Protocolo de Comunicación para el Sistema de Posicionamiento de COXIRIS

17 de marzo de 2025

Versión 1.0

Contenido

1 Introducción

Este documento detalla el protocolo de comunicación serial implementado para controlar el sistema de posicionamiento 3D de COXIRIS. El protocolo permite enviar comandos a través de una interfaz serial para realizar movimientos, consultar estados y diagnosticar errores en el sistema.

1.1 Configuración de la Comunicación

La comunicación se realiza a través del puerto serial con los siguientes parámetros:

Parámetro	Valor
Baud Rate	115200
Timeout	50 ms
Buffer de Comandos	64 bytes

1.2 Estructura General de los Comandos

Todos los comandos siguen la siguiente estructura:

```
COMANDO [parámetro1] [parámetro2] ... [parámetroN]
```

Características importantes:

- Los comandos no distinguen entre mayúsculas y minúsculas (se convierten a mayúsculas internamente), sin embargo el uso de mayúsculas es recomendado para distinguir los comandos de otros strings.
- Los espacios en blanco al inicio y al final son eliminados automáticamente.
- Los comandos deben terminar con un carácter de nueva línea (`\n` o `\r`).
- La longitud máxima de un comando (incluyendo parámetros) es de 63 caracteres.

1.3 Estructura de las Respuestas

Las respuestas siguen el siguiente formato:

```
ACK [COMANDO]
[Resultados específicos del comando, incluyendo errores, si aplica]
DONE [COMANDO] [: Datos adicionales, si aplica]
```

1.4 Estructura de los Mensajes de Error

Los errores siguen una estructura consistente para facilitar su interpretación:

```
ERROR: [mensaje]
```

Los errores siempre estarán entre un mensaje `ACK` y `DONE` por lo que pueden ser fácilmente asociados al comando al que pertenecen.

2 Comandos del Protocolo

2.1 Comando HELP

Formato:	`HELP`
Parámetros:	Ninguno
Descripción:	Muestra un mensaje de ayuda con todos los comandos disponibles.

Respuesta:	`ACK HELP`
	`[ERROR: error_description] (Si hay errores)`
	`[Lista con todos los comandos disponibles junto con su descripción]`
	`DONE HELP`

2.2 Comando SET_HOME

Formato:	`SET_HOME`
Parámetros:	Ninguno
Descripción:	Establece la posición actual como el origen del sistema de coordenadas (0,0,0).

Respuesta:	`ACK SET_HOME`
	`[ERROR: error_description] (Si hay errores)`
	`DONE SET_HOME`

2.3 Comando GO_HOME

Formato:	`GO_HOME`
Parámetros:	Ninguno
Descripción:	Mueve el sistema a la posición de origen (0,0,0).

Respuesta:	`ACK GO_HOME`
	`[ERROR: error_description] (Si hay errores)`
	`DONE GO_HOME`

2.4 Comando ABSOLUTE_MOVE

Formato:	`ABSOLUTE_MOVE x y z`
Parámetros:	• `x`: Coordenada X absoluta (número decimal) • `y`: Coordenada Y absoluta (número decimal) • `z`: Coordenada Z absoluta (número decimal)
Descripción:	Mueve el sistema a la posición absoluta especificada por las coordenadas X, Y, Z.

Respuesta Exitosa:	`ACK ABSOLUTE_MOVE`
	`[ERROR: error_description] (Si hay errores)`
	`DONE ABSOLUTE_MOVE`

2.5 Comando DELTA_MOVE

Formato:	`DELTA_MOVE dx dy dz`
Parámetros:	• `dx`: Desplazamiento relativo en el eje X (número decimal) • `dy`: Desplazamiento relativo en el eje Y (número decimal) • `dz`: Desplazamiento relativo en el eje Z (número decimal)
Descripción:	Mueve el sistema de manera relativa a la posición actual según los desplazamientos especificados.

Respuesta Exitosa:	`ACK DELTA_MOVE`
	`[ERROR: error_description] (Si hay errores)`
	`DONE DELTA_MOVE`

2.6 Comando GET_POSITION

Formato:	`GET_POSITION`
Parámetros:	Ninguno
Descripción:	Devuelve la posición actual del sistema.

Respuesta:	`ACK GET_POSITION`
	`[ERROR: error_description] (Si hay errores)`
	`DONE GET_POSITION: X Y Z`
	Donde X, Y, Z son las coordenadas actuales expresadas como números decimales separados por espacios.

2.7 Comando SET_SPEED

Formato:	`SET_SPEED speed`
Parámetros:	• `speed`: Velocidad de movimiento en mm/s (número decimal positivo).
Descripción:	Establece la velocidad de movimiento del sistema.

Respuesta Exitosa:	`ACK SET_SPEED`
	`[ERROR: error_description] (Si hay errores)`
	`DONE SET_SPEED`

2.8 Comando GET_SPEED

Formato:	`GET_SPEED`
Parámetros:	Ninguno
Descripción:	Devuelve la velocidad actual de movimiento del sistema.

Respuesta:	`ACK GET_SPEED`
	`[ERROR: error_description] (Si hay errores)`
	`DONE GET_SPEED: speed`
	Donde `speed` es la velocidad actual en mm/s.

2.9 Comando GET_MIN_SPEED

Formato:	`GET_MIN_SPEED`
Parámetros:	Ninguno
Descripción:	Devuelve la velocidad mínima permitida para el sistema.

Respuesta:	`ACK GET_MIN_SPEED`
	`[ERROR: error_description] (Si hay errores)`
	`DONE GET_MIN_SPEED: min_speed`
	Donde `min_speed` es la velocidad mínima en mm/s.

2.10 Comando GET_MAX_SPEED

Formato:	`GET_MAX_SPEED`
Parámetros:	Ninguno
Descripción:	Devuelve la velocidad máxima permitida para el sistema.

Respuesta:	`ACK GET_MAX_SPEED`
	`[ERROR: error_description] (Si hay errores)`
	`DONE GET_MAX_SPEED: max_speed`
	Donde `max_speed` es la velocidad máxima en mm/s.

2.11 Comando GET_ID

Formato:	`GET_ID`
Parámetros:	Ninguno
Descripción:	Devuelve el identificador único del dispositivo.

Respuesta:	`ACK GET_ID`
	`[ERROR: error_description] (Si hay errores)`
	`DONE GET_ID: DEVICE_ID`
	Donde `DEVICE_ID` es el identificador único del dispositivo (CX25F7TK9P).

2.12 Comando CHECK_ERRORS

Formato:	`CHECK_ERRORS`
Parámetros:	Ninguno
Descripción:	Realiza un diagnóstico del sistema y reporta cualquier error encontrado.

Respuesta:	`ACK CHECK_ERRORS`
	`[ERROR: error_description] (Si hay errores)`
	`DONE CHECK_ERRORS`

3 Referencia Rápida de Comandos

Formato	Descripción
HELP	Muestra la ayuda
SET_HOME	Establece la posición de origen
GO_HOME	Va a la posición de origen
ABSOLUTE_MOVE x y z	Movimiento a una posición absoluta
DELTA_MOVE dx dy dz	Movimiento relativo
GET_POSITION	Obtiene la posición actual
SET_SPEED speed	Establece la velocidad
GET_SPEED	Obtiene la velocidad actual
GET_MIN_SPEED	Obtiene la velocidad mínima
GET_MAX_SPEED	Obtiene la velocidad máxima
GET_ID	Obtiene el identificador único del dispositivo (CX25F7TK9P)
CHECK_ERRORS	Diagnostica errores

4 Librería CommandParser

Para facilitar la integración del protocolo descrito, se ha desarrollado una librería para Arduino llamada `CommandParser`. Esta librería está ubicada en la carpeta `CommandParser`.

4.1 Estructura de la Librería

La librería está organizada de la siguiente manera:

Archivo	Descripción
CommandParser.h	Archivo de cabecera que contiene las definiciones de la clase.
CommandParser.cpp	Implementación de la clase CommandParser.
keywords.txt	Palabras clave para resaltado de sintaxis en el IDE de Arduino.

4.2 Instalación de la Librería

Para instalar la librería se deben seguir estos pasos:

1. Copiar la carpeta `CommandParser` a la carpeta `libraries` dentro de la carpeta de instalación de Arduino.
2. Reiniciar el IDE de Arduino si está abierto.
3. La librería estará disponible en el menú `Sketch > Include Library`.

También se puede instalar la librería directamente desde el IDE de Arduino:

1. Compirmir la carpeta `CommandParser` como un `.zip`.
2. En el IDE de Arduino, ir a `Sketch > Include Library > Add .ZIP Library`.
3. Seleccionar el archivo `ZIP`.

4.3 Uso de la Librería

La librería CommandParser facilita la implementación del protocolo de comunicación serial descrito en este documento en un proyecto Arduino. Una vez implementada, el sistema podrá recibir y procesar todos los comandos especificados en el protocolo a través del puerto serial.

```
#include <CommandParser.h>

// Crear una instancia de CommandParser
CommandParser parser;

void setup() {
  // Inicializar la librería
  parser.begin();

  // Configurar los callbacks
  parser.config(
    setHomeCallback,
```

```
    goHomeCallback,
    absoluteMoveCallback,
    deltaMoveCallback,
    getPositionCallback,
    setSpeedCallback,
    getSpeedCallback,
    getMinSpeedCallback,
    getMaxSpeedCallback,
    checkErrorsCallback
  );
}

void loop() {
  // Leer y procesar comandos seriales
  parser.read();
}
```

4.4 Configuración de Funciones Callback

La librería utiliza un sistema de callbacks para procesar los comandos. El usuario debe implementar estas funciones con las firmas descritas a continuación:

Función Callback	**Firma**
setHomeCallback	`void setHomeCallback()`
goHomeCallback	`void goHomeCallback()`
absoluteMoveCallback	`void absoluteMoveCallback(float &x, float &y, float &z)`
deltaMoveCallback	`void deltaMoveCallback(float &dx, float &dy, float &dz)`
getPositionCallback	`void getPositionCallback(float &x, float &y, float &z)`
setSpeedCallback	`void setSpeedCallback(float &speed)`
getSpeedCallback	`void getSpeedCallback(float &speed)`
getMinSpeedCallback	`void getMinSpeedCallback(float &minSpeed)`
getMaxSpeedCallback	`void getMaxSpeedCallback(float &maxSpeed)`
checkErrorsCallback	`void checkErrorsCallback()`

4.5 Errores en las funciones Callback

Cualquier error dentro de una función callback se debe implementar siguiendo el formato definido en la sección Sección 1.4. Esto en Arduino se puede lograr mediante el siguiente código:

```
Serial.print("ERROR: ");
Serial.println(errorMessage);
```