en términos funcionales de todos los clones, que tienen el mismo funcionamiento que el de la original.

Una de las posibles razones por las cuales la gran mayoría de los sitios con explicaciones de control de múltiples servos utilizan la PCA9685 es el muy bajo costo de los clones, que suele ser de hasta diez veces menos costo que las demás placas (incluida la original de Adafruit).

En la gran mayoría de los casos, las explicaciones disponibles para utilizar una PCA9685, incluyendo la propia documentación de Adafruit, involucra una combinación de:

- Una placa de desarrollo Arduino o un microcontrolador similar al de una placa Arduino
- La biblioteca provista por Adafruit: Adafruit_PWMServoDriver.h
- Alguno de los ejemplos de la propia biblioteca de Adafruit

Para el caso particular de los motores a los que se hace referencia en este documento, la PCA9685 tiene un inconveniente particular: la alimentación de motores se documenta con un máximo de 6V y sin especificación de límite de intensidad (Amperes) o energía (Watts). Sin explicar en detalle los problemas generados por esta limitación para sistemas con estos servos, se pueden resumir con:

- Los servos de varios kg de torque usualmente se alimentan con baterías LiPo de 2 ó 3 celdas, lo cual implica tener alimentación “básica” de 7.4V o 11.1V.
- La reducción de tensión de 7.4V o 11.1V a 6V o menos no solamente implica pérdida de torque de los servos sino que suele aumentar el costo del sistema, principalmente cuando el consumo es de mucha intensidad (Amperes).
- Los motores con los cuales se construye el robot con los cuales se llevan a cabo los experimentos en este documento requieren alimentación de hasta 7.4V, posiblemente dimensionados para aplicaciones con alimentación a base de baterías LiPo de 2 celdas.

Un inconveniente diferente del de la alimentación de la PCA9685 es el relativamente bajo nivel de abstracción de la biblioteca y ejemplos de Adafruit. En todos los casos se hace referencia o directamente se deben utilizar funciones de biblioteca que implican conocer detalles de frecuencia de PWM, frecuencia de reloj de referencia para las comunicaciones y la relación de valores de PWM en milisegundos para cada posición del servo.

El sistema al que se orienta el trabajo con múltiples servos es el del robot de la Fig. 1, y en ese sistema se pretende el control del usuario via WiFi/web del sistema. La placa a utilizar en principio será del tipo de la NodeMCU, sea como la ya documentada con ESP8266 o directamente con

ESP32. Dado que estas placas con microcontroladores, como muchas similares tiene GPIO de 3.3V se llevan a cabo algunos experimentos para probar que el sistema completo de la Fig. 12 es funcionalmente correcto cuando se conecta un servo.

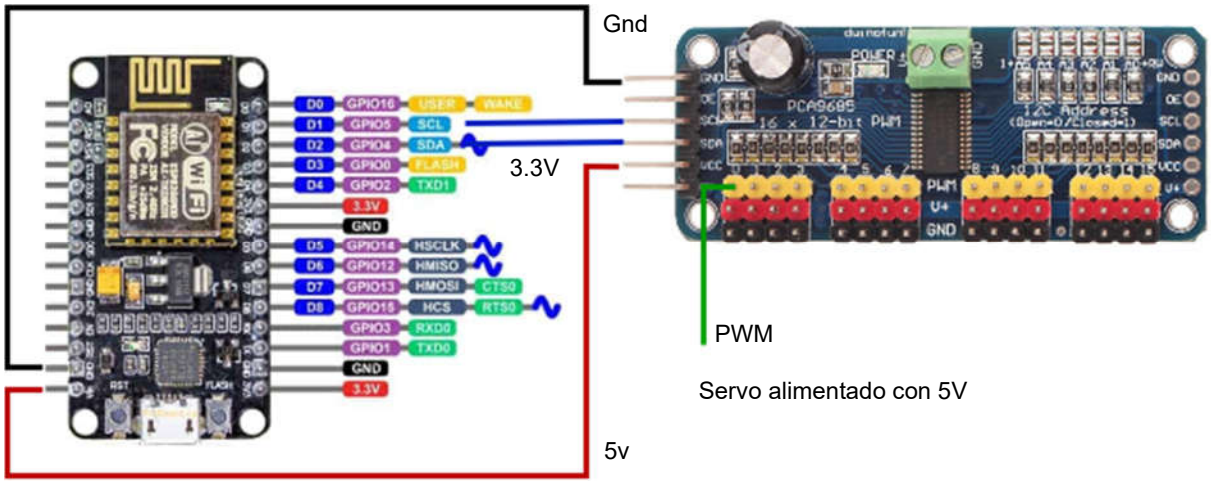


Figura 12: NodeMCU - PCA9685.

La Fig. 13 muestra con el osciloscopio que la señal generada por la PCA es de 5v, con lo que no se tendrían distintos niveles de tensión en el servo. Además, el rango de posiciones 5° - 175° se puede considerar correcto, aunque siempre se puede recurrir a la generación de PWM para mejorar el rango y precisión de las posiciones del servo. Es decir: la placa PCA9685 interpreta correctamente las señales SCL-SDA de las comunicaciones I^2C aunque se generen con 3.3V en la NodeMCU-ESP8266 y como está alimentada por 5V, la señal de servo que genera tiene nivel de 5V. El servo está alimentado con los mismos 5V con los que se alimenta a la NodeMCU y a la PCA9685.

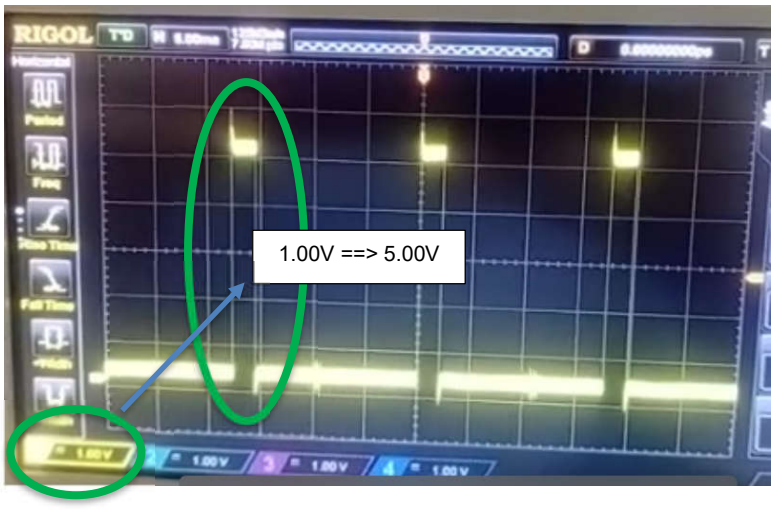


Figura 13: Señal PWM Generada por PCA9685.

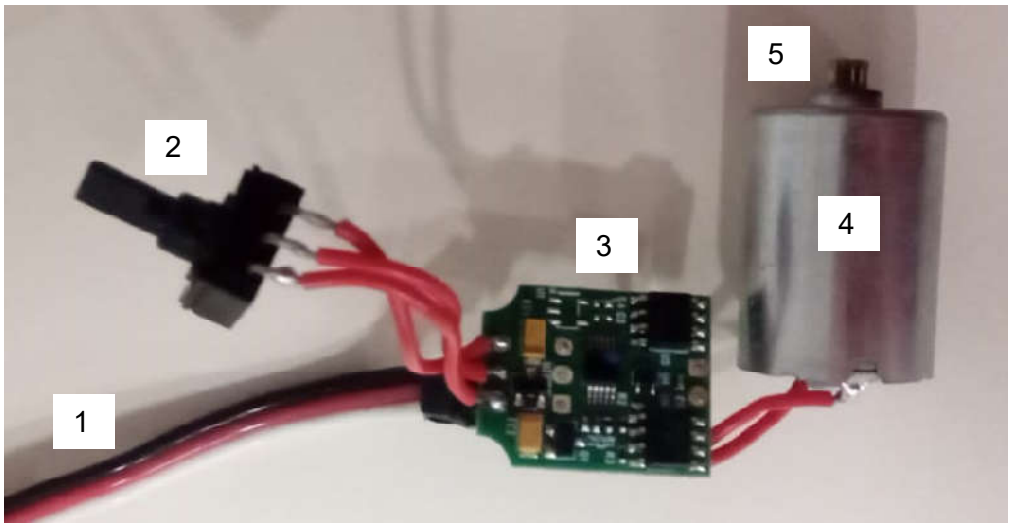
Queda pendiente la verificación del funcionamiento de los servos para el caso en que se los alimenta con 7.4V cuando la señal PWM tiene amplitud de 5V. Aunque es de esperar que el servo funcione correctamente y se posicione teniendo en cuenta solo la señal PWM, queda por confirmar esa suposición con los experimentos correspondientes.

6.- Conclusiones

El control de motores servos tiene una base conceptual clara y la mayoría de los motores responden al mismo tipo de señales/codificación para el posicionamiento del eje. Sin embargo, las posiciones específicas del eje usualmente varían en función de la señal PWM que reciben por la conexión específica de control. No está claro si la diferencia de tensión de las señales de alimentación y control afectan el posicionamiento del eje de cada servo, aunque algunos sitios/documentación afirman que se pueden considerar independientes dentro de rangos o umbrales de tensión "razonables". Se muestra experimentalmente que aún implementaciones de la misma biblioteca servo de Arduino para diferentes plataformas puede tener sus propias diferencias que se verán reflejadas en las aplicaciones finales y por lo tanto se debe prestar mucha atención a las tareas de calibración. Las placas de control de múltiples servos no solamente permiten dedicar pocos pines de los microcontroladores para manejar varios servos, sino que desacoplan y uniformizan el control de estos motores, simplificando mucho la implementación de aplicaciones.

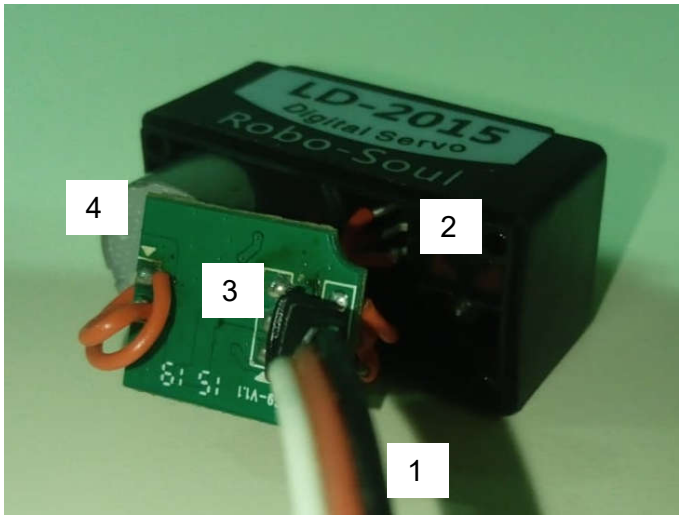
Apéndice: Detalles Mecánicos del LD-2015

Aunque no es esencial el conocimiento mecánico ni las partes de un servo, se muestran a continuación algunas imágenes de un servo LD-2015 que pueden ser útiles para conocer algunos detalles. En la imagen a continuación se muestran las partes más "internas" del servo:



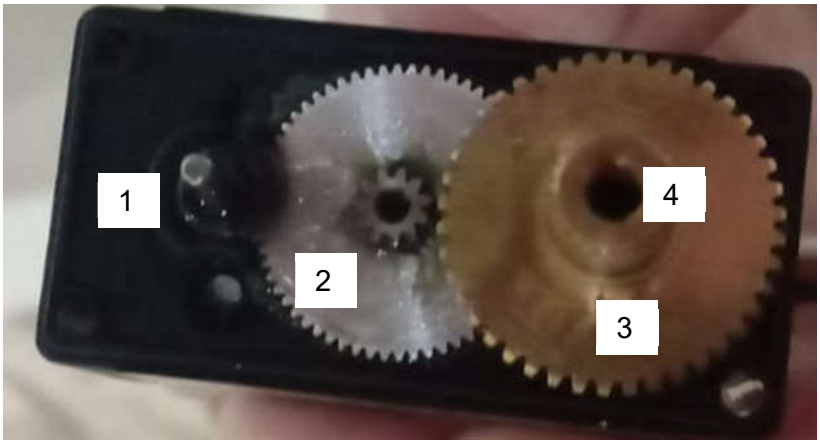
1: Cable de alimentación (Vcc-Gnd: rojo y negro respectivamente) y control (PWM: blanco) del servo.
2: Potenciómetro, usado para definir la posición del servo.
3: Placa de control, donde usualmente se tiene un Puente H para definir el sentido de movimiento del motor y un comparador de señales que conceptualmente utiliza las señales de PWM y del potenciómetro para fijar la posición final del eje del servo.
4: Motor de corriente continua (CC).
5: Engranaje del eje del motor, que integrará a la serie de engranajes de la caja reductora del servo.

En la imagen anterior no se muestra el montaje del servo ni la caja reductora. El montaje de las partes mostradas en la imagen anterior se muestra a continuación, en lo que sería la cara opuesta del servo respecto de la que tiene el eje al exterior. Se utilizan los números de identificación de cada parte mostrados anteriormente. La perilla (*knob*) del potenciómetro es efectivamente el eje del servo, tal como se identificará más explícitamente en imágenes siguientes. En esta imagen el potenciómetro queda casi completamente cubierto para mostrar las demás partes del servo.



En la imagen que se muestra a continuación se muestra la parte interna de la cara que contiene el eje que sale al exterior del servo y donde no se muestra uno de los engranajes de la caja reductora para visualizar mejor las partes:
1: Engranaje del eje del motor de CC, que se acopla al primer engranaje de la caja reductora.
2: Primer engranaje de la caja reductora que se acopla al engranaje del motor y al segundo engranaje de la caja reductora, que no aparece en esta imagen.
3: Engranaje “final” de la caja reductora, que es la que finalmente determina la posición final del eje del servo, porque se encastra directamente la perilla del potenciómetro

4: Encastre de la perilla del potenciómetro en el engranaje final de la caja reductora.



En la siguiente imagen se muestra la caja reductora completa, que es la que transfiere el movimiento desde el motor de CC al eje del servo de la que dependen gran parte del torque y velocidad de movimiento del servo. En este caso, todos los engranajes de la caja reductora son metálicos, en los servos de menor torque y costo el material de todos los engranajes suele ser plástico.

Referencias

[1] Arduino, Servo Motor Basics with Arduino, Rev. 14/06/2023, https://docs.arduino.cc/learn/electronics/servo-motors

[2] Arduino, Servo - Arduino Reference, https://reference.arduino.cc/reference/en/libraries/servo/

[3] Arduino, Servo - write() - Arduino Reference,
https://reference.arduino.cc/reference/en/libraries/servo/write/

[4] B. Earl, Adafruit PCA9685 16-Channel Servo Driver, 2012,
https://learn.adafruit.com/16-channel-pwm-servo-driver

[5] B. Earl, Adafruit PCA9685 16-Channel Servo Driver, Update 2023-01-20,
https://cdn-learn.adafruit.com/downloads/pdf/16-channel-pwm-servo-driver.pdf

[6] Espressif Systems, Chipsets | Espressif Systems, 2015-2023,
https://www.espressif.com/en/products/socs

[7] Espressif Systems, ESP8266EX Datasheet, Version 6.9, 2023.

[8] I. Grokhotkov, Reference - ESP8266 Arduino Core 3.1.2-13-g8b33e2e2 documentation,
2017, https://arduino-esp8266.readthedocs.io/en/latest/reference.html

[9] Hiwonder, Servo Controller, 2023,
https://www.hiwonder.com/collections/servo-controller

[10] Hiwonder, LSC-32, 2023,
https://www.hiwonder.com.cn/store/learn/24.html

[11] O. Liang, PWM and PPM Difference and Conversion, 5th November 2013,
https://oscarliang.com/pwm-ppm-difference-conversion/

[12] L. Llamas, “Hasta 32 de servos en Arduino con el controlador USC-32”, 2016,
https://www.luisllamas.es/hasta-32-de-servos-en-arduino-con-el-controlador-usc-32/

[13] L. Llamas, “Controlar 16 servos o 16 salidas PWM en Arduino con PCA9685”, 2016,
https://www.luisllamas.es/controlar-16-servos-o-16-salidas-pwm-en-arduino-con-pca9685/

[14] NodeMcu Team, NodeMcu -- An open-source firmware based on ESP8266 wifi-soc, 2014-
2018, https://www.nodemcu.com/index_en.html

[15] Open Hardware Design Group LLC, “What is a Breakout Board for Arduino?”,
https://www.programmingelectronics.com/what-is-a-breakout-board-for-arduino/

[16] Pololu Blog, Servo control interface in detail, 9 February 2011,
https://www.pololu.com/blog/17/servo-control-interface-in-detail

[17] Spacehuhn Blog, NodeMCU, ESP12, ESP8266 - What is the difference?, 2023, https://blog.spacehuhn.com/nodemcu-vs-esp8266