

TRABAJO PRACTICO N° 4

Integrador (POO, Python, C++, red de datos)

A. Elementos necesarios

Guías de clase.

Trabajos Prácticos N° 1 a 3.

Documentos relacionados (ver anexo 2)

Dispositivos y plataformas de desarrollo.

B. Entrega y Evaluación

La fecha de entrega del informe y de la presentación grupal habitualmente se indica al principio del ciclo, pero puede ser ajustada por mutuo acuerdo en clase.

Esta presentación es realizada por el/los autor/es, con exposición pública y posterior coloquio grupal e individual de ampliación y defensa del trabajo, abordando aspectos teórico-prácticos tomando como base la implementación realizada.

C. Actividades Generales

1. A partir del enunciado de la consigna específica, de los temas especiales involucrados (gestión de archivos, gestión de memoria, subprocesos, comunicaciones mediante protocolos y/o mecanismos de red) y de los temas opcionales de su interés particular (interfaces gráficas de usuario, interacción con teclado y/o mouse, manipulación de gráficos, sonido, animaciones, Web, uso de móvil, etc.), investigue sobre las capacidades de los lenguajes Python y C++ para manejar los mismos.
2. Puede acordar con el profesor los detalles que no estén especificados con claridad en este enunciado.
3. Proponga e implemente una Solución Orientada a Objetos aplicando los conceptos fundamentales: Clases, Objetos, Instanciación, Herencia, Agregación, Polimorfismo, Streams de E/S, Constructores/Deconstructores, Sobrecarga de Métodos, Sobrecarga de Operadores, Control de errores mediante Clases de Excepción, Contenedores de Objetos, etc.
4. Una vez finalizada la implementación, confeccione un informe, siguiendo un esquema similar al indicado en el anexo 1 (puede tomar como base el utilizado en las entregas anteriores y ajustarlo convenientemente).
5. Elabore un video breve (10'-15'). En el mismo use de 5' a 7' para explicar sintéticamente su solución, preferentemente basada en gráficos y/o esquemas asociados tanto conceptuales como de la implementación realizada; y en el tiempo restante muestre el funcionamiento/uso de su aplicación.
6. Realice la entrega de los archivos resultantes en formato comprimido (código fuente del proyecto, incluyendo librerías de terceros usadas, ejecutables, informe y videos), en la actividad de entrega habilitada en el aula virtual (o envíe por correo al profesor si se indicara explícitamente).

D. Consigna específica**1. Dominio del problema**

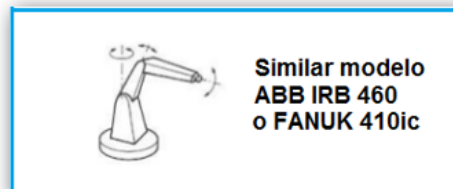
- Programación Orientada a Objetos para Manipulación Remota de un Robot.

2. Objetivo

- Desarrollar un aplicativo para controlar un robot de 3 grados de libertad con efector final.

3. Robot específico

- Tipo: robot 3DF.
- Configuración: RRR

**4. Requerimientos del Software**

- Aspectos generales
 - o El aplicativo/sistema a desarrollar, debe estar representado e implementado usando paradigma orientado a objetos en todas sus capas y puntos de despliegue.
 - o Debe diseñarse usando preferentemente 3 capas (MVC), de manera que la solución pueda usarse independientemente de que haya interfaz gráfica o no.
 - o Efector final: simple (puede considerarlo tipo pinza, sólo con acciones del tipo activar/desactivar).
 - o Dimensiones (extremas): anchura=A mm, profundidad=B mm, altura=C mm. Consultar la documentación anexa disponible en el aula virtual.
 - o Velocidades máximas: lineal=VL mm/s en Z y en el plano XY. Consultar la documentación anexa disponible en el aula virtual
 - o El controlador del robot recibe órdenes en formato G-Code y devuelve mensajes con información de estado. Se provee el firmware, adaptado para simulación, usando tecnología Arduino. Disponible en el aula virtual.
 - o Dicho controlador, conoce los parámetros geométricos y cinemáticos del robot. Realiza un control parcial de la viabilidad de cada orden recibida, retornando un breve mensaje o código si corresponde. Consultar la documentación anexa disponible en el aula virtual
 - o Los módulos del lado servidor deben estar desarrollados en Python.
 - o Los módulos del lado cliente deben estar desarrollados en C++.
 - o Los mensajes (órdenes y datos) enviados entre cliente y servidor deben serializarse como un flujo de objetos.
 - o Opcional: Una vez que el equipo termine dejando operativa una de las opciones, y desee implementar la otra opción, se sugiere hacerlo en un tercer lenguaje (Java, C#, javascript, etc.), y usando especialmente interfaces de usuario gráficas (escritorio, web o móvil).

- Comportamiento del lado servidor:
 - Ofrecer servicios vía RPC (usando formato XML o JSON a elección) que permita realizar:
 - Validar las peticiones mediante usuario y clave.
 - Conexión/desconexión al Robot, vinculado al servidor mediante una comunicación serie.
 - Activación/Desactivación de los motores del robot.
 - Listar los comandos disponibles de control.
 - Ofrecer mensaje de ayuda al operador, con ejemplificación de la sintaxis requerida para el comando indicado.
 - Ofrecer un reporte en consola de información general (conteniendo el estado de la conexión, la posición y el estado de actividad actual, el momento en que se inició la actividad, un listado con el detalle de las órdenes y la cantidad de ellas que han sido solicitadas desde la ultima conexión, detallando las que produjeron un error).
 - Ofrecer un reporte en consola con detalles del log de trabajo del servidor (peticiones, IP, usuario, fallos/exitos, marca temporal, etc.)
 - Selección de los modos de trabajo (manual o automático, absoluto o relativo)
 - [modo manual]: Movimiento lineal del efector especificando la velocidad y la posición final.
 - [modo manual]: Movimiento lineal del efector especificando sólo la posición final.
 - [modo manual]: Activación/Desactivación del efector final.
 - [modo manual]: Actividad del efector final (propias de la herramienta particular elegida) especificando velocidad de trabajo, tiempo de operación y sentido del movimiento.
 - [modo manual]: Realizar movimiento del efector a su "posición de origen/descanso".
 - [modo manual]: Aprendizaje de trayectoria con almacenamiento en archivos de texto (cuyo nombre es provisto por el operador), de las órdenes correspondientes en formato G-Code.
 - [modo automático]: Carga y ejecución de un archivo dado conteniendo una secuencia de trabajo específica, en formato G-code. El operador debe poder elegir el archivo a usar.
 - Mantener un archivo con el registro del trabajo (log acumulativo) realizado por el servidor. Este archivo responde a un formato CSV estándar.
 - Mantener un archivo con los usuarios autorizados y sus claves de acceso (como mínimo). Este archivo debe gestionarse mediante serialización y deserialización de objetos.
 - Ofrecer un panel de control del robot, mediante consola tipo CLI, capaz de:
 - [sólo el administrador] Mostrar/editar los parámetros de conexión del robot (los mismos se se resguardan en un archivo de configuración en formato json).

- [sólo el administrador] Encender o apagar el servidor RPC, de modo habilitar o no el acceso remoto
 - [sólo el administrador] Mostrar las últimas 100 líneas del archivo con el log de trabajo.
 - Opciones para ejecutar las mismas operaciones que las indicadas para las peticiones remotas vía RPC.
 - Manejar las E/S propias de la interacción con el operador.
- Comportamiento del lado Cliente:
 - o Ofrecer una interfaz de usuario que permita controlar al robot de manera remota, con despliegue visual y comportamiento similares a los de la interfaz de consola disponible en el servidor.
 - o Opción de menú para subir al servidor, archivos de tareas en formato G-Code y almacenarlos, de modo que puedan ser usados por el robot.
 - o [opcional] GUI de control y monitoreo de parámetros.
 - o [opcional] Despliegue visual del robot, usando vista 3D o proyecciones. En cualquiera de los casos puede usarse una estructura simplificada del robot, ejes/planos y escala (a su criterio).
 - o [opcional] Streaming de video mostrando el comportamiento del Robot usando una cámara remota.
 - o [opcional] Efectos de sonido que indiquen el arranque/parada del robot, los cambios de posición del efector final y los momentos de actividad del mismo.
 - o [opcional] Generación de alarmas visuales/auditivas cuando se intenten desplazamientos fuera del espacio de trabajo (aproximado).
 - Observaciones y restricciones:
 - o Defina al menos 2 packages, uno para el cliente y otro para el servidor.
 - o El efector final no agrega grados de libertad especiales, más allá de algún desplazamiento fijo en las coordenadas asumidas para el punto de operación.
 - o Para simplificar el desarrollo, considere que en modo manual, el robot recibe las órdenes de a una por vez.
 - o El tipeo o no de órdenes G-Code en la interfaz por parte del operador, queda a criterio del equipo de desarrollo.
 - o Considere las situaciones de concurrencia que pudieran darse y proponga los mecanismos/algoritmos para resolverlas.
 - o Las dimensiones y velocidades son usadas para validar/limitar las entradas de datos por parte del operador.
 - o Otros datos constructivos del robot o plataforma pueden ser indicados durante el desarrollo.
 - o Se sugiere partir de la implementación de un prototipo global ejecutable no funcional, e ir mejorándolo en sucesivos ciclos de desarrollo y revisión.

E. Iteración 2: Actividad para puesta en común y discusión de soluciones: 2/10

1. Construya un diagrama de clases global que muestre el modelo inicial previsto para los requerimientos enunciados del sistema de Robot RRR
 - Clases
 - Relaciones
 - Multiplicidad (estimada)
 - Atributos (para cada uno indicar tipo de dato estimado)
 - Métodos (para cada uno indicar argumentos y firma estimados)
2. Realice un esquema de la interfaz de usuario que tendrá el producto final, para el panel de comando del lado servidor. En caso que prevea el uso de cliente con GUI, agregue su propuesta.
3. Entregue en el aula virtual un informe mínimo (1 por grupo) que contenga:
 - una imagen en formato vectorial (o mapa de bits con resolución adecuada para evitar el pixelado al hacer zoom) del diagrama de clases obtenido.
 - archivo de imagen con la/s interfaz/ces de usuario propuesta/s.

ANEXOS

I. Esquema del Informe

1. Carátula (donde consten datos de identificación personales y de contexto)
 - a. Identificación de materia
 - b. Identificación del trabajo
 - c. Identificación de alumno/s
 - d. Identificación de docente/s
 - e. Identificación de carrera
 - f. Identificación de facultad/universidad
 - g. Identificación de ciclo lectivo
2. Índice
3. Introducción
 - a. Descripción breve del problema
4. Cuerpo
 - a. Conceptos/fundamentos/esquema o gráfica descriptiva, con detalles estructurales o funcionales para cada uno de los Temas Especiales agregados (agregue los esquemas o imágenes que estime convenientes para acompañar sus descripciones).
 - b. Descripción de componentes software (librerías, frameworks, aplicaciones, etc.) usados para implementar los aspectos anteriores (agregue los esquemas o imágenes que estime convenientes para acompañar sus descripciones).
 - c. Descripción/ejemplificación paso a paso del modo de instalación/vinculación/uso, aplicado a la solución específica
 - d. Incluya el modelo OO de lo implementado (tanto del lado servidor como del lado cliente), mediante Diagramas de Clases de diseño (debe ser consistente con la implementación)
 - e. Incluya un Diagrama de Secuencia para un escenario completo de trabajo, a partir de un requerimiento de usuario (debe ser consistente con la implementación). Elija uno de complejidad intermedia, no trivial.
 - f. Incluya un Diagrama de Actividad, que muestre una perspectiva global del aplicativo Servidor.
 - g. Describa los detalles de instalación, configuración y ejecución particulares de la implementación.
 - h. Capturas de pantalla mostrando la ejecución del aplicativo, para cada una de las interfaces de entrada/salida propuestas para interacción con el usuario.
 - i. Módulo y reporte de Prueba Unitaria para alguno de los siguientes módulos (a elección):
 - Interfaz de línea de comandos, lado servidor.
 - Clase Robot (o su equivalente) perteneciente a la capa de modelo.
 - Gestor de archivo/s
 - Servidor RPC
5. Comentarios y Conclusiones

- a. Comentarios sobre las dificultades enfrentadas y su resolución, justificaciones a la solución propuesta (de diseño, de implementación, de instalación, etc.)
 - b. Análisis de ventajas/desventajas de los componentes usadas
 - c. Posibles extensiones a la solución dada
6. Referencias
- a. Bibliografía/Sitios web consultados
 - b. URLs a los recursos software (tutorial, video, ejemplo, herramienta, etc.) que se hayan usado.

II. Robot y simulador

El desarrollo original pertenece a Florin Tobler, se encuentra en <https://www.thingiverse.com/thing:1718984>. Luego, el trabajo fue adaptado y continuado por diferentes autores.

Para la propuesta de la este Trabajo Práctico se hizo una extensión del diseño, disponible en <https://www.thingiverse.com/thing:3674358>

Y una bifurcación del código. Algunos de los cambios se encuentran como aporte a la comunidad RobotArm (<https://www.facebook.com/groups/robotarm>).

Para su aplicación en este Trabajo Práctico, se derivó en 2 versiones, una para usar como simulador de comportamiento del robot útil en la fase de desarrollo OO de las interfaces de usuario y otra con el firmware operativos para usar con el robot físico. Ambos códigos, provistos en el aula de la asignatura.

El software simulador es una modificación del código que funciona en el robot real.

Para que el producto resultante de este TP, pueda utilizar el simulador se requiere cargar este último a cualquier placa Arduino con capacidad de memoria suficiente y mantenerla conectada al puerto serie de la computadora de desarrollo.

Se puede realizar una interacción primaria, a través de un programa monitor del puerto serie (como el incorporado en Arduino IDE).

Si bien se han deshabilitado todas las salidas del microcontrolador, por precaución se recomienda que la placa no se encuentre conectada a ningún periférico, ni plaqueta auxiliar, y colocar la misma dentro de algún gabinete aislante para evitar se produzca algún cortocircuito durante el uso.

Las órdenes enviadas al robot/simulador hacen uso de lenguajes habituales en máquinas de control numérico, como son G-Code y M-Code.

- Tabla de mensajes de entrada/salida del simulador

En la siguiente tabla se pueden observar los comandos típicos que recibirá el robot durante su operación y las respuestas correspondientes que se recibirán por puerto serie.

A los efectos del TP, los comandos que implican movimientos del robot devuelven un tiempo

Entrada	Acción	Respuesta
M3	Activar gripper.	INFO: GRIPPER ON
M5	Desactivar gripper.	INFO: GRIPPER OFF
G28	Hacer homing.	INFO: HOMING INFO: HOMING COMPLETE t=7.50s
G1 Xa Yb Zc Fv	Mover el brazo a la posición (a,b,c)[mm] a una velocidad v [mm/s].	INFO: LINEAR MOVE: [X:150.00 Y:150.00 Z:0.00 E:0.00] t=7.50s En caso de que el punto solicitado este fuera del espacio de trabajo devuelve un error: ERROR: POINT IS OUTSIDE OF WORKSPACE
	Modo de coordenadas absolutas por defecto.	
	Devuelve las coordenadas absolutas luego de realizado el movimiento, independientemente de que se haya realizado un movimiento relativo.	
M114	Reporte de modo de coordenadas y posición actual del robot.	INFO: ABSOLUTE MODE INFO: CURRENT POSITION: [X:0.00 Y:170.00 Z:120.00 E:0.00]
G90	Modo de coordenadas absolutas.	INFO: ABSOLUTE MODE ON
G91	Modo de coordenadas relativas.	INFO: RELATIVE MODE ON
M17	Activar motores	INFO: MOTORS ENABLED
M18	Desactivar motores	INFO: MOTORS DISABLED

(supuesto) correspondiente al que demora el robot en realizar dicho movimiento.

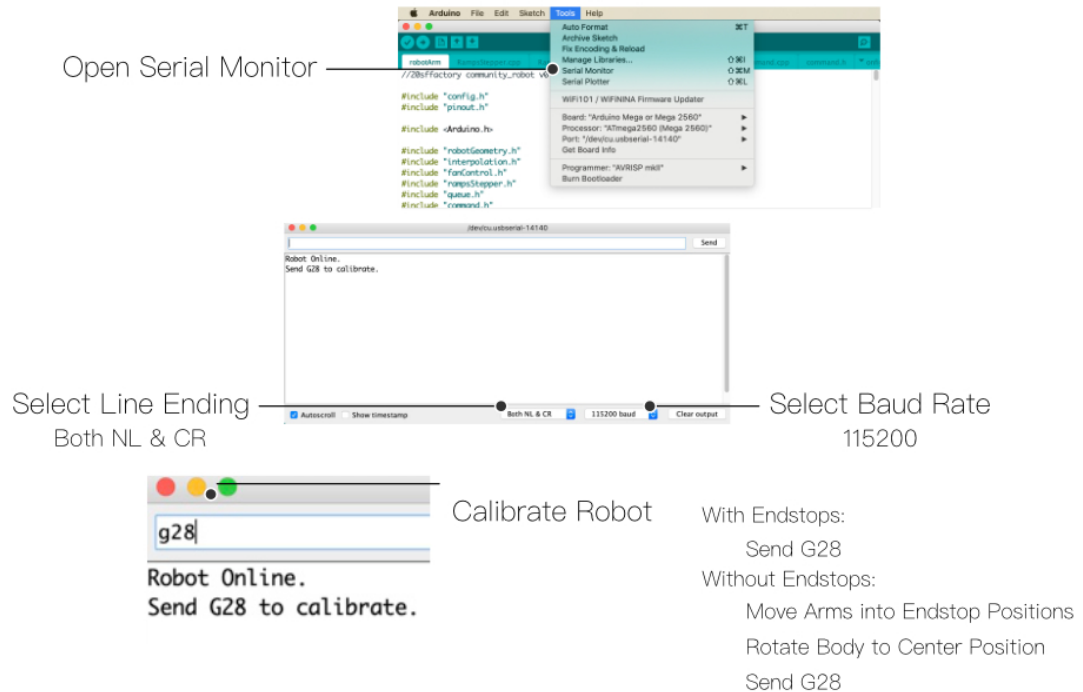
A continuación, otra documentación obtenida del proyecto original, que puede ser de interés.

- Comandos

Command Guide (Arduino IDE)

Arduino IDE Serial Monitor can perform basic single command serial communications.
Suitable for quick testing after firmware upload is completed.

Open Serial Monitor



Select Line Ending
Both NL & CR

Select Baud Rate
115200

Calibrate Robot

Robot Online.
Send G28 to calibrate.

With Endstops:
Send G28

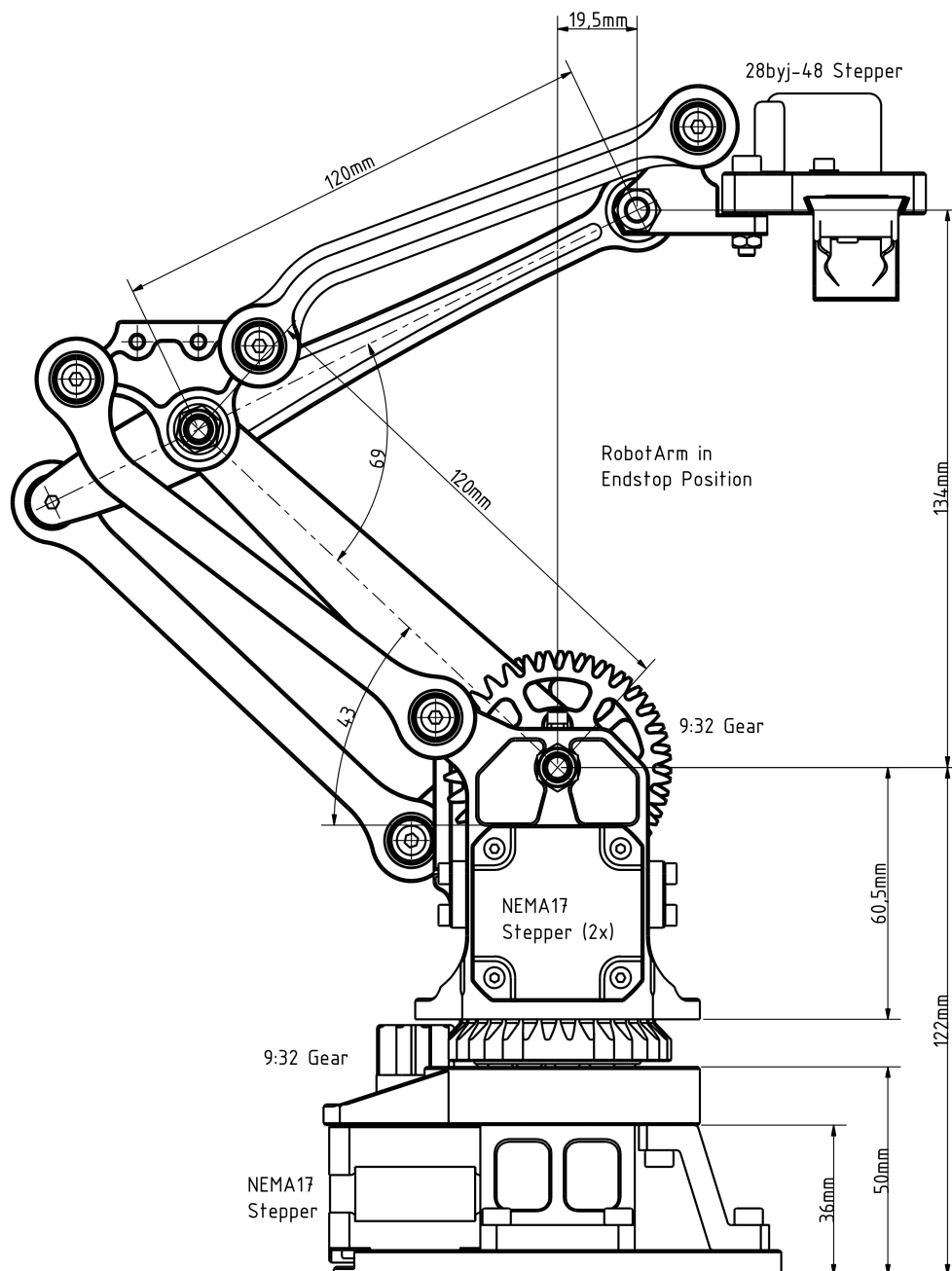
Without Endstops:
Move Arms into Endstop Positions
Rotate Body to Center Position
Send G28

Command List

See Cartesian Guide for Coordinate Reference

Linear Move	G0 X<x_mm> Y<y_mm> Z<z_mm> F<mm/s> eg: G0X120Z-10F30	Pump On/Off	M1 / M2
Dwell	G4 S<sec> (waits for <sec> seconds) eg: G4S10	Gripper On/Off	M3 / M5
Homing	G28 (home all motors if homing enabled)	Laser On/Off	M6 / M7
Absolute Mode	G90 (moves in absolute coordinate)	Steppers Enable/Disable	M17 / M18
Relative Mode	G91 (moves in increment value)	Fan Enable/Disable	M106 / M107
Set Position	G92 X<x_mm> Y<y_mm> Z<z_mm> E<mm/s> eg: G92 Z0 (sets current z value to 0) eg: G92 (resets all offset values)	Report Coordinates	M114
Set Speed Curve	M205 S[SPEED_PROFILE] eg: M205 S0 (sets flat speed curve) eg: M206 S2 (sets cosin speed curve)	Report Endstop State	M119

- Dimensiones del Robot:

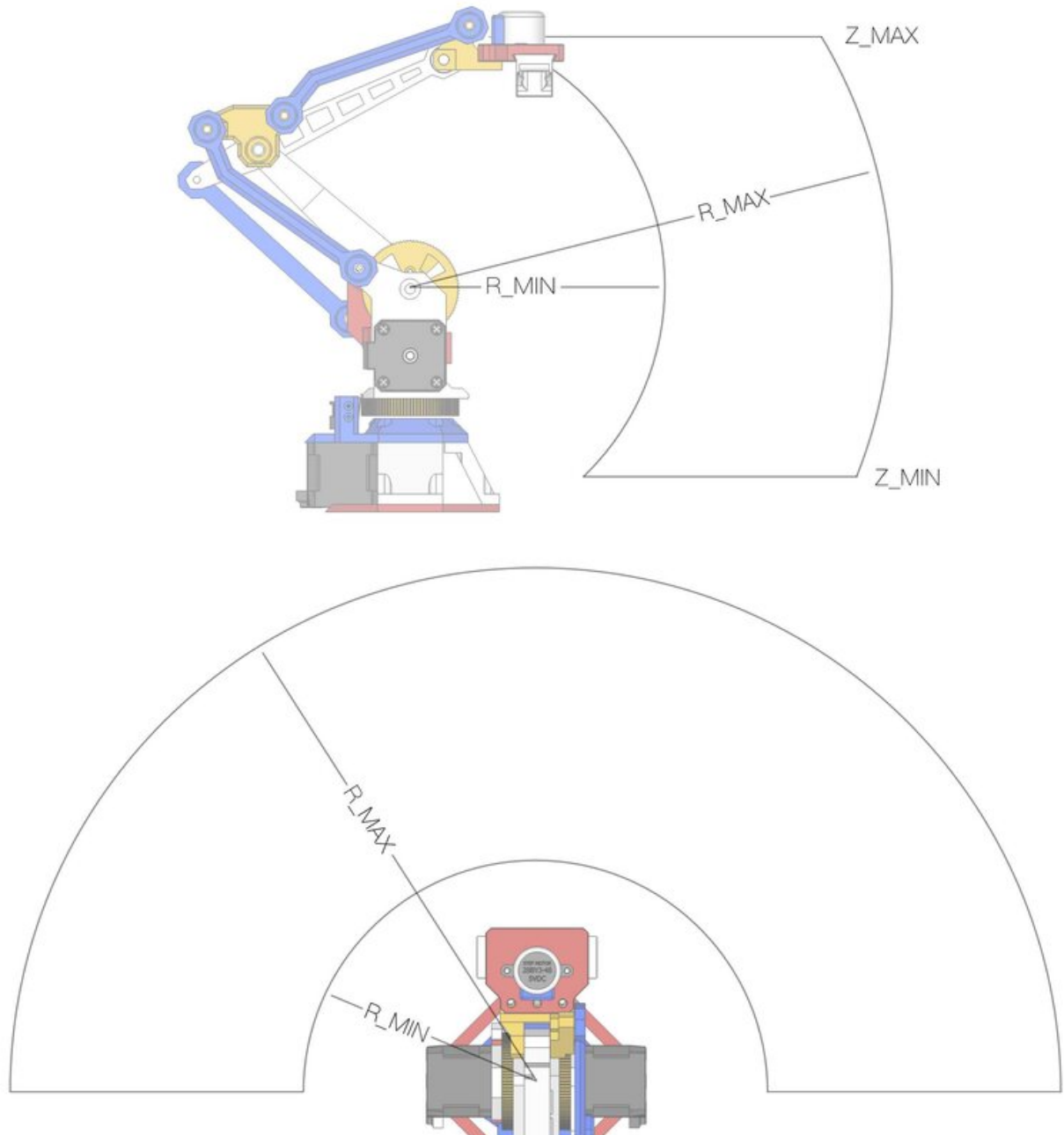


- Espacio de trabajo:

Operation Range Guide

Z_MAX, Z_MIN, R_MAX, R_MIN Values Located in "config.h"

Robot moves beyond range will be terminated

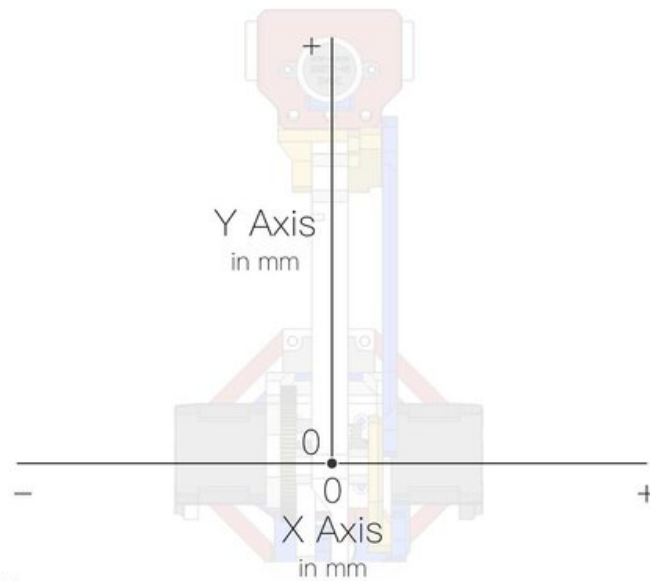


- Orientación cartesiana

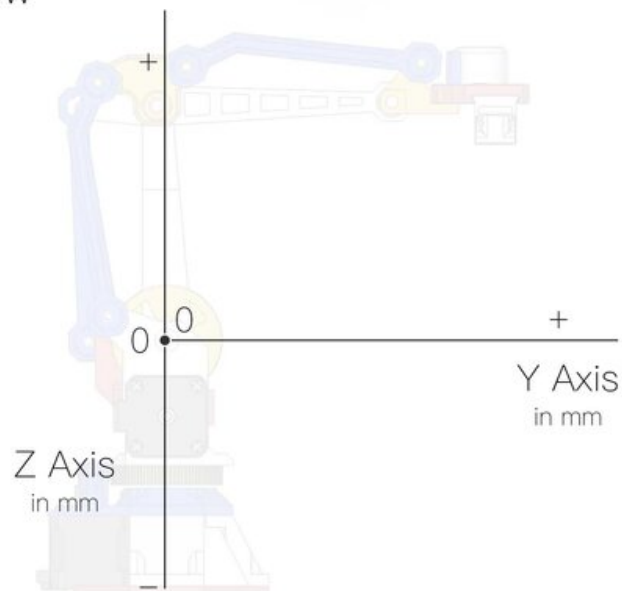
Cartesian Guide

each axis has ranges of coordinates which robot cannot reach, either too far or in conjunction with itself

Top View



Side View



III. Programación de máquinas NC con códigos G&M

La programación nativa de la mayoría de las máquinas de Control Numérico Computarizado (como fresadoras, cortadoras, tornos, impresoras 3D y robots) utiliza, además de algunos lenguajes propietarios y/o tradicionales, un lenguaje de bajo nivel habitualmente llamado G&M.

Los programas de G&M son simples archivos de texto ASCII (sólo mayúsculas, números y signos de puntuación tradicionales, por lo que es muy frecuente que los programas se almacenen y comuniquen usando un formato restringido de 6 bits).

El nombre G&M proviene de que el programa está constituido por instrucciones Generales y Misceláneas.

Si bien en el mundo existen aún diferentes dialectos de programación con códigos G&M, existe una estandarización que promovió la ISO. Esta estandarización fue adoptada por la totalidad de los fabricantes industriales de CNC (tales como Siemens Sinumeric, FANUC, Haas, Heidenhain o Mazak) y permite utilizar los mismos programas en distintas máquinas CNC de manera directa o con adaptaciones menores.

El G-code, también es conocido como RS-274 o lenguaje de programación G. Es usado principalmente en automatización y forma parte de la ingeniería o fabricación asistida por computadora. Los comandos G-Codes generalmente están acompañados de M-Codes.

En términos generales, G-code es un lenguaje mediante el cual se le indica a las CNC qué hacer y cómo hacerlo. Se trata de un lenguaje de programación vectorial mediante el que se describen acciones simples y entidades geométricas sencillas (básicamente segmentos de recta y arcos de circunsferencia) junto con sus parámetros de maquinado (velocidades de husillo y de avance de herramienta).

Existen diversas implementaciones del lenguaje. Además, los distintos fabricantes de máquinas CNC han añadido todo tipo de extensiones y variaciones al lenguaje, por lo que los operadores de las máquinas deben conocer las peculiaridades concretas que el fabricante ha previsto para su uso.

G-code comenzó siendo un tipo de lenguaje bastante limitado que carecía de estructuras como bucles, operadores condicionales y variables declaradas por el usuario. Sin embargo, las implementaciones más recientes de G-code sí que incluyen tales estructuras, creando un lenguaje algo más parecido a lo que podría ser un lenguaje de alto nivel.

A modo de ejemplo, se muestran los códigos de programación más utilizados en tornos de CNC de tecnoedu. Según el modelo, algunos de los códigos pueden estar deshabilitados.

Códigos Generales

- G00: Posicionamiento rápido (sin maquinar)
- G01: Interpolación lineal (maquinando)
- G02: Interpolación circular (horaria)
- G03: Interpolación circular (antihoraria)
- G04: Compás de espera
- G15: Programación en coordenadas polares
- G20: Comienzo de uso de unidades imperiales (pulgadas)
- G21: Comienzo de uso de unidades métricas
- G28: Volver al home de la máquina
- G40: Cancelar compensación de radio de curvatura de herramienta
- G41: Compensación de radio de herramienta a la izquierda
- G42: Compensación de radio de herramienta a la derecha
- G50: Cambio de escala
- G68: Rotación de coordenadas

- G73: Ciclos encajonados
- G74: Perforado con ciclo de giro antihorario para descargar virutas
- G76: Alesado fino
- G80: Cancelar ciclo encajonado
- G81: Taladrado
- G82: Taladrado con giro antihorario
- G83: Taladrado profundo con ciclos de retracción para retiro de viruta
- G90: Coordenadas absolutas
- G91: Coordenadas relativas
- G92: Desplazamiento del área de trabajo
- G94: Velocidad de corte expresada en avance por minuto
- G95: Velocidad de corte expresada en avance por revolución
- G98: Retorno al nivel inicial
- G99: Retorno al nivel R
- G107: Programación del 4º eje

Códigos Misceláneos

- M00: Parada
- M01: Parada opcional
- M02: Reset del programa
- M03: Hacer girar el husillo en sentido horario
- M04: Hacer girar el husillo en sentido antihorario
- M05: Frenar el husillo
- M06: Cambiar de herramienta
- M08: Abrir el paso del refrigerante
- M09: Cerrar el paso de los refrigerantes
- M10: Abrir mordazas
- M11: Cerrar mordazas
- M13: Hacer girar el husillo en sentido horario y abrir el paso de refrigerante
- M14: Hacer girar el husillo en sentido antihorario y abrir el paso de refrigerante
- M30: Finalizar programa y poner el puntero de ejecución en su inicio
- M38: Abrir la guarda
- M39: Cerrar la guarda
- M62: Activar salida auxiliar 1
- M67: Esperar hasta que la entrada 2 esté en ON
- M71: Activar el espejo en Y
- M80: Desactivar el espejo en X
- M81: Desactivar el espejo en Y
- M98: Llamada a subprograma
- M99: Retorno de subprograma

Referencias

- https://web.archive.org/web/20120226220922/http://www.tormach.com/machine_codes.html
- <http://s3.cnccookbook.com/Downloads/eBook/CNCCookbookGCodeCourse.pdf>
- <https://tecnoedu.com/CNC/GM.php>
- <https://en.wikipedia.org/wiki/G-code>