# ESP32 con Servomotores Dynamixel

## Servomotores Dynamixel

Los servos Dynamixel (véase en la figura 1) son actuadores inteligentes desarrollados por ROBOTIS que integran un motor, un controlador y un sistema de retroalimentación en un solo dispositivo. Son ampliamente utilizados en robótica debido a su capacidad para proporcionar movimientos precisos, configurables y de alto torque, ideales para aplicaciones como brazos robóticos, robots humanoides y sistemas de automatización. Controlar estos servos es fundamental para aprovechar sus funciones avanzadas, como el monitoreo en tiempo real de posición, velocidad, temperatura y carga. Además, su protocolo de comunicación permite la conexión de múltiples unidades en red, facilitando el desarrollo de sistemas robóticos complejos y flexibles. Aprender a controlarlos es esencial para diseñar soluciones robustas y eficientes en proyectos de robótica avanzada.

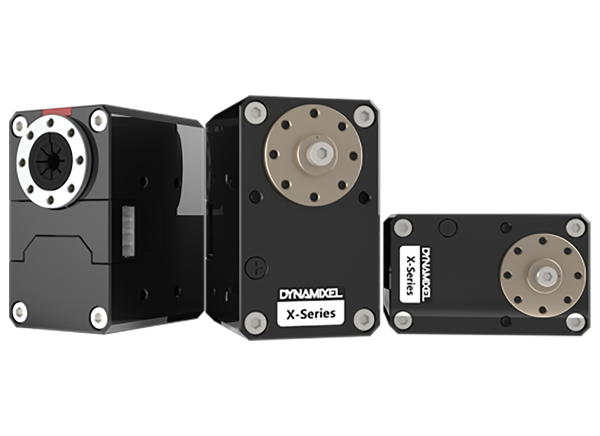


Figura 1. Foto de Servomotores Dynamixel [4]

**Dynamixel2Arduino**

La librería Dynamixel2Arduino [3] y DYNAMIXEL Shield son herramientas indispensables a la hora de controlar a la familia Dynamixel.

La biblioteca DynamixelShield es una extensión específica para la placa de expansión DYNAMIXEL Shield, diseñada para facilitar la integración de servos Dynamixel con placas Arduino. Esta biblioteca hereda las funcionalidades de la biblioteca Dynamixel2Arduino, adaptándolas para su uso con la DYNAMIXEL Shield. Es importante destacar que, para utilizar la biblioteca DynamixelShield, es necesario tener instalada previamente la biblioteca Dynamixel2Arduino.

Por otro lado, la biblioteca Dynamixel2Arduino es una biblioteca de código abierto que proporciona una interfaz sencilla para controlar y monitorear servos Dynamixel en proyectos con Arduino. Es independiente de la DYNAMIXEL Shield y puede utilizarse en una variedad de plataformas y configuraciones de hardware.

## Instalación

Para la instalación de la librería se puede utilizar el gestor de bibliotecas de Arduino IDE, buscando la palabra “Dynamixel” te aparecerá opciones y debemos dar click en instalar en la biblioteca Dynamixel2Arduino [3], adicionalmente podemos instalar la biblioteca Dynamixel Shield como se ve en la figura 2.

Texto

Descripción generada automáticamente

Figura 2. Menú de gestión de bibliotecas en Arduino IDE

## DYNAMIXEL Shield MKR

El **DYNAMIXEL Shield MKR** es una placa de expansión diseñada para simplificar la integración de servos DYNAMIXEL en proyectos basados en placas Arduino MKR. Este *Shield* permite controlar hasta 253 servos conectados en cadena, brindando retroalimentación en tiempo real sobre parámetros como posición, velocidad, voltaje, corriente, temperatura y estado de movimiento. Además, ofrece la capacidad de ajustar las características de movimiento y ampliar sistemas existentes, o incluso reemplazar unidades por modelos más avanzados o similares.

Aunque la DYNAMIXEL *Shield* está optimizada para las placas Arduino MKR, también puede utilizarse con una placa ESP32 mediante algunas conexiones adicionales. Para ello, es necesario conectar el pin TX de la ESP32 al pin TX de la *Shield* y el pin RX de la ESP32 al pin RX de la *Shield*. Asimismo, se debe conectar un pin de control de flujo (por ejemplo, cualquier puerto disponible) al pin correspondiente en la Shield (A6). Es importante alimentar el pin VCC con 3.3V y el pin +5V con 5V para asegurar la alimentación de los circuitos lógicos de la *Shield*. Por último, el servomotor debe ser alimentado con 12V a través del conector VIN, ya sea utilizando el pin VIN o el terminal de tornillo verde visible en la figura 3.

Es importante tener en cuenta que la salida hacia los servos Dynamixel es TTL. Este detalle es relevante porque los servomotores pueden comunicarse mediante TTL o RS-485. Para identificar el tipo de comunicación de nuestro servo, basta con observar si tiene 3 pines (TTL) o 4 pines (RS-485). En el caso de utilizar servos con RS-485, es posible adquirir módulos convertidores de RS-485 a TTL.

Diagrama, Esquemático

Descripción generada automáticamente

Figura 3. Pin out del DYNAMIXEL Shield MKR [2]

## Circuito TTL Alternativo

Si no disponemos del **DYNAMIXEL Shield MKR**, podemos crear un circuito alternativo para establecer la comunicación. Es importante tener en cuenta que debemos incluir un convertidor de nivel lógico (de 3.3V a 5V), ya que la salida UART del ESP32 opera a 3.3V, mientras que el integrado de la figura 4, requiere una señal de 5V para funcionar correctamente.Diagrama

Descripción generada automáticamente

Figura 4. Circuito comunicación TTL [1]

## Programación de configuración Dynamixel

Para poder configurar el servo XM430-210T con la ESP32 debemos primero incluir la librería, encargada de manejar la comunicación entre el microcontrolador (en este caso, el **ESP32**) y el servo **Dynamixel**.

#include <**Dynamixel2Arduino**.h>

Además, debemos definir los pines para la comunicación UART entre la ESP32 y el *Shield* (o circuito de comunicación) con las siguientes líneas:

// Definición de los pines RX y TX

const int RX\_PIN = 16;

const int TX\_PIN = 17;

// Definición del pin de dirección para el bus half-duplex

const int DXL\_DIR\_PIN = 5; // Ajusta según tu configuración

Los pines **RX\_PIN** y **TX\_PIN son utilizados para la comunicación UART y el DXL\_DIR\_PIN** es utilizado paracontrolar la dirección de comunicación en el bus **half-duplex**, permitiendo que los datos viajen en una sola dirección en un momento dado.

Ahora debemos configuración el puerto serial, ya que utilizamos la ESP32, nos permite utilizar varios puertos serial, para comunicarnos con el Dynamixel, utilizaremos el Serial2, para declararlo debemos agregar esta línea de código

// Inicialización del puerto serial para Dynamixel

HardwareSerial& DXL\_SERIAL = **Serial2**;

Después hacemos una instancia de la clase Dynamixel2Arduino, asociando el puerto serial y el pin de dirección para realizar la comunicación de la siguiente manera:

// Creación de la instancia de Dynamixel2Arduino

**Dynamixel2Arduino** dxl(DXL\_SERIAL, DXL\_DIR\_PIN);

Ahora declaramos los parámetros de Protocolo utilizado, Baudrate y el ID del servomotor, en caso de que tengamos estos parámetros debemos declararlos de la siguiente manera:

// Parámetros del Dynamixel

const uint8\_t DXL\_ID = 1;

const float DXL\_PROTOCOL\_VERSION = 2.0;

const int DXL\_Baud = 57600;

Además, debemos escribir esta línea para acceder a la tabla de control del servo Dynamixel, para poder modificar algunos parámetros como *GoalPosition, PresentPosition, TorqueEnable,* etc.

//This namespace is required to use Control table item names

using namespace ControlTableItem;

Dentro del *setup()*, debemos inicializar la comunicación con el servo a partir de los parámetros del servomotor:

dxl.begin(DXL\_Baud);

dxl.setPortProtocolVersion(DXL\_PROTOCOL\_VERSION);

dxl.ping(DXL\_ID);

Después debemos desactivar el torque del motor para poder acceder y modificar la configuración del servo, ya que podemos realizar un control de posición, velocidad, posición PID, posición-Torque, etc. Para ello debemos agregar las siguientes líneas:

dxl.torqueOff(found\_id);

dxl.setOperatingMode(found\_id, OP\_POSITION);

dxl.torqueOn(found\_id);

En caso de que quiera implementar un controlador PID debemos agregar esta línea de código para declarar sus valores PID.

// Aplicar valores iniciales del PID

dxl.writeControlTableItem(POSITION\_P\_GAIN, DXL\_ID, position\_p\_gain);

dxl.writeControlTableItem(POSITION\_I\_GAIN, DXL\_ID, position\_i\_gain);

dxl.writeControlTableItem(POSITION\_D\_GAIN, DXL\_ID, position\_d\_gain);

## Subrutina escaneo

En caso de que no cuente con los parámetros de protocolo, baudrate e ID, podemos hacer una subrutina en el *setup()* utilizando la función *Scan()*, se mostrara a continuación un ejemplo de un posible algoritmo:

// Escaneo de baud rates y protocolos

for (int protocol = 1; protocol <= 2; protocol++) {

 dxl.setPortProtocolVersion((float)protocol);

**Serial**.print("Escaneando con protocolo ");

**Serial**.println(protocol);

 for (int i = 0; i < MAX\_BAUD; i++) {

**Serial**.print("Escaneando a baud rate: ");

**Serial**.println(baud[i]);

   dxl.begin(baud[i]);

   found=dxl.scan();

**Serial**.print("inicio Scan : ");

**Serial**.println(found);

   // Escaneo de IDs

   if (found==1){

           for (int id = 1; id < 253; id++) {

     if (dxl.ping(id)) {

**Serial**.print("Dynamixel encontrado - ID: ");

**Serial**.print(id);

**Serial**.print(", Número de modelo: ");

**Serial**.println(dxl.getModelNumber(id));

       found\_id = id;

       break;

     }

   }

   }

   if (found\_id != -1) break;

 }

 if (found\_id != -1) break;

}

## Programación de posición Dynamixel

Ya encontrado y configurado el Servo Dynamixel, podemos realizar el control de posición ya sea en el formato de la variable interna del servo (Figura 5) o en los grados (Figura 6) :

dxl.setGoalPosition(DXL\_ID, 512); // Posición en formato raw

Figura 5. Posición en Raw(variable interna del Dynamixel)

dxl.setGoalPosition(DXL\_ID, 90.0, UNIT\_DEGREE); // Posición en grados

Figura 6. Posición en grados

Adicionalmente podemos obtener la posición en tiempo real en formato de la variable interna o en grados con la siguiente línea de código:

dxl.getPresentPosition(DXL\_ID)

dxl.getPresentPosition(DXL\_ID, UNIT\_DEGREE)

## Programación con manejo de memoria

Los servomotores Dynamixel cuentan con una tabla de control (Control Table) ubicada en las memorias RAM y EEPROM. Esta estructura de datos permite gestionar todos los aspectos importantes del servomotor. Algunos registros son de lectura y escritura, identificados como RW (Leer-Escribir), y entre ellos se encuentran el ID, el Baudrate, la posición objetivo, entre otros. Por otro lado, existen registros de solo lectura, indicados como R (Leer), como el número de modelo, el protocolo (1.0 o 2.0) y variables relacionadas con el estado actual del movimiento, como la posición presente, entre otros.

Para poder manipular la Control Table debemos consultar la dirección y el tamaño de los datos que deseamos manipular (se manejan en bytes).

Tabla 1. Tabla de control de la EEPROM del AX18 [1]

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Tabla de control de la EEPROM del AX18** | | | | | |
| **Dirección** | **Tamaño (Byte)** | **Nombre del dato** | **Descripción** | **Tipo de acceso** | **Valor Inicial** |
| 0 | 2 | Model Number | Model Number | R | 18 |
| 2 | 1 | Firmware Version | Firmware Version | R | - |
| 3 | 1 | ID | DYNAMIXEL ID | RW | 1 |
| 4 | 1 | Baud Rate | Communication Speed | RW | 1 |
| 5 | 1 | Return Delay Time | Response Delay Time | RW | 250 |
| 6 | 2 | CW Angle Limit | Clockwise Angle Limit | RW | 0 |
| 8 | 2 | CCW Angle Limit | Counter-Clockwise Angle Limit | RW | 1023 |
| 11 | 1 | Temperature Limit | Maximum Internal Temperature Limit | RW | 75 |
| 12 | 1 | Min Voltage Limit | Minimum Input Voltage Limit | RW | 60 |
| 13 | 1 | Max Voltage Limit | Maximum Input Voltage Limit | RW | 140 |
| 14 | 2 | Max Torque | Maximun Torque | RW | 983 |
| 16 | 1 | Status Return Level | Select Types of Status Return | RW | 2 |
| 17 | 1 | Alarm LED | LED for Alarm | RW | 36 |
| 18 | 1 | Shutdown | Shutdown Error Information | RW | 36 |

## Manejo de Servomotor AX18

Para la codificación debemos realizar unos ajustes para el movimiento de un servo AX18, como la definición de las direcciones de los datos que deseamos manipular y leer, y su correspondiente tamaño de los datos al inicio del programa:

// Definición de direcciones de control

#define TORQUE\_ENABLE\_ADDR 24

#define CW\_ANGLE\_LIMIT\_ADDR 6

#define CCW\_ANGLE\_LIMIT\_ADDR 8

#define GOAL\_POSITION\_ADDR 30

#define LED\_ADDR 25

#define MOVING\_SPEED\_ADDR 32  // Registro para la velocidad de movimiento

// Definición de longitudes de los registros

#define TORQUE\_ENABLE\_ADDR\_LEN 1

#define ANGLE\_LIMIT\_ADDR\_LEN 2

#define GOAL\_POSITION\_ADDR\_LEN 2

#define LED\_ADDR\_LEN 1

#define MOVING\_SPEED\_ADDR\_LEN 2  // Longitud del registro de velocidad

// Tiempo de espera

#define TIMEOUT 1000

Adicionalmente debemos definir un tiempo máximo de espera para completar la operación y no considerarla como fallida, la llamaremos TIMEOUT y le asignaremos un 1 segundo (1000 ms), este parámetro es importante para las funciones *write ()* y *getPresentPosition ()*.

Antes de realizar cualquier cambio en los parámetros del servo, es necesario desactivar el torque de salida. Esto se logra utilizando la siguiente línea de código en el *setup()*:

dxl.write(DXL\_ID, TORQUE\_ENABLE\_ADDR, (uint8\_t\*)&turn\_off , TORQUE\_ENABLE\_ADDR\_LEN, TIMEOUT);

La función *write()* se usa para modificar registros de la memoria del servo Dynamixel. Los parámetros que requiere son:

* El ID del servo (DXL\_ID).
* La dirección del dato a modificar (TORQUE\_ENABLE\_ADDR).
* El valor que deseamos escribir (turn\_off), en este caso, desactivamos el torque.El formato es un int de 8 bits.
* La longitud del dato (TORQUE\_ENABLE\_ADDR\_LEN).
* Un tiempo de espera máximo (TIMEOUT) en milisegundos.

En este caso, la línea desactiva el torque del servo, permitiendo configuraciones sin resistencia mecánica.

La función nos devolverá un valor lógico, es decir, 1 (se pudo realizar la operación) o 0 (no se pudo realizar la operación), útil para ponerlo en una estructura condicional o algoritmo para validar si el comando fue ejecutado correctamente y, en caso contrario, tomar acciones como mostrar un mensaje de error, intentar de nuevo o detener el programa.

Después podemos definir los límites de los ángulos de nuestro servomotor, tanto en el sentido levógiro como el sentido dextrógiro con la siguiente línea:

dxl.write(DXL\_ID, CW\_ANGLE\_LIMIT\_ADDR, (uint8\_t\*)&goalPosition1, ANGLE\_LIMIT\_ADDR\_LEN, TIMEOUT);

dxl.write(DXL\_ID, CCW\_ANGLE\_LIMIT\_ADDR, (uint8\_t\*)&goalPosition2, ANGLE\_LIMIT\_ADDR\_LEN, TIMEOUT));

Para la definición de la velocidad, podemos declarar primero una variable de velocidad objetivo y escribir la función Write() para manipular la velocidad en el Control Table con las siguientes líneas:

uint16\_t speed = 50; //velocidad objetivo

dxl.write(DXL\_ID, MOVING\_SPEED\_ADDR, (uint8\_t\*)&speed, MOVING\_SPEED\_ADDR\_LEN, TIMEOUT));

Para finalizar la configuración debemos reactivar el torque de salida para permitir el movimiento del Servo.

dxl.write(DXL\_ID, TORQUE\_ENABLE\_ADDR, (uint8\_t\*)&turn\_on, TORQUE\_ENABLE\_ADDR\_LEN, TIMEOUT);

Podemos adicionalmente prender un led integrado en el servomotor para indicar alguna situación en la lógica. Para ello escribimos la siguiente línea:

dxl.write(DXL\_ID, LED\_ADDR, (uint8\_t\*)&turn\_on, LED\_ADDR\_LEN, TIMEOUT);

y un proceso similar para apagarlo:

dxl.write(DXL\_ID, LED\_ADDR, (uint8\_t\*)&turn\_off, LED\_ADDR\_LEN, TIMEOUT);

Ahora para poder mandar una posición objetivo debemos escribir la siguiente línea:

dxl.write(DXL\_ID, GOAL\_POSITION\_ADDR, (uint8\_t\*)&goalPosition1, GOAL\_POSITION\_ADDR\_LEN, TIMEOUT);

Para este caso, el control de posición se efectuará con la variable interna del encoder integrado en el servo(Raw).

## Referencias

[1] ROBOTIS, "AX-18A," ROBOTIS e-Manual. [En línea]. Disponible en: https://emanual.robotis.com/docs/en/dxl/ax/ax-18a

[2] ROBOTIS, "Arduino MKR Shield," ROBOTIS e-Manual. [En línea]. Disponible en: https://emanual.robotis.com/docs/en/parts/interface/mkr\_shield

[3] ROBOTIS-GIT, "Dynamixel2Arduino," GitHub repository, 2025. [En línea]. Disponible en: https://github.com/ROBOTIS-GIT/Dynamixel2Arduino

[4] Mouser Electronics, "Robotis Dynamixel X-Series modules," Mouser. [En línea]. Disponible en: https://www.mouser.mx/new/robotis/robotis-dynamixel-x-modules

[5] J. Lopez Pob, "SS," GitHub repository. [En línea]. Disponible en: https://github.com/JesusLopezPob/SS