



UNIVERSIDAD POLITÉCNICA DE MADRID
ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA Y DISEÑO INDUSTRIAL

Grado en Ingeniería Electrónica Industrial y Automática

TRABAJO FIN DE GRADO

Diseño y programación de brazo SCARA para integrar como actuador primario en plataforma robótica móvil

Autor: Juan Manuel Astorga Diz

Cotutor:

Carlos Pastor

Kubernetes S. L.

Tutor:

Basil Mohammed Al-Hadithi

Departamento de Ingeniería Eléctrica,
Electrónica, Automática y Física Aplicada

Madrid, febrero 2022



POLITÉCNICA

escuela técnica superior de
ingeniería
y diseño
industrial

UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA Y DISEÑO INDUSTRIAL

Grado en Ingeniería Electrónica Industrial y Automática

TRABAJO FIN DE GRADO

Diseño y programación de brazo SCARA para integrar como actuador primario en plataforma robótica móvil

Autor: Juan Manuel Astorga Diz

Cotutor:

Carlos Pastor

Kubernetes S. L.

Tutor:

Basil Mohammed Al-Hadithi

Departamento de Ingeniería Eléctrica,
Electrónica, Automática y Física Aplicada

Madrid, febrero 2022

Firma Autor

Visto Bueno Tutor

Visto Bueno Co-Tutor

Título del trabajo: Diseño y Programación de brazo SCARA para integrar como actuador primario en plataforma robótica móvil.

Autor: Juan Manuel Astorga Diz

Tutor: Basil Mohammed Al-Hadithi

Co-tutor: Carlos Pastor

EL TRIBUNAL

Presidente:

Vocal:

Secretario:

Realizado el acto de defensa y lectura del Trabajo Fin de Grado el día de de en, en la Escuela Técnica Superior de Ingeniería y Diseño Industrial de la Universidad Politécnica de Madrid, acuerda otorgarle la CALIFICACIÓN de:

SECRETARIO

VOCAL

PRESIDENTE

AGRADECIMIENTOS

Este trabajo va dedicado a mis padres, ya que gracias a ellos he podido continuar mis estudios para seguir desarrollándome como profesional. Por otro lado, me es preciso agradecer a mi tutor Basil Mohammed por guiarme durante todo este proceso de desarrollo. Su manera de enfocar el método de trabajo me ha servido de gran ayuda a la hora de ir estableciendo un proyecto completo que fuera orientado a mis intereses como profesional. Además, él me ha puesto en contacto con Carlos Pastor a quien también quiero agradecer todo el tiempo y dedicación que ha aportado para ayudarme a conseguir las metas propuestas.

ÍNDICE

AGRADECIMIENTOS	I
ÍNDICE	I
RESUMEN	V
ABSTRACT	VI
LISTA DE ABREVIATURAS	VII
LISTA DE FIGURAS.....	IX
LISTA DE TABLAS	13
CAPÍTULO 1. INTRODUCCIÓN	14
1.1. MOTIVACIONES	15
1.2. OBJETIVOS.....	15
1.3. ESTRUCTURA DEL DOCUMENTO	16
CAPÍTULO 2. ESTADO DEL ARTE	17
2.1. ROBOT SCARA.....	18
2.2. SOFT GRIPPER.....	20
CAPÍTULO 3. FUNDAMENTOS TEÓRICOS	22
3.1. ROBOT SCARA.....	23
3.1.1. Clasificación	23
3.2. CINEMÁTICA DE SCARA.....	24
3.2.1. Cinemática directa	25
3.2.2. Cinemática inversa	28
3.3. SUJECIÓN DE OBJETOS ELIPSOIDALES	30
3.3.1. Objetos Elipsoidales	30
3.3.2. Soft Grippers	31
3.4. SENSORES DE FUERZA	32

3.4.1.	Funcionamiento	32
3.4.2.	Modos de funcionamiento	33
3.4.3.	Circuito de implementación	33
3.5.	MOTORES PASO A PASO	34
3.5.1.	Funcionamiento estático y dinámico	35
3.5.2.	Microstepping	35
3.5.3.	Tipos de SM.....	36
3.5.4.	Control de un SM.....	37
3.6.	PROTOCOLOS SERIALES DE COMUNICACIÓN	39
3.6.1.	SPI	40
3.6.2.	I2C	42
3.7.	MODELO CLIENTE-SERVIDOR Y COMUNICACIÓN RED.....	43
3.7.1.	Tipos de redes.....	43
3.7.2.	Modelo OSI	44
3.7.3.	Protocolo TCP/IP vs UDP/IP.....	44
3.7.4.	Sockets en UNIX.....	45
3.7.5.	paquete JSON	46
CAPÍTULO 4. DESARROLLO MECÁNICO DE ROBOT SCAR.....		47
4.1.	DISEÑO CAD Y REQUERIMIENTOS MECÁNICOS.....	48
4.1.1.	Prototipo de Gripper	48
4.1.2.	Soft Gripper definitivo.....	49
4.1.3.	Prototipos cuerpo SCARA	53
4.1.4.	Cuerpo SCARA definitivo.....	56
4.2.	PRODUCCIÓN DE PIEZAS	66
4.2.1.	Fabricación con tecnología FDM.....	67
4.2.2.	Mecanizado láser.....	70
4.3.	DESARROLLO ELÉCTRICO Y ELECTRÓNICO	71
4.3.1.	Unidad de control	72
4.3.2.	Periféricos	73
4.3.3.	Esquema de conexiones con PCB.....	83
CAPÍTULO 5. DESARROLLO: SW		85
5.1.	HERRAMIENTAS UTILIZADAS.....	87
5.1.1.	Arduino IDE.....	87
5.1.2.	Visual Studio Code.....	88
5.1.3.	GitHub desktop	88

5.2.	MÓDULO GUI	89
5.2.1.	Implementación de GUI con python	90
5.2.2.	Objetos de SW GUI	91
5.3.	MÓDULO SW DE CONTROL.....	99
5.3.1.	Implementación de esquema secuencial	100
5.3.2.	Configuración de módulo de control.....	101
5.3.3.	Servicios implementados.....	102
5.3.4.	Procesos principales en módulo de control	104
5.4.	SIMULACIÓN CON ROBOT MÓVIL.....	108
5.5.	PROTOCOLO DE COMUNICACIÓN ENTRE MÓDULOS SW	110
5.5.1.	Comandos JSON.....	110
5.6.	MOVIMIENTO DEL ROBOT	112
5.6.1.	Cálculo de trayectoria lineal.....	112
5.6.2.	Controlador de posición	114
CAPÍTULO 6. RESULTADOS DE HW Y SW DE SCARA		115
6.1.	HW FABRICADO Y MONTAJE.....	116
6.1.1.	Módulo BS	117
6.1.2.	Módulo ESB1	118
6.1.3.	Módulo ESB2	119
6.1.4.	Módulo ESB3	120
6.1.5.	Módulo PCBOX.....	121
6.1.6.	Módulo de SG	122
6.1.7.	Montaje completo de SCARA.....	122
6.2.	FUNCIONAMIENTO DE ELECTRÓNICA Y PERIFÉRICOS.....	125
6.3.	FUNCIONAMIENTO DE HMI	126
6.3.1.	Precondiciones.....	126
6.3.2.	Resultado de GUI con Robot	127
6.3.3.	Resultados con SG.....	127
6.4.	FUNCIONAMIENTO DEL CUERPO DEL ROBOT	128
6.4.1.	Resultados pruebas unitarias	128
6.4.2.	Pruebas con cinemática inversa	130
6.5.	RESULTADOS DE SUJECCIÓN CON SG	131
6.6.	RESUMEN DE CARACTERÍSTICAS DE ROBOT SCARA	133
6.7.	RESULTADOS DE SIMULACIÓN CON BASE MÓVIL	133
6.8.	DISCUSIÓN DE RESULTADOS	135
CAPÍTULO 7. CONCLUSIONES Y FUTUROS TRABAJOS.....		136

7.1.	CONCLUSIONES	137
7.2.	FUTUROS TRABAJOS Y MEJORAS.....	139
BIBLIOGRAFÍA.....		141
ANEXOS 1. TABLA DE COSTES.....		145
ANEXOS 2. ESQUEMÁTICOS PCB		147
ANEXOS 3. IMÁGENES ADICIONALES		149
ANEXOS 4. DATA SHEET: SCARMOV		153
1.	FIABILIDAD-FLEXIBILIDAD-DESEMPEÑO	155
2.	DIMENSIONES CUERPO DEL ROBOT.....	156
3.	DIMENSIONES ACOPLADOR GRIPPER	157
4.	EJES DE MOVIMIENTO	158
5.	RANGO DE TRABAJO	158
6.	ESPECIFICACIONES TÉCNICAS.....	159

RESUMEN

El siguiente trabajo incluye la elaboración completa de un robot SCARA con posibilidad de desempeñar numerosas aplicaciones de sujeción. En este sentido el objetivo principal es conseguir un diseño que permita el cambio de actuadores de forma sencilla, tanto a nivel de Hardware como Software y que tenga la posibilidad de aplicarse en múltiples aplicaciones industriales.

Para este propósito se hace una investigación de los trabajos actuales realizados en el área para tener un punto de partida desde el cual comenzar el desarrollo. Luego se evalúan los conceptos y matemática implicada en el proceso de control para la implementación del SW y HW de la aplicación. Además, se incluye la investigación de los SG como uno de los posibles tipos de actuadores que puede ser integrado con el robot principal de forma que se puedan ejecutar tareas de sujeción de objetos elipsoidales con control de presión.

Durante el desarrollo se verá el proceso de producción de todo el cuerpo del robot mediante el uso de tecnologías FDM y corte láser. Después se hará un estudio de la arquitectura SW y su implementación con el uso de lenguajes de programación de alto nivel, como Python, y de bajo nivel, como C y C++. Para ello se desarrollaron 2 módulos de aplicación que fueron conectados mediante uso de comunicación IP/UDP. Finalmente se hace el montaje de una simulación en la que se prueba el SCARA siendo utilizado como actuador principal de un robot móvil.

Este trabajo pretende establecer los pilares de un FWR que cuente con todos los requerimientos necesarios para que un usuario pueda, desde una HMI sencilla, controlar y monitorizar un robot SCARA a través de WiFi. De esta forma se cuenta con un punto de partida para proyectos de mayor envergadura en el área de control, automatización e IoT.

ABSTRACT

The following work includes the complete development of a SCARA robot with the possibility of numerous clamping applications. In this sense, the main objective is to achieve a design that allows the change of actuators in a simple way, both at the Hardware and Software level and that has the possibility of being applied in multiple industrial applications.

For this purpose, an investigation of the current works carried out in the area is made to have a starting point from which the development will begin. Then the concepts and mathematics involved in the control process for the implementation of the SW and HW of the application will be evaluated. In addition, the investigation of the SG is included as one of the possible types of actuators that can be integrated with the main robot so that clamping tasks of ellipsoidal objects with pressure control can be executed. Also, see the production process of the entire robot body by using FDM and laser cutting technologies. Afterwards, a study of the SW architecture and its implementation will be made with the use of high-level programming languages, such as Python, and low-level ones, such as C and C++. For this, 2 application modules were developed that were connected using IP/UDP communication. Finally, a simulation is assembled in which the SCARA is tested being used as the main actuator of a mobile robot.

This work aims to establish the pillars of a FWR that has all the necessary requirements so that a user can, from a simple HMI, control and monitor a SCARA robot through WiFi. In this way, there is a starting point for larger projects about control, automation and IoT.

LISTA DE ABREVIATURAS

API	Application Programming Interface
ARM	Advanced RISC Machine
CAD	Computer-Aided Design
CAM	Computer-Aided Manufacturing
CLK	Clock
CU	Control Unit
DOF	Degree Of Freedom
ETSIDI	Escuela Técnica Superior de Ingeniería y Diseño Industrial
FDM	Fused Deposition Modeling
FSR	Force Sensing Resistor
FW	Firmware
FWK	Framework
GUI	Graphic User Interface
HMI	Human-Machine Interface Protocol
HT	Holding Torque
HW	Hardware
I2C	Inter-Integrated Circuit
IC	Integrated Circuit
IDE	Integrated Development Environment
IFR	International Federation of Robotics
IRQ	Interrupt Request
JSON	JavaScript Object Notation
LAN	Local Area network
LS	Limit Switch
MCU	Microcontroller Unit (μC)
MISO	Master Input Slave Output
MMI	Machine-Machine Interface

MOSI	Master Output Slave Input
NEMA	National Electrical Manufacturers Association
OSI	Open Systems Interconnection
PCB	Printed Circuit Board
PLA	Polylactic Acid
OOP	Object Oriented Programming
RISC	Reduced Instruction Set Computing
SG	Soft Gripper
SM	Stepper Motor
SP	Set Point
SPI	Serial Peripheral Interface
SS	Slave Select
SW	Software
TCP	Transmission Control Protocol
IP	Internet Protocol
TFG	Trabajo Fin de Grado
UART	Universal Asynchronous Receiver-Transmitter
UDP	User Datagram Protocol
USB	Universal Serial Bus
VSC	Visual Studio Code
WAN	Wide Area Network
WWW	World Wide Web

LISTA DE FIGURAS

<i>Figura 2.1 PT80 fast Picker [1]</i>	18
<i>Figura 2.2 SCARA Fanuc [2]</i>	19
<i>Figura 2.3 SCARA de HTM [3]</i>	19
<i>Figura 2.4 BRI gripper from Thingiverse [5]</i>	20
<i>Figura 2.5 Gripper con autocuración [6]</i>	20
<i>Figura 2.6 FSR en SG [11]</i>	21
<i>Figura 2.7 RayFin finger [12]</i>	21
<i>Figura 3.1 Esquema de SCARA [14]</i>	24
<i>Figura 3.2 Relación entre Cinemática inversa y directa [14]</i>	24
<i>Figura 3.3 Parámetros de un elipsoide</i>	30
<i>Figura 3.4 Clasificación SG [18]</i>	31
<i>Figura 3.5 Resistencia de FSR vs Fuerza</i>	32
<i>Figura 3.6 Mecanismos de FSR [23]</i>	33
<i>Figura 3.7 Divisor de voltaje FSR [23]</i>	34
<i>Figura 3.8 Dinámica en trayectoria un Motor por pasos [26]</i>	35
<i>Figura 3.9 SM de reluctancia con 6 polos y rotor con 6 dientes</i>	36
<i>Figura 3.10 MS unipolar [26]</i>	37
<i>Figura 3.11 MS bipolar [26]</i>	37
<i>Figura 3.12 Planta de control con realimentación y PID [28]</i>	37
<i>Figura 3.13 Modelo electromecánico de motor DC [31]</i>	39
<i>Figura 3.14 Topologías de comunicación SPI [32]</i>	40
<i>Figura 3.15 Ejemplo comunicación SPI [32]</i>	41
<i>Figura 3.16 Modos SPI [32]</i>	41
<i>Figura 3.17 Ejemplos de trama de datos I2C [32]</i>	42
<i>Figura 3.18 Esquema de conexión maestro esclavo I2C [32]</i>	42
<i>Figura 3.19 Señales I2C</i>	42
<i>Figura 3.20 LAN-WAN</i>	43
<i>Figura 3.21 Modelo OSI</i>	44
<i>Figura 3.22 Cliente Servidor UNIX</i>	45
<i>Figura 3.23 Ejemplo paquete JSON</i>	46
<i>Figura 4.1 Primera versión SG</i>	48
<i>Figura 4.2 Comparación V1 y V2 SG</i>	49
<i>Figura 4.3 versión 3 de SG</i>	49
<i>Figura 4.4 SG definitivo</i>	49
<i>Figura 4.5 Primera versión de base definitiva para SG</i>	50

<i>Figura 4.6 Versión 2 de SG</i>	51
<i>Figura 4.7 Base de SG con sujeción por tornillos</i>	51
<i>Figura 4.8 Cuarta versión de SG</i>	51
<i>Figura 4.9 Primera versión de Pieza flexible</i>	52
<i>Figura 4.10 Segunda versión de pieza flexible con base rígida</i>	52
<i>Figura 4.11 Pieza flexible versión 3</i>	52
<i>Figura 4.12 Versión definitiva pieza flexible de SG</i>	53
<i>Figura 4.13 Segunda versión de prototipo SCARA</i>	53
<i>Figura 4.14 Segunda versión de cuerpo SCARA</i>	54
<i>Figura 4.15 ESB2 Versión 2</i>	54
<i>Figura 4.16 Cuarta versión cuerpo SCARA</i>	55
<i>Figura 4.18 Quinta versión cuerpo SCARA</i>	55
<i>Figura 4.17 Corte lateral de cuerpo SCARA versión 5</i>	55
<i>Figura 4.19 Sistema de poleas en base de quinta versión de cuerpo SCARA</i>	56
<i>Figura 4.20 Diseño final montado SCARA</i>	56
<i>Figura 4.21 Base de cuerpo final SCARA</i>	57
<i>Figura 4.22 BS_BOT_TENS y BS_TOP_TENS</i>	58
<i>Figura 4.23 BS_TOP_MOT2</i>	58
<i>Figura 4.24 BS_TOP_COV</i>	58
<i>Figura 4.25 BS_BOT_MOT1</i>	59
<i>Figura 4.26 BS_BOT_COV</i>	59
<i>Figura 4.27 BS_PULL_P27</i>	59
<i>Figura 4.28 BS_PULL_P62 y BS_PULL_P20</i>	60
<i>Figura 4.29 Corte transversal en sistema de poleas de base SCARA</i>	60
<i>Figura 4.30 Eslabón 1 definitivo para cuerpo SCARA</i>	60
<i>Figura 4.31 ESB1_SJBS_PUL62</i>	61
<i>Figura 4.32 ESB1_SJBS_TOP</i>	61
<i>Figura 4.34 ESB1_SJ2_TOP</i>	61
<i>Figura 4.33 ESB1_SJ2_BOT</i>	61
<i>Figura 4.35 ESB1_PRC_TOP</i>	62
<i>Figura 4.36 ESB1_PRC_BOT</i>	62
<i>Figura 4.37 Modulo ESB2 definitivo</i>	62
<i>Figura 4.38 ESB2_SJ3</i>	63
<i>Figura 4.39 ESB2_SJ1_PUL21</i>	63
<i>Figura 4.40 ESB2_SJ1_TOP</i>	64
<i>Figura 4.41 ESB2_SJ1_BOT</i>	64
<i>Figura 4.42 Módulo ESB3</i>	64
<i>Figura 4.43 ESB3_MGRP</i>	65
<i>Figura 4.44 ESB3_MOT3</i>	65
<i>Figura 4.45 PCBOX</i>	66
<i>Figura 4.46 PCBOX_TOP</i>	66
<i>Figura 4.47 PCB_BOT</i>	66
<i>Figura 4.48 Cortadora láser Gesmain Co2 5080</i>	71
<i>Figura 4.49 Pines de Arduino MKR1010</i>	73

<i>Figura 4.50 TMC5160 SilentStepStick [52]</i>	74
<i>Figura 4.51 Modelo un maestro y múltiples esclavos con TMC5160 [51]</i>	75
<i>Figura 4.52 Rampa de velocidades TMC5160 [51]</i>	76
<i>Figura 4.53 Gráfica de corriente en motor en función de registro IHOL_IRUN</i>	77
<i>Figura 4.54 FSR07 de Ohmite [53]</i>	78
<i>Figura 4.55 Toma de datos para caracterización FSR</i>	78
<i>Figura 4.56 Gráfica de resultados de caracterización FSR07 (R - F)</i>	78
<i>Figura 4.57 Gráfica primer tramo de respuesta FSR07 (Peso - Resistencia)</i>	79
<i>Figura 4.58 Gráfica segundo tramo de respuesta FSR07 (Peso - Resistencia)</i>	79
<i>Figura 4.59 Gráfica tercer tramo de respuesta FSR07 (Peso - Resistencia)</i>	79
<i>Figura 4.60 LS de LERDGE</i>	80
<i>Figura 4.61 Pantalla OLED AZDelivery</i>	81
<i>Figura 4.62 NEMA 17</i>	82
<i>Figura 4.63 Nema 23</i>	83
<i>Figura 4.64 Esquema de conexión SPI de PCB</i>	84
<i>Figura 4.65 Configuración de sensores en PCB</i>	84
<i>Figura 5.1 Arquitectura SW general</i>	86
<i>Figura 5.2 Arduino IDE 2.0 Beta</i>	87
<i>Figura 5.3 Visual Studio Code</i>	88
<i>Figura 5.4 GitHub Desktop</i>	89
<i>Figura 5.5 Arquitectura general SW GUI</i>	90
<i>Figura 5.6 Hilo de recepción de Objeto de conexión</i>	92
<i>Figura 5.7 Hilo principal de Objeto de conexión</i>	92
<i>Figura 5.8 Caso de uso de lectura de valor desde el objeto mánager</i>	93
<i>Figura 5.9 Caso de uso en escritura de valor de atributo del objeto Robot</i>	94
<i>Figura 5.10 Proceso principal de mánager</i>	95
<i>Figura 5.11 Estructura de Interfaz gráfica</i>	98
<i>Figura 5.12 Inicio de objeto GUI</i>	98
<i>Figura 5.13 Diagrama de procesos activos con GUI loop</i>	99
<i>Figura 5.14 Bucle principal de módulo de control</i>	101
<i>Figura 5.15 Caso de uso de gestor de texto para sincronización entre gestor de estados y gestor de conexión</i>	103
<i>Figura 5.16 Esquema de capas de módulo SW con interfaz MCU</i>	104
<i>Figura 5.17 Bucle de ejecución para gestor de comandos y control en módulo SW de control</i>	105
<i>Figura 5.18 Proceso de gestión UDP en módulo SW de control</i>	106
<i>Figura 5.19 Proceso de control de estados de módulo de control</i>	107
<i>Figura 5.20 Ejemplo de mensaje JSON de estado</i>	108
<i>Figura 5.21 Proceso de lectura de sensores en módulo SW de control</i>	108
<i>Figura 5.22 Interfaz de CoppeliaSim y modelo de SCARA</i>	109
<i>Figura 5.23 Opciones de Simulación en interfaz de aplicación SW</i>	110
<i>Figura 5.24 Comando JSON de control</i>	112
<i>Figura 5.25 Desplazamiento de p1 a p2 en línea recta</i>	113
<i>Figura 5.26 Casos especiales de trayectoria lineal</i>	113
<i>Figura 6.1 Pieza impresa BS_BOT_MOT1</i>	117

<i>Figura 6.2 Pieza impresas BS_PULL_P27</i>	117
<i>Figura 6.3 Sistema de poleas impreso de BS</i>	118
<i>Figura 6.4 Pieza impresa BS_TOP_MOT2</i>	118
<i>Figura 6.5 Piezas impresa de cobertura de BS</i>	118
<i>Figura 6.6 Módulo construido ESB1</i>	119
<i>Figura 6.7 Pieza impresa ESB2_SJ1_TOP</i>	119
<i>Figura 6.8 Pieza impresa ESB2_SJ1_PUL21</i>	119
<i>Figura 6.9 Piezas impresa ESB2_SJ1_BOT y ESB2_SJ3</i>	120
<i>Figura 6.10 Módulo montado ESB3</i>	120
<i>Figura 6.11 Pieza impresa ESB3_MGRP</i>	121
<i>Figura 6.12 Pieza impresa PCBOX_TOP</i>	121
<i>Figura 6.13 Pieza impresa PCBOX_BOT</i>	121
<i>Figura 6.14 Montaje módulo PCBOX</i>	121
<i>Figura 6.15 Pieza impresa SG_FLX</i>	122
<i>Figura 6.16 Módulo montado SG</i>	122
<i>Figura 6.17 Pieza de soporte de perfiles ESC_SUP</i>	123
<i>Figura 6.18 Base se soporte de SCARA</i>	123
<i>Figura 6.19 Pieza de cobertura de extremos de perfil COVER_PR</i>	123
<i>Figura 6.20 Montaje de los módulos del robot SCARA</i>	124
<i>Figura 6.21 Estructura completa de robot SCARA</i>	124
<i>Figura 6.22 Resultado de ensamblaje de SG</i>	124
<i>Figura 6.23 Salida OLED</i>	125
<i>Figura 6.24 Configuración de conexión</i>	126
<i>Figura 6.25 Parámetros de configuración de robot</i>	126
<i>Figura 6.26 Comprobación de configuración de tipo de aplicación de control</i>	126
<i>Figura 6.27 Prueba de botón de conexión de GUI</i>	126
<i>Figura 6.28 Monitorización de señales con GUI</i>	127
<i>Figura 6.29 Resultados de control de SG</i>	128
<i>Figura 6.30 Conexión a simulador CoppeliaSim desde GUI</i>	133
<i>Figura 6.31 Cinemática inversa en CoppeliaSim</i>	134
<i>Figura 6.32 Simulación con base móvil en CoppeliaSim</i>	134

LISTA DE TABLAS

<i>Tabla 3.1: D-H para SCARA.....</i>	28
<i>Tabla 3.2 Actores en comunicación red.....</i>	43
<i>Tabla 4.1 Comparación entre impresoras 3D utilizadas.....</i>	68
<i>Tabla 4.2 parámetros impresión piezas con PLA.....</i>	69
<i>Tabla 4.3 Parámetros de impresión con TPU A85.....</i>	70
<i>Tabla 4.4 Características CO2 3050.....</i>	71
<i>Tabla 4.5 Señales de TMC5160 SilentStepStick [52].....</i>	74
<i>Tabla 4.6 Registros más relevantes de TMC5160 [51].....</i>	75
<i>Tabla 4.7 Respuesta de corriente según registro IHOLD_IRUN.....</i>	77
<i>Tabla 5.1 Requerimientos para SW de GUI.....</i>	91
<i>Tabla 5.2 Comandos de Objeto mánager con estructura de datos</i>	95
<i>Tabla 5.3 Comandos de ejecución de funciones en objeto Mánager.....</i>	96
<i>Tabla 5.4 Dependencias de módulo de control.....</i>	100
<i>Tabla 6.1 Ficheros para fabricación e información adicional.....</i>	116
<i>Tabla 6.2 Resultado de medición de fuerza con FSR.....</i>	125
<i>Tabla 6.3 Resultados de errores medios en desplazamiento de eslabones.....</i>	129
<i>Tabla 6.4 Errores medios de movimientos XY.....</i>	130
<i>Tabla 6.5 Objetos de prueba de sujeción</i>	131
<i>Tabla 6.6 Características de SCARA.....</i>	133

Capítulo 1. INTRODUCCIÓN

1.1. MOTIVACIONES

Los robots son actualmente uno de los desarrollos más importantes de la humanidad. Han servido tanto para agilizar la producción como para cumplir objetivos que antes eran inimaginables. A lo largo del tiempo han surgido numerosos avances en el mundo de la robótica que han servido para hacer a estas máquinas una opción más accesible y rentable. Esto ha permitido una significativa reducción en el coste en su construcción y a su vez una mayor fiabilidad en el momento de trabajar junto con personas.

En general un robot está pensado para que cumpla al menos con una tarea: transporte, *pick and place*, atornillado, etc. Sin embargo, un nuevo enfoque en este campo ha sido buscar una mayor versatilidad funcional. Es por ello que en este proyecto se busca crear un robot que cumpla con un gran rango de alcance y numerosas funcionalidades de sujeción mediante un sistema que acepte distintos tipos de gripper. Por otro lado, este robot debe ser capaz de realizar tareas de pick and place buscando la trayectoria más eficiente para su movimiento y que cuente con un gran rango de alcance. Todo ello mediante un control remoto vía WiFi que será gestionado desde una interfaz intuitiva para cualquier usuario.

Con todo esto se busca crear una herramienta guía que esté al alcance de cualquier persona que quiera entrar en este mundo de la robótica y desee construir su propio robot con ayuda de las nuevas tecnologías de impresión 3D. Por esta razón se verá que, a lo largo del proyecto, existe un gran enfoque en las versiones de diseño que se fueron creando y cómo han surgido las soluciones que mejor cumplen los requerimientos funcionales del SCARA.

Finalmente, este proyecto obliga a investigar todo lo necesario para crear una aplicación de usuario completa que pueda servir de base para futuras implementaciones en la programación con Python y a su vez aplicar los conocimientos enfocados en POO, sincronización de procesos y programación de MCU.

1.2. OBJETIVOS

- Servir de guía completa para replicaciones futuras. Se incluirán todos los archivos necesarios para la implementación de este proyecto en su totalidad, tanto desarrollo HW como SW.
- Desarrollar un robot capaz de cumplir con las tareas de pick and place buscando una trayectoria recta que permita el desplazamiento rápido del eslabón final
- Diseñar y producir un SG que cuente con capacidad de sujeción. El objetivo será recoger objetos convexos con solamente la necesidad de un motor y que pueda sujetar objetos de hasta 300g y de un diámetro máximo de 8cm
- Adaptar el sistema para contar con la posibilidad de cambio de gripper con distintas funcionalidades
- Monitorizar la presión del gripper para evitar daños en los objetos sujetados
- Controlar remotamente por wifi de baja latencia al robot
- Crear una interfaz completa para mandar comandos al robot con Python que cuente con capacidad de hacer cambios y ampliaciones del sistema.

- Establecer los pilares básicos para futuros desarrollos enfocados en la visión artificial y nuevas tecnologías de control.
- Simular el control del robot SCARA desarrollado sobre una base móvil

1.3. ESTRUCTURA DEL DOCUMENTO

En la siguiente lista se puede ver la distribución por capítulos junto con una breve descripción de lo que se expone en el mismo:

- **Capítulo1: Introducción.** Exposición breve del proyecto y sus objetivos principales
- **Capítulo2: Estado del arte.** Se exponen proyectos de referencia similares y actuales para posicionar el proyecto en el contexto de la robótica. Este capítulo fue separado en 2 para diferenciar las referencias de los SACARA con los desarrollos de SG.
- **Capítulo3: Fundamentos Teóricos.** Definición más extendida de los conceptos principales y elaboración de las matemáticas necesarias para el control y manejo de periféricos del robot.
- **Capítulo4: Desarrollo mecánico de robot SCARA.** Se hace un estudio de la evolución del diseño del robot para luego explicar cada una de las partes que conforman el cuerpo definitivo y las tecnologías requeridas para su producción. También se explican los requerimientos eléctricos y electrónicos que tiene el SCARA y se muestran diagramas de conexión.
- **Capítulo 5: desarrollo SW.** Se estudia la arquitectura de todo el SW con diagramas de flujo que explican su funcionamiento y conexión entre sus dos módulos: SW de GUI y SW de control.
- **Capítulo 6: Resultados.** Demostración de montaje total del robot e implementación del control e integración con la interfaz gráfica, así como los resultados obtenidos de las pruebas realizadas.
- **Capítulo 7: Conclusiones y futuros trabajos.** Análisis de resultados en comparación con los objetivos planteados. Evaluación de mejoras y futuros objetivos.

Capítulo 2. ESTADO DEL ARTE

2.1. ROBOT SCARA

En el mercado existen numerosas versiones de robots SCARA. Esto es principalmente debido a que, como se verá más adelante, es un robot que cuenta con una gran rigidez y un rendimiento bastante elevado a la hora de desempeñar tareas de pick and place. Se verán algunos de los modelos más modernos que sirvieron de referencia para el diseño y funcionalidad básica del robot desarrollado.



Figura 2.1 PT80 fast Picker [1]

El TP-80 Fast Picker (Figura 2.1) es un SCARA de Staubli¹ que fue la principal inspiración en el diseño del cuerpo del robot. Además, cumple con los principales objetivos de funcionamiento y rendimiento [1]. Entre ellos se encuentran:

- Capacidad de carga de 1Kg
- Amplio rango de movimiento (Figura 2.1)
- Rapidez de movimiento
- Alta precisión
- Flexibilidad de montaje. Opción para base móvil.
- Estructura rígida

Otro interesante diseño es el presentado por FANUC², el SR-6iA (Figura 2.2). En cuanto a este segundo robot se debe destacar el modo de funcionamiento del último eslabón. Este sirvió de referencia para desarrollar el tercer eslabón del robot. Además, es evidente que el SR-6iA es un robot de mayores prestaciones [2]. Entre las características de mayor interés están:

- Notable rapidez de acción
- Amplio rango de acción (Figura 2.2)
- Capacidad de carga de 6Kg

¹ www.staubli.com

² www.fanuc.eu

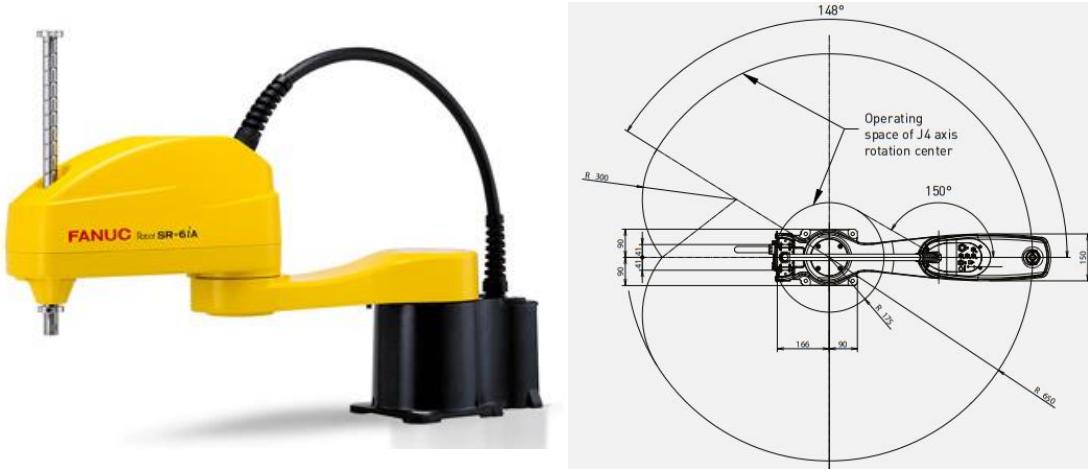


Figura 2.2 SCARA Fanuc [2]

El siguiente caso menciona un proyecto desarrollado por *How Mechatronics*¹ en el cual se construye y utiliza un SCARA controlado con un Arduino [3] UNO que a su vez controla 4 motores de pasos con una placa CNC Shield para controladores de potencia. Además, todos los eslabones y base principal fueron producidas con impresión 3D (Figura 2.3).

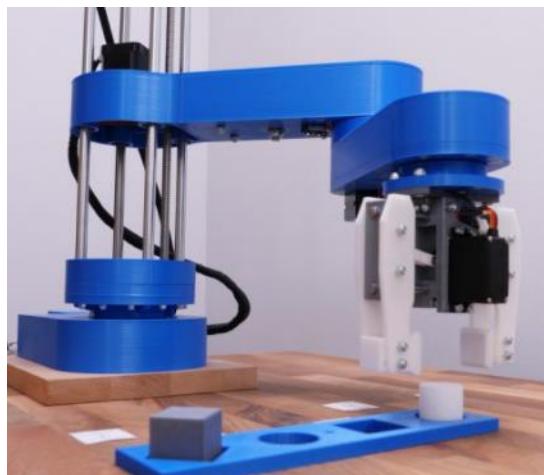


Figura 2.3 SCARA de HTM [3]

Lo más interesante de este proyecto es que cuenta con enfoque sencillo de control para un robot SCARA. Gracias a este desarrollo se pudo comprobar la cinemática inversa que se estaba utilizando e implementarla como base inicial en el SW del proyecto [3]. Por otro lado, también inspiró la interfaz gráfica para mandar las señales de control hacia el robot. Por ende, este fue un buen punto de partida para empezar a establecer los objetivos principales que se quería alcanzar:

- Interfaz gráfica
- Estructura del cuerpo impresa en 3D

¹ www.howtomechatronics.com

2.2. SOFT GRIPPER

Una parte del desarrollo será dedicada a producir un SG que permita sujetar objetos elipsoidales. Desde el 2011 se ha empezado con los primeros SG [4] y gracias a la revolución de la impresión 3D y nuevas tecnologías para materiales elastómeros la producción de este tipo de sujetadores ha aumentado en gran medida[4].

El primer diseño (Figura 2.4) evaluado fue uno que se encontró en Thingiverse¹, una web dedicada a compartir archivos de diseño digital creados por el usuario, a nombre de *Black Ram Industries* [5].



Figura 2.4 BRI gripper from Thingiverse [5]

Este diseño es el que inspiró el modelo básico con el que contaría el SG del robot. Se puede ver (Figura 2.4) que sigue los parámetros expuestos en [4] en cuanto a la sujeción de objetos elipsoidales. Sin embargo, el archivo STL no sirvió para su uso ya que las dimensiones tuvieron que ser adaptadas a los requerimientos del diseño del cuerpo del SCARA (4.1). Sin embargo, el funcionamiento mecánico corresponde con el que se busca.

Se verán más adelante (4.1.2) los cambios que se realizaron sobre esta idea principal para que no fueran necesarias articulaciones mecánicas y para adaptar los sensores de presión en las pinzas flexibles.



Figura 2.5 Gripper con autocuración [6]

Lo interesante de usar materiales flexibles es que hay algunos que cuentan con características que pueden servir para determinadas aplicaciones. Tal es el caso del SG

¹ www.thingiverse.com

desarrollado la VUI [6] que cuenta con la capacidad de autocuración (Figura 2.5). Esta se logra con un material llamado DPBM-FT5000 que tiene la capacidad de ser estructuralmente estable en temperatura ambiente y que, al aumentar la temperatura, permite la adhesión de fisuras [6]. Dicho material tiene, además, una superficie con alto coeficiente de adhesión, por lo que la sujeción es bastante sencilla y no requiere de altas presiones sobre los objetos a sujetar.

Una de las principales aportaciones de este artículo [6] fue el mecanismo de acción que efectuaba el movimiento de las pinzas. Sin embargo, se comprobó posteriormente (4.1.1) que las partes mecánicas en las articulaciones podría ser un problema de mantenimiento y posibles fallos durante el funcionamiento. Sin embargo, este proyecto [6] demuestra la gran capacidad que tienen los SG para contar con determinadas características que pueden ser útiles en ciertas aplicaciones. Además, los dedos se lograron imprimir con impresoras 3D, mediante la tecnología de impresión FDM [7], [8] que será usado en el actual proyecto siguiendo el mismo diseño general triangular (4.1.2).

Se ha visto en las imágenes anteriores que existe un patrón más o menos parecido en el diseño de estos SG (Figura 2.5 y Figura 2.4). Esta manera de actuación es gracias a lo que se conoce como efecto Fin Ray [9], [10]. Entre uno de los casos estudiados se encuentra el de [11] donde se puede ver un proyecto que junta la base del funcionamiento del SG en una única pieza estructural. Esto reduce el peso del mecanismo y hace que la producción reduzca costes y número de procesos involucrados.

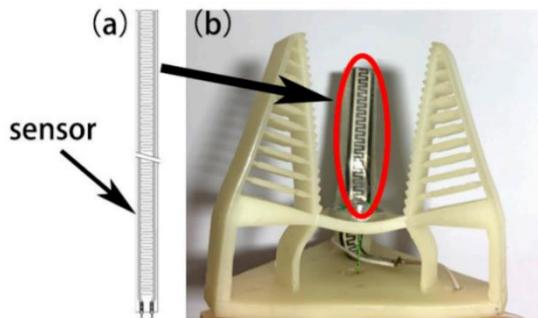


Figura 2.6 FSR en SG [11]

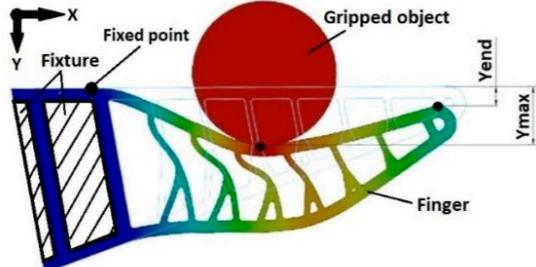


Figura 2.7 RayFin finger [12]

Además, se incluye un sensor [11] de presión rectangular (Figura 2.6) con el objetivo de controlar la fuerza ejercida por el SG sobre el objeto a sujetar. Esto permite tener control de la presión en toda la zona de contacto que existe entre el SG y el objeto.

Finalmente, en el estudio de [12] se hace una investigación más completa de la eficiencia de la sujeción del SG (Figura 2.7) y también de su producción con impresión FDM. Además, se busca la mejor estructura para utilizar la menor fuerza posible del actuador y suministrar una fuerza de sujeción máxima siguiendo un enfoque matemático [12] pero sin establecer una estructura ideal sino una manera de medir la efectividad de cualquier estructura que se use para este efecto. Por esto es que este estudio [12] aportó una noción clara de cómo podía afectar la estructura interna del SG y también escoger los parámetros de fabricación que mejor resultado presentan.

Capítulo 3. FUNDAMENTOS TEÓRICOS

En este capítulo estudiaremos los conceptos principales que han sido necesarios para el desarrollo del SCARA a nivel SW y HW. Para ello se divide en varias secciones que explican en, rasgos generales, un elemento o concepto que será implementado de forma específica para el funcionamiento del robot como un solo sistema.

3.1. ROBOT SCARA

Es un robot que cuenta con un diseño enfocado en tareas de ensamblado y pick and place. Esto se debe que ofrece una gran rigidez en el eje vertical y un gran desempeño en el plano de trabajo. Normalmente cuenta con 3 o 4 articulaciones principales:

- Articulación prismática: esta puede encontrarse en la base del robot o en el último eslabón que sujeta al gripper. Permite los movimientos en el eje vertical o z.
- Articulaciones de revolución: se encargan de realizar los movimientos en el plano de movimiento del robot. Puede incluir una articulación que se encargue de orientar el gripper.

Las principales ventajas que tienen estos robots son su fácil instalación ya que requiere de un espacio de trabajo bien acotado. Esto a su vez se traduce en un menor costo y mantenimiento. Sin embargo, esta limitación tiene como inconveniente la relativa baja capacidad para interactuar con el entorno de trabajo debido a su bajo número de grados de libertad [13].

3.1.1. CLASIFICACIÓN

Los robots se suelen clasificar de diversas formas siempre atendiendo a distintos criterios y características que pueden depender de las del mismo robot o de las tareas que va a desempeñar [14].

Siguiendo los criterios basados en ISO 8373 los SCARA se pueden clasificar [14] según las categorías siguientes:

- La generación: el objetivo del proyecto supone una adaptación de un robot de primera generación ya que no recoge información directa de su entorno. Sin embargo, esta base serviría para desarrollos futuros en los que se quiere tener un sistema de segunda generación como mínimo.
- Área de Aplicación: según la IFR el robot que se desarrollará cumplirá las aplicaciones de manipulación de materiales con el código 230 [14].
- Tipo de Actuador: este criterio hace referencia al tipo de energía que se usa para mover al robot. En el caso de estudio se cuenta con un robot Eléctrico.
- Tipo de control: se plantea un robot controlado por trayectoria ya que se irá ejecutando instrucciones en el tiempo de movimiento de forma de obtener la siguiente posición objetivo. De esta forma es posible especificar toda la trayectoria de manera continua. A su vez se busca un robot con cierto grado de control adaptativo ya que se contará con sensores de presión que permita una realimentación de la fuerza que se está ejerciendo sobre el objeto sujetado. De este modo el robot puede cambiar su tarea de acuerdo con la información recibida. Finalmente, para esta clasificación de tipo de

control se cuenta como un robot teleoperado ya que los comandos de movimiento serán enviados de forma remota vía WiFi.

3.2. CINEMÁTICA DE SCARA

Para controlar y monitorizar la posición del robot se hizo el estudio de la cinemática inversa y directa del robot (Figura 3.2). En este caso el problema se reduce únicamente a una solución presente en un espacio de 2 dimensiones ya que el tercer eslabón determina el desplazamiento en z sin necesidad de ningún cálculo de cinemática. Solución que igualmente se verá más adelante con distintos métodos.

Para el robot SCARA desarrollado existen las siguientes articulaciones:

- Articulación 1(q_1): de revolución que es la que se conoce como el hombro y cuenta con un rango de movimiento de 230°
- Articulación 2(q_2): de revolución que se conoce como codo ya que conecta la base con el extremo del robot y permite hacer de pivote para efectuar el movimiento. Cuenta con un rango de 260°
- Articulación 3 (q_3): de tipo lineal que ofrece el desplazamiento vertical del gripper en el espacio. Cuenta con un rango de 28,5 cm.

Cada articulación aporta 1 grado de libertad al robot, por ende, el SCARA que se desarrolla tiene 3 DOF en total (Figura 3.1)

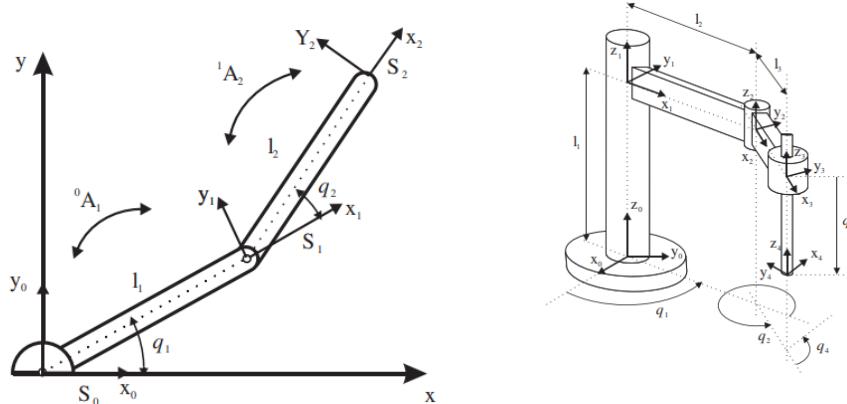


Figura 3.1 Esquema de SCARA [14]

A la izquierda se ve una vista superior de SCARA. Esta corresponde con la vista de perspectiva de la derecha.

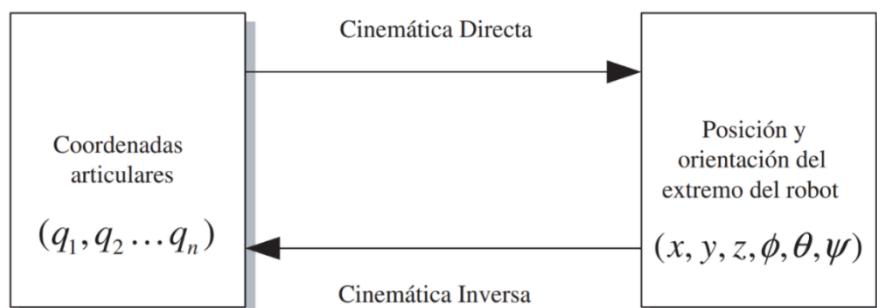


Figura 3.2 Relación entre Cinemática inversa y directa [14]

Antes que nada, es necesario recordar que lo que se quiere hacer con el estudio de la cinemática del robot se debe ir de un espacio de articulaciones a un espacio cartesiano y viceversa (Figura 3.1). Por ello hay que tomar en cuenta la relación bidireccional entre las coordenadas articulares y la posición y orientación del extremo del robot. Esto es posible con el uso de la cinemática inversa y directa que se explicaran a continuación.

3.2.1. CINEMÁTICA DIRECTA

El problema cinemático directo consiste en determinar cuál es la posición y orientación del extremo final del robot, con respecto a un sistema de coordenadas que se toma como referencia, conocidos los valores de las articulaciones y los parámetros geométricos de los elementos del robot [14]. Existen varios métodos para la obtención de esta posición. En este apartado se verán dos posibles soluciones que comprobarán la efectividad y concordancia de ambos métodos de cálculo.

3.2.1.1. MÉTODO DE CAMBIOS DE BASE

Como lo que se quiere es tener la solución general del problema se optó por un método basado en cambios de sistema de referencia entre articulaciones ya que los métodos geométricos cuentan, como se verá más adelante, con una gran limitación a medida que se aumenta el número de DOF.

Este método [14] se basa en la definición de la matriz de dimensiones 4x4 de transformación homogénea A que representa la posición y orientación relativa entre los sistemas asociados a dos eslabones consecutivos (Ecuación 3.1).

$${}^{i-1}A_i = \begin{bmatrix} n & s & a & p \\ 0 & 0 & 0 & 1 \end{bmatrix} = Rot(x, \emptyset)Rot(y, \theta)Rot(z, \Psi)T(p)$$

Ecuación 3.1

En la Ecuación 3.1 se tiene que n , s y a son vectores ortogonales y unitarios que representan la rotación relativa entre los sistemas $i - 1$ e i . Y que p representa la posición del sistema i respecto del sistema $i - 1$. Es decir, que esta matriz homogénea puede dividirse en una matriz de rotación por cada eje de coordenadas basadas en los ángulos de Euler [14] y una de traslación.

Por ello es por lo que, como en el caso de estudio existen 3 DOF, al representar la localización del tercer eslabón a partir de la base habría que ir haciendo el mismo proceso para los 3 casos existentes y así daría como resultado lo mostrado en Ecuación 3.2.

$${}^0A_3 = {}^0A_1 {}^1A_2 {}^2A_3$$

Ecuación 3.2

Finalmente, se debe tomar en cuenta que en cada caso de cambio de base se tendrá una sola matriz de rotación. Esto se debe a que en cada relación de bases existe una rotación con respecto a un solo eje. Por ejemplo, entre q_1 y q_2 solamente existe una rotación respecto del eje z. Sin embargo, en el caso de q_3 y q_4 no existe ninguna rotación ya que la articulación solamente efectúa movimientos lineales. Todo ello se refleja, a partir de Ecuación 3.1 y tomando en cuenta la resolución de los ángulos de Euler para cada caso, en la Ecuación 3.3, la Ecuación 3.4 y la Ecuación 3.5.

$${}^1A_2 = \begin{bmatrix} c_1 & -s_1 & 0 & 0 \\ s_1 & c_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & l_1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} c_1 & -s_1 & 0 & l_1 c_1 \\ s_1 & c_1 & 0 & l_1 s_1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Ecuación 3.3

$${}^2A_3 = \begin{bmatrix} c_2 & -s_2 & 0 & 0 \\ s_2 & c_2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & l_2 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} c_2 & -s_2 & 0 & l_2 c_2 \\ s_2 & c_2 & 0 & l_2 s_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Ecuación 3.4

$${}^3A_4 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & q_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Ecuación 3.5

Se puede ver que en la Ecuación 3.5 resulta que, tal como se esperaba, no existe rotación en el último eslabón y, por ende, el desplazamiento en z dependerá de este valor de forma directa.

Finalmente, si se juntan estos resultados en Ecuación 3.2 resulta Ecuación 3.6 ¹.

$${}^0A_3 = \begin{bmatrix} c_{12} & -s_{12} & 0 & l_1 c_1 + l_2 c_{12} \\ s_{12} & c_{12} & 0 & l_1 s_1 + l_2 s_{12} \\ 0 & 0 & 1 & d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Ecuación 3.6

Queda únicamente igualar con la matriz homogénea tal como se muestra en Ecuación 3.7.

$$\begin{bmatrix} n & o & a & p \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} c_{12} & -s_{12} & 0 & l_1 c_1 + l_2 c_{12} \\ s_{12} & c_{12} & 0 & l_1 s_1 + l_2 s_{12} \\ 0 & 0 & 1 & d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Ecuación 3.7

Solamente queda igualar término a término las matrices. Como $p^T = [x \ y \ z \ 0]$:

¹ Los diminutivos hacen referencia a suma de ángulos. Ejemplo: $c_{12} = \cos(q_1 + q_2)$

$$x = l_1 \cos(q_1) + l_2 \cos(q_1 + q_2)$$

Ecuación 3.8

$$y = l_1 \sin(q_1) + l_2 \sin(q_1 + q_2)$$

Ecuación 3.9

$$z = d_3$$

Ecuación 3.10

Es evidente, a partir de los resultados, que la Ecuación 3.10 es la que podría deducirse de forma directa tomando en cuenta que el tercer eslabón se desplaza hacia arriba partiendo desde su máxima elongación tomada como $z = 0$. Es decir, si se toma en cuenta que el primer motor que efectúa la rotación q_1 se encuentra en una base de altura inferior (Figura 3.1) se tendría un ligero cambio en las ecuaciones planteadas. Esto se refleja, siguiendo el mismo proceso anterior de cambio de bases, en la Ecuación 3.11, la Ecuación 3.12 y la Ecuación 3.13.

$$x = l_2 \cos(q_1) + l_3 \cos(q_1 + q_2)$$

Ecuación 3.11

$$y = l_2 \sin(q_1) + l_3 \sin(q_1 + q_2)$$

Ecuación 3.12

$$z = l_1 - q_3$$

Ecuación 3.13

3.2.1.2. MÉTODO DENAVIT-HARTENBERG

Este método utiliza una matriz de transformación homogénea que describe la relación existente entre dos elementos rígidos adyacentes en el espacio. De esta forma el problema cinemático directo es reducido a encontrar una matriz de transformación homogénea 4×4 que relacione la localización espacial del extremo del robot con respecto al sistema de coordenadas de su base [14].

Partiendo de lo anteriormente mencionado en cuanto a que cada matriz de conexión entre articulación depende de una matriz de rotación y una de translación, se puede definir fácilmente este método como la reducción del problema cinemático [14] a 4 variables principales:

- α : rotación en torno del eje x para hacer coincidir los ejes z entre articulaciones consecutivas.
- θ : rotación en torno del eje z para hacer coincidir los ejes x entre articulaciones consecutivas.
- d : desplazamiento en el eje z para hacer coincidir el origen de los sistemas de coordenadas entre articulaciones.
- a : desplazamiento en el eje x para hacer coincidir el origen de los sistemas de coordenadas entre articulaciones.

Dicho esto, se ve claramente que el problema queda resumido a la siguiente expresión:

$${}^{i-1}A_i = \text{Rotz}(\theta_i)T(0,0,d_i)T(0,0,a_i)\text{Rotx}(\alpha_i)$$

Ecuación 3.14

Si en Ecuación 3.14 se agregan todas las variables resulta Ecuación 3.15.

$${}^{i-1}A_i = \begin{bmatrix} c\theta_i & -c\alpha_i s\theta_i & s\alpha_i s\theta_i & a_i c\theta_i \\ s\theta_i & c\alpha_i c\theta_i & -s\alpha_i c\theta_i & a_i s\theta_i \\ 0 & s\alpha_i & c\alpha_i & d_i \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Ecuación 3.15

Con la ecuación Ecuación 3.15 se puede seguir el mismo método planteado con resolución por cambio de base y se llegaría a la misma solución siguiendo el algoritmo de resolución D-H [14]. Con estos pasos aclarados se puede construir la tabla de D-H (Tabla 3.1) a partir de un robot SCARA de 3 DOF (Figura 3.1).

Articulación	θ_i	d_i	a_i	α_i
1	q_1	l_1	l_2	0
2	q_2	0	l_3	0
3	0	q_3	0	0

Tabla 3.1: D-H para SCARA

Nótese que si se toma el q_3 con desplazamiento positivo hacia abajo resultaría un $\alpha_3 = 180^\circ$ necesarios para alinear los ejes z.

Finalmente, metiendo los datos en Ecuación 3.15 resulta Ecuación 3.16.

$${}^1A_2 = \begin{bmatrix} cq_1 & -sq_1 & 0 & l_2 cq_1 \\ sq_1 & cq_1 & 0 & l_2 sq_1 \\ 0 & 0 & 1 & l_1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Ecuación 3.16

Si recordamos lo anteriormente mencionado se puede ver que la Ecuación 3.16 es la Ecuación 3.3. Es decir, que si hacemos este mismo cálculo para los 2 siguientes eslabones se llega a la Ecuación 3.6 con la diferencia de que en este caso existe un desplazamiento en z propio del cambio de referencia y que corresponde con los resultados de la Ecuación 3.11, Ecuación 3.12 y Ecuación 3.13. Es decir, que este método no es más que una simplificación del método de cambios de base (3.2.1.1).

3.2.2. CINEMÁTICA INVERSA

Es la que se utiliza para conocer cada una de las coordenadas articulares que debe tener el robot para que el extremo de este se posicione en una coordenada específica. Básicamente, como su nombre lo indica, se quiere hacer el análisis contrario de la cinemática directa visto

anteriormente (Figura 3.2). Por ello, es que es la que más va a ser necesaria a la hora de controlar los movimientos del SCARA.

En este caso se hará uso del método geométrico con el cual se busca una relación trigonométrica entre los eslabones y ángulos articulares del robot (Figura 3.1). El principal problema que se va a tener a la hora de hacer este cálculo es que la resolución depende fuertemente de las características del robot (por ejemplo, los límites físicos de las articulaciones). Además, dichas ecuaciones, a medida que aumentan los DOF, suelen ser trascendentales siendo así necesario cálculos iterativos de gran tamaño y que no garantizan una solución correcta. Es por ello por lo que este método está limitado para un robot de pocos DOF. Por ello es por lo que cuando se trabaja con robots de más de 3 DOF se utiliza el método de desacoplo mecánico que permite reducir el número de DOF y simplificar los cálculos [14]. Sin embargo, existen casos de robots que debido a su complejidad requieren de cálculos matemáticos e iterativos (por ejemplo, con la matriz Jacobiana [15]).

En el caso del SCARA se puede hacer, teniendo una vista superior del caso de estudio (Figura 3.1), un análisis geométrico tal como se describe en [15]. Las relaciones trigonométricas requieren del teorema del seno para su resolución. A partir de esta relación resulta la Ecuación 3.17.

$$x_p^2 + y_p^2 = l_1^2 + l_2^2 + 2l_1l_2 \cos(q_2)$$

Ecuación 3.17

Donde x_p y y_p son los puntos a los que se quieren llegar en el plano XY. Si ahora se despeja de la Ecuación 3.17 el ángulo de interés q_2 resulta Ecuación 3.18.

$$q_2 = \pm \cos^{-1} \left(\frac{x_p^2 + y_p^2 - l_1^2 - l_2^2}{2l_1l_2} \right)$$

Ecuación 3.18

En este caso el ángulo q_2 tiene dos posibles soluciones ya que el coseno es una función par. Esto a su vez se traduce en que el ángulo q_1 tenga, a su vez, dos posibles respuestas y por ende el robot cuente con dos lógicas de posicionamiento: una positiva y otra negativa.

Si se usa la Ecuación 3.11 y Ecuación 3.12 se obtienen los siguientes resultados:

$$\begin{cases} \sin(q_1) = \left(\frac{l_2 \sin(q_2)z_p + (l_1 + l_2 \cos(q_2))y_p}{(l_2 \sin(q_2))^2 + (l_1 + l_2 \cos(q_2))^2} \right) \\ \cos(q_1) = \left(\frac{(l_1 + l_2 \cos(q_2))z_p - l_2 \sin(q_2)y_p}{(l_2 \sin(q_2))^2 + (l_1 + l_2 \cos(q_2))^2} \right) \end{cases} \rightarrow q_1 = \tan^{-1} \left(\frac{l_2 \sin(q_2)z_p + (l_1 + l_2 \cos(q_2))y_p}{(l_1 + l_2 \cos(q_2))z_p - l_2 \sin(q_2)y_p} \right)$$

Ecuación 3.19

Es evidente que, a partir de Ecuación 3.19 y como ya se había mencionado, que este método es muy poco escalable para mayores DOF.

Finalmente, con la Ecuación 3.13 se puede tomar la última variable que queda para definir todo el volumen de trabajo del SCARA. Resulta entonces:

$$q_3 = z - l_1$$

Ecuación 3.20

En este caso hay que resaltar que esta ecuación parte de que l_1 es la distancia que tiene la base del robot del primer escalón (Figura 3.1). Con esto en cuenta se puede decir que con la Ecuación 3.17, la Ecuación 3.18 y la Ecuación 3.19 se puede tener una descripción geométrica completa para tener el todo momento la posición del robot.

3.3. SUJECIÓN DE OBJETOS ELIPSOIDALES

Un robot por lo general cuenta con un extremo que tiene una herramienta que cumple un fin específico de sujeción o cualquier otra acción. Existen múltiples herramientas que logran tomar objetos con distintas tecnologías. Dichos eslabones se conocen normalmente como grippers.

Recientes estudios [4], [16], [17] han demostrado la increíble facilidad con la que cierto tipo de sujetadores pueden servir para numerosos tipos de formas con la acción de un único motor. Esto es importante a la hora de reducir el peso, complejidad y coste del robot.

En este apartado se estudiará un tipo de sujetadores que está revolucionando el campo de la sujeción en la robótica, los SG. Además, se evaluará su uso con objetos elipsoidales y porqué son los más indicados para este fin.

3.3.1. OBJETOS ELIPSOIDALES

Un elipsoide se describe como un objeto de 3 dimensiones formado por 3 parámetros característicos. Dichos parámetros serán los radios de las esferas que formen el elipsoide a, b, c y que tienen las relaciones de Ecuación 3.21. En este sentido si $a=b=c$ entonces se generará una esfera (Figura 3.3).

$$\alpha = \frac{b}{a}, \beta = \frac{c}{a}$$

Ecuación 3.21

A partir de las relaciones de Ecuación 3.21 es posible definir 2 tipos de objetos elipsoidales:

- Elipsoides proláticos: son aquello en los cuales $a=c$ y $b>a$
- Elipsoides obláticos: son aquellos en los cuales $a=c$ y $a>b$

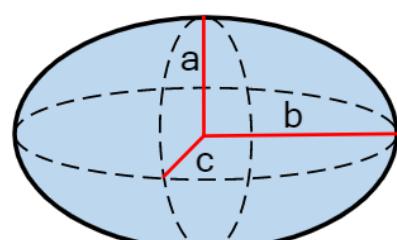


Figura 3.3 Parámetros de un elipsoide

3.3.2. SOFT GRIPPERS

Tradicionalmente los grippers se presentan como un conjunto de uniones de piezas rígidas con ejes que efectúen la acción de sujeción [18]. Sin embargo, un nuevo paradigma de actuación está siendo desarrollado con los denominados SG. Este tipo de sujetadores presenta numerosas ventajas en cuanto a su funcionamiento y mantenimiento. Además, su rendimiento es mayor a la hora de sujetar formas, como los objetos elipsoidales proláticos, que sería más complicado para los grippers.

Los materiales más usados para este tipo de sujetadores son los elastómeros ya que presentan las características necesarias para desempeñar el funcionamiento de este tipo de grippers. Entre los más usados se encuentran TPU y TPE [4], [9], [10].

Estos sujetadores pueden ser a su vez divididos según [18] en 3 categorías a partir de su tipo de actuación:

- Sujeción por acción: es este caso el gripper se puede amoldar a la forma que sujeta de manera controlada. Este tipo de actuador utiliza la respuesta de acción sobre el objeto para realizar la acción de sujeción. Se conocen también como grippers adaptativos [19] y cuentan con la ventaja de integrar fácilmente sensores de fuerza para tener reconocimiento al tacto de los objetos. Un ejemplo claro es el tipo de gripper basado en el efecto Fin Ray [9], [10].
- Dureza controlable: formado por materiales que permiten el control de su dureza. Su desventaja es que dependen del tiempo de reacción del material y por ello puede tener un funcionamiento lento.
- Adhesión controlada: este tipo de sujeción aprovecha las capacidades de adhesión de ciertos materiales para sujetar el objeto. La ventaja de estos es que no requiere del uso de presión para realizar la sujeción y por ende evita deformaciones por contacto [6].

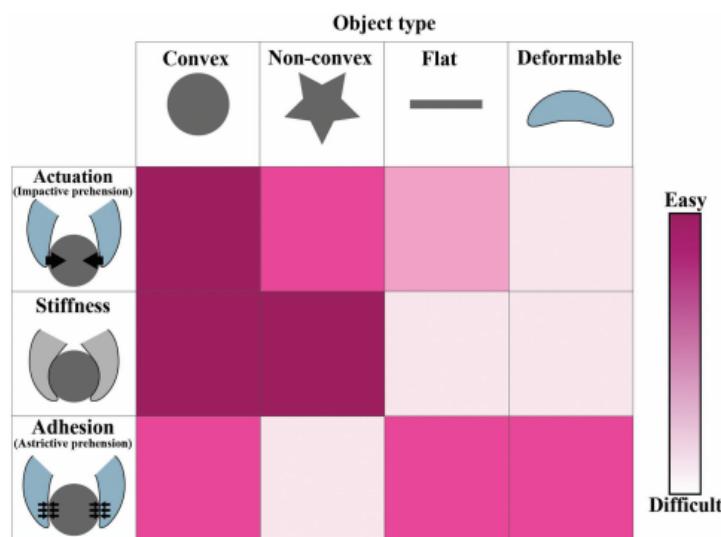


Figura 3.4 Clasificación SG [18]

El tipo de actuación del SG puede incluso determinar qué objetos puede sujetar (Figura 3.4). Cada tecnología tendrá sus ventajas y limitaciones. Incluso, en ciertos casos, suelen funcionar de manera combinada.

3.4. SENSORES DE FUERZA

El paradigma actual de desarrollo en robótica consiste en implementar soluciones que permitan satisfacer los requerimientos de adaptación a los cambios en el ambiente [20] de trabajo del robot. Uno de estos medios para resolver estos requerimientos viene dado por los sensores de fuerza que dan una perspectiva del tacto que tiene la máquina con el entorno de trabajo o el objeto que manipule.

Este tipo de sensores aprovechan el comportamiento piezoelectrónico de ciertos materiales para poder hacer la medición de distintas señales [21]. De esta forma se comportan como transductores que convierten la entrada de fuerzas mecánicas en señales analógicas. Por ello es que son utilizados normalmente para aplicaciones de HMI o MMI. Sin embargo, no presentan una medida de alta precisión de fuerza ya que su uso va orientado a la determinación de una constante que haga referencia a la fuerza aplicada dentro de un rango específico de funcionamiento. Es decir, que no reemplazan las galgas extensiométricas, pero son una buena solución para aplicaciones de robótica ya que su montaje es relativamente más sencillo y su coste es inferior [22].

3.4.1. FUNCIONAMIENTO

Los FSR se presentan como una resistencia dinámica que se relaciona con la fuerza que se le aplica sobre su superficie de contacto. Dicho cambio presenta un funcionamiento inversamente proporcional a la fuerza aplicada (Figura 3.5).

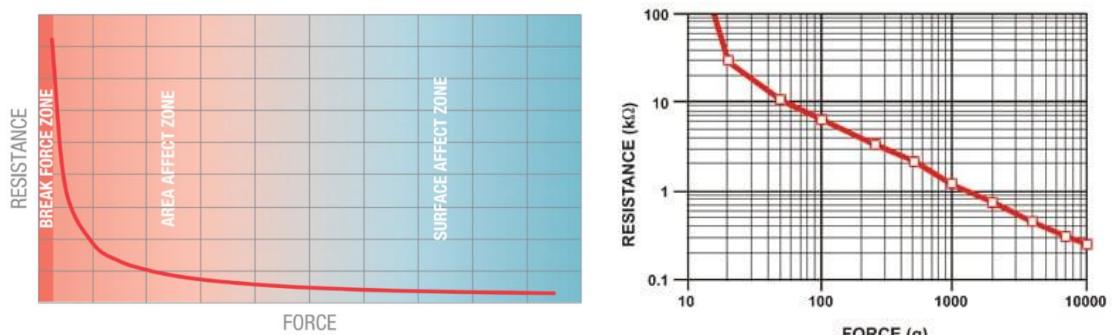


Figura 3.5 Resistencia de FSR vs Fuerza

En la imagen izquierda se muestra una gráfica de funcionamiento del FSR en escala lineal [23] y con sus tres mecanismos de funcionamiento. En la imagen de la derecha podemos ver una representación de la respuesta del FSR ante presión en escala logarítmica [22].

Existen 3 mecanismos que integran las características resistivas que describen la respuesta del sensor ante la presión [23] (Figura 3.5). Se dividen en:

- **Fuerza de quiebre:** fuerza umbral necesaria para tener contacto entre las dos membranas que forman el sensor. En este caso su resistencia se presenta como infinita, pero en la realidad el fabricante indica la resistencia a la que el sensor se encuentra. Suele estar en el orden de los $10\text{M}\Omega$.
- **Efecto de área:** en este caso la resistencia irá disminuyendo de forma inversa con la fuerza aplicada.

- **Región estable:** es la última parte del funcionamiento en dónde se observa que a pesar de que la fuerza aumente, la disminución de resistencia es casi nula.

3.4.2. MODOS DE FUNCIONAMIENTO

Los FSR se presentan en dos modos de funcionamiento [23]:

- Modo shunt: Estos sensores cuentan con un diseño basado en la construcción de 2 capas de polímero flexible. Su principal característica es que, si en este tipo de sensores es que, si se encuentra el circuito abierto o no existe ninguna fuerza aplicada, su resistencia tiende a infinito. Además, tienen un rango de medidas más amplio ya

Figura 3.6 Mecanismos de FSR [23]

cuenta con una curva de funcionamiento menos pronunciada después de la fuerza umbral. Por último, su coste suele ser más reducido que los de modo thru ya que cuentan con un número menor de capas.

- Modo thru: Estos tienen una mayor sensibilidad a fuerzas menores y un comportamiento más lineal a diferencia de los de Shunt. Sin embargo, debido a su proceso de fabricación tienen un coste mucho más elevado y el rango de medición suele ser más reducido, por lo que no se utilizan normalmente a menos que la aplicación así lo requiera.

3.4.3. CIRCUITO DE IMPLEMENTACIÓN

Para el uso de este tipo de sensores existen distintos circuitos recomendados [23]-[25]. En este apartado se estudiarán las posibles configuraciones que recomiendan los fabricantes para el uso de los sensores en modo shunt que es el que se va a usar para el SG.

3.4.3.1. DIVISOR DE VOLTAJE

El sensor es conectado en serie a una resistencia constante que será la que tenga la medida de caída de tensión que se quiere muestrear. Esta tensión que pasa por un amplificador seguidor para que esté aislado del MCU que muestrea la señal (Figura 3.7). Sin embargo, esta implementación puede realizarse sin hacer uso de un amplificador operacional. La resistencia R_M puede tener configuración de *pull-up* (R_M a V de alimentación) o de *pull-down* (R_M a tierra) y, según esto, se tendrán distintas ecuaciones que describen la salida de tensión que se va a medir. Si se tiene la configuración de *pull-up* la salida describe la Ecuación 3.22.

$$V_{out} = \frac{R_M V}{(R_M + R_{FSR})}$$

Ecuación 3.22

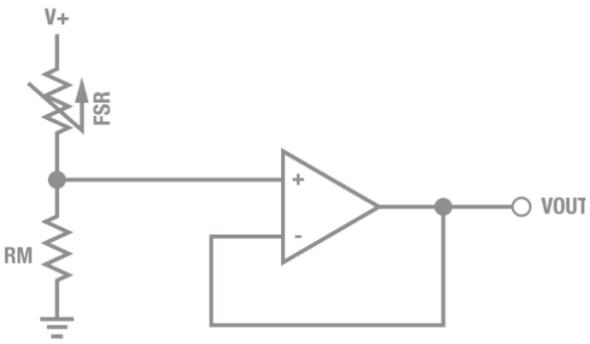


Figura 3.7 Divisor de voltaje FSR [23]

Cuenta con configuración de Pull-Down

Si se tiene una configuración *pull-down* la salida del voltaje cambiaría simplemente las resistencias de la Ecuación 3.22 tal como se muestra en la Ecuación 3.23.

$$V_{out} = \frac{R_{FSR}V}{(R_M + R_{FSR})}$$

Ecuación 3.23

En ambos casos de funcionamiento la resistencia R_M se elige para maximizar la sensibilidad requerida según el rango en el que se trabaje y a su vez para limitar la corriente que pasa por el circuito [23]. Su valor suele estar en el rango de los $10K\Omega$.

3.5. MOTORES PASO A PASO

Este tipo de componentes eléctricos se pueden describir como motores sin conmutadores y que suelen contar con un devanado que forma parte del estator. Sin embargo, el rotor depende del tipo de motor paso a paso que se utilice [26]. La característica más llamativa con la que cuentan estos motores es que pueden ser movidos un paso a la vez con el controlador adecuado. Además, son muy utilizados por su HT ya que en ciertas aplicaciones estáticas puede ser fundamental esta capacidad [27].

Una de las principales razones por las cuales se han elegido este tipo de motores, en contraposición con los servomotores, es que estos últimos requieren de una realimentación, normalmente de un potenciómetro, que le indique la posición del rotor para tener la corriente de control requerida para el movimiento necesario. Por ello la fiabilidad de los servos dependerá de del sistema de realimentación usado. Por otro lado, la capacidad de los SM va directamente relacionada con la geometría del rotor y por ende pueden ser utilizados con una configuración de bucle abierto ya que no presentan error de posicionamiento acumulado y sus movimientos son bastante precisos. Esto, como bien ya se ha dicho, utilizando un controlador apropiado (4.3.2.1). Sin embargo, para aplicaciones que requieran de grandes aceleraciones por tener cargas variables es posible que los pasos del motor se pierdan. En estos caso es aconsejable incluir un sistema de realimentación que provea de una lectura fidedigna de la posición de rotor.

3.5.1. FUNCIONAMIENTO ESTÁTICO Y DINÁMICO

Para caracterizar el funcionamiento de un SM se puede establecer según [26] que normalmente cuentan con un movimiento de S radianes por paso. De forma simplificada se puede representar el funcionamiento estático de estos motores como una sinusode que describe el torque en función de la posición angular en que se encuentra el rotor. Dicha sinusode tendrá un periodo de xS dónde x es doble del número de devanados o el número de imanes permanentes. Eso se puede describir con la Ecuación 3.24.

$$T = -h \sin\left(\left(\frac{\left(\frac{\pi}{x}\right)}{S}\right)\theta\right)$$

Ecuación 3.24

Donde: T = torque. h = torque de mantenimiento. S = ángulo de paso en radianes. θ = ángulo del eje. x = número característico del tipo de motor paso a paso

Sin embargo, en la realidad cada vez que el motor avanza un paso el equilibrio de esta representación estática se desplaza. Esto hace que exista un desfase respecto de la posición anterior [26]. Este desfase dependerá del par de arranque que se requiera para el motor. Es decir, que se presenta una característica dinámica en el momento de iniciar una trayectoria (Figura 3.8).

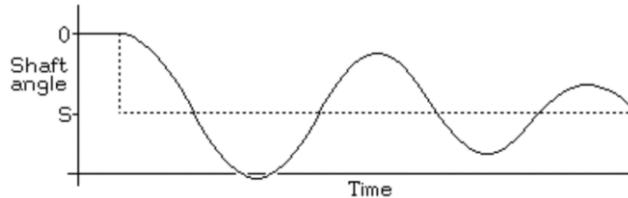


Figura 3.8 Dinámica en trayectoria un Motor por pasos [26]

Nota: la línea punteada hace referencia a el SP de S , donde S es la posición angular requerida. La línea curva sólida hace referencia a la trayectoria que sigue el motor para posicionarse en S .

3.5.2. MICROSTEPPING

Esta es una capacidad bastante importante de los SM ya que puede determinar el porqué de su elección y del controlador que lo maneje. Este concepto parte primero del *half-stepping*, que básicamente es la capacidad de duplicar el número de pasos del SM con la inclusión de una fase más de alimentación [26]. Es decir, que si se tiene un motor unipolar con un paso de S radianes y se incluye una fase más de forma controlada se consigue tener un paso de $S/2$ radianes.

El concepto de microstepping utiliza este comportamiento de los SM, pero llevado al extremo mediante la alimentación de los devanados de tal forma que la señal sea suavizada y se aumente el número de saltos que realiza en cada paso. Para ello es necesario el uso de un

controlador que cuente con un sistema de troceado (*chopper*) que se encargue de hacer este control por medio de PWM. Así es posible aumentar la resolución en el control del campo magnético que provoca el movimiento del motor durante el desplazamiento angular [26] y, por lo tanto, aumentar el número de pasos del SM. Llegando incluso a múltiplos de 256 micropasos.

3.5.3. TIPOS DE SM

El funcionamiento de los SM depende principalmente de como esté formado en su interior. Por esto se repasarán los tipos principales de SM a partir de lo explicado en [26].

3.5.3.1. RELUCTANCIA VARIABLE

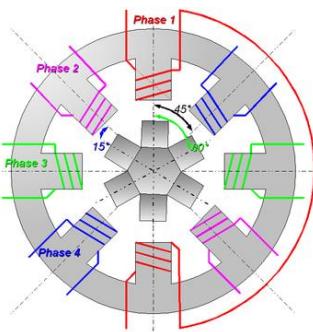


Figura 3.9 SM de reluctancia con 6 polos y rotor con 6 dientes

Nota: imagen extraída de
https://es.wikipedia.org/wiki/Motor_de_reluctancia_variable

Estos motores cuentan con un cable común para cada uno de los devanados que normalmente va a la alimentación positiva. Su modo de funcionamiento es realizar la alimentación de forma secuencial entre cada devanado para así producir un movimiento continuo [26].

El paso de estos motores depende del número de devanados que este tenga tal como se muestra en la Ecuación 3.25.

$$S = \frac{\pi}{n}$$

Ecuación 3.25

Donde: S = paso del motor en radianes. n = número de devanados

El movimiento en este tipo de SM es logrado con la alimentación secuencial de los devanados. La corriente crea un campo magnético que atrae al rotor, que es un imán permanente, para minimizar la reluctancia. El rotor cuenta con dientes que pueden tener distintos desfases [26] de tal forma que cuanto menor sea su desfase o mayor sea el número de dientes se podrá conseguir un paso de menor tamaño (Figura 3.9).

3.5.3.2. UNIPOLAR

Este tipo de motores cuenta con dos bobinas principales en el estator que están conectadas de forma que en la mitad de estas se tenga alimentación positiva y sus dos extremos se conecten alternativamente a tierra dependiendo de las necesidades de dirección que se requieran (Figura 3.10).

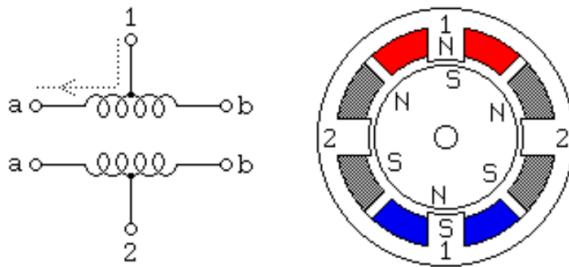


Figura 3.10 MS unipolar [26]

A diferencia de los motores de reluctancia variables estos motores no buscan minimizar la trayectoria de flujo magnético entre el estator y el rotor, sino que su movimiento se genera simplemente por la atracción de los polos N o S permanentes en el rotor con los polos del estator.

La resolución en este tipo de SM se logra aumentando el número de polos en el rotor. Es decir, que se podría calcular con Ecuación 3.25 tomando como n el número de pares de polos en el rotor [26].

Existen 2 tipos de motores que funcionan de esta forma: motores de imán permanente y motores híbridos. Los últimos cuentan con capacidades mayores de resolución y son los que normalmente se usan en aplicaciones de robótica y suelen llamarse NEMA (4.3.2.5).

3.5.3.3. BIPOLAR

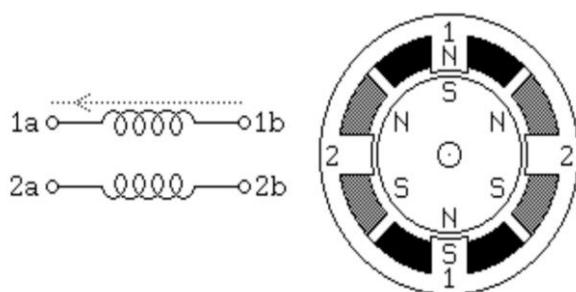


Figura 3.11 MS bipolar [26]

En este tipo de SM, a diferencia de los SM unipolares, no se requiere de una conexión entre los devanados hacia la alimentación [26]. De esta forma el cableado es mucho más sencillo, pero requiere de una mayor capacidad de control para poder revertir el flujo de corriente. Esta capacidad se logra con un controlador de puente en H [26] que permite variar la polaridad de cada devanado de forma independiente (Figura 3.11).

3.5.4. CONTROL DE UN SM

Un sistema de control debe estar basado en modelos para poder tener una representación del funcionamiento que tendrá el sistema en la realidad. Por esta razón se verá el modelo electromagnético que tiene un motor para que pueda ser controlado con un sistema de realimentación en bucle cerrado (Figura 3.12).

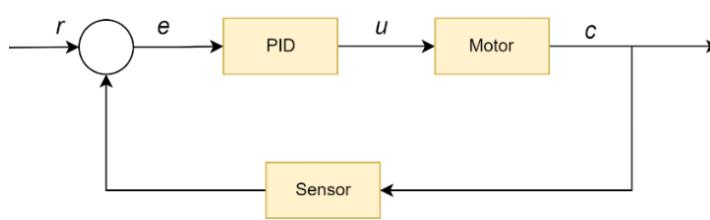


Figura 3.12 Planta de control con realimentación y PID [28]

*r: señal de control de entrada. Conocida como SP
e: error existente en la señal de control y el valor actual
u: señal de control en base al error e
c: señal de posición del motor*

En un sistema básico de control por bucle cerrado es necesario tener los siguientes componentes [28] (Figura 3.12). Se pueden resaltar:

- PID: va a ser la capa de SW que se encargue de proporcionar la señal de control en base al error que se registre entre SP y la señal proveniente del sensor. En este sentido constará con un dominio discreto que dependerá del HW en el cual se ejecute la aplicación de control.
- Motor: recibirá los comandos directamente desde el controlador de motores (4.3.3.1) quien será el encargado de realizar los cambios de alimentación necesarios para hacer que el motor realice el comando de velocidad o posición que se haya establecido.
- Sensor: devuelve la señal de realimentación que se registra del motor. En este caso no se ha incluido sensores reales en el sistema, sino que se ha aprovechado de los registros de posición del controlador de los motores.

Tomando esto en cuenta es posible desarrollar lo que sería la planta para este sistema en 2 partes: modelo mecánico y modelo eléctrico. Estos dos pueden finalmente unirse para resultar en el modelo electromecánico de forma de tener todo su funcionamiento simplificado y unificado [29]. Estos se verán de forma general ya que esta etapa de control queda cubierta con el controlador de motores (4.3.3.1).

3.5.4.1. MODELO ELÉCTRICO

Según [30] un motor se puede modelar como un circuito LRC para simplificar el sistema. Sin embargo, este modelo cuenta con una fuerza contraelectromotriz e_b como respuesta a la inducción de un voltaje [31] (Figura 3.13). Finalmente, de acuerdo con la ley de Voltaje de Kirchhoff, queda la Ecuación 3.26.

$$u_m = R_m i(t) + L_m \frac{di(t)}{dt} + e_b$$

Ecuación 3.26

Donde u_m : tensión de entrada, R_m : resistencia terminal, L_m : inductancia terminal, i : corriente.

Es necesario recordar que este modelo eléctrico corresponde a cada uno de los devanados que compongan el motor. Además, esta es una simplificación ya que existen modelos de alta frecuencia que toman mayores parámetros para modelar la potencia activa disipada. Sin embargo, esta no es la finalidad de este trabajo.

3.5.4.2. MODELO MECÁNICO

Para este modelo se deben incluir todos los aspectos que afectan a la transmisión de la energía de forma mecánica: rozamientos, inercias, torques, momentos. etc. De forma simplificada es posible modelar un motor como un sólido rígido al cual le afectan distintos momentos. Entonces, la ecuación mecánica del motor según [31] es la que se muestra en Ecuación 3.27.

$$\tau_m(t) = J_m \ddot{\theta}_m(t) + B_m \dot{\theta}_m(t) + \tau_c(t)$$

Ecuación 3.27

Donde θ : posición angular, $J_m \ddot{\theta}_m$: par de inercia del motor, $B_m \dot{\theta}_m$: par de fricción viscosa, τ_c : par que ejerce la carga (Figura 3.13).

3.5.4.3. MODELO ELECTROMECÁNICO

Este, como ya se explicó, representa la síntesis de los dos modelos anteriormente mostrados. Para su desarrollo se considera que un motor DC satisface las relaciones de acople electromecánico [31] mostradas en la Ecuación 3.28 y la Ecuación 3.29.

$$e_b = k_b \dot{\theta}_m(t)$$

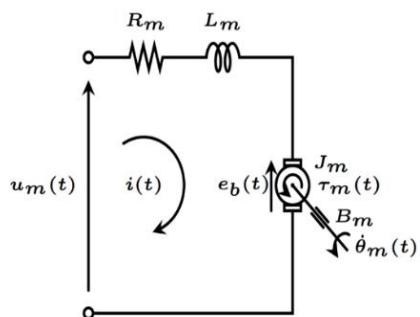
Ecuación 3.28

$$\tau_m = k_m i(t)$$

Ecuación 3.29

Donde k_b : constante de fuerza electromotriz, k_m : constante de par.

Gracias a estas 2 relaciones se pueden construir varias funciones de transferencia entre las cuales las más relevantes serán aquellas que relacionen el voltaje de entrada con la velocidad angular y la corriente. A partir de Ecuación 3.28 y Ecuación 3.29 y considerando condiciones iniciales nulas se obtienen las siguientes relaciones:



$$u_m = R_m i(t) + L_m \frac{di(t)}{dt} + k_b \dot{\theta}_m(t)$$

Ecuación 3.30

$$\tau_m(t) = J_m \ddot{\theta}_m(t) + B_m \dot{\theta}_m(t)$$

Ecuación 3.31

Figura 3.13 Modelo electromecánico de motor DC [31]

Con Ecuación 3.30 y Ecuación 3.31 se cuenta con un modelo completo de un motor eléctrico (Figura 3.13).

3.6. PROTOCOLOS SERIALES DE COMUNICACIÓN

En los sistemas digitales modernos es necesaria la comunicación entre los distintos periféricos y IC que en él se encuentren funcionando. Para esta comunicación se suelen utilizar

interfaces seriales que se encarguen de enviar los paquetes de datos siguiendo un protocolo [32]. Entre los más populares están el protocolo I2C y el SPI.

3.6.1. SPI

Desde sus inicios con Motorola [32] la interfaz SPI se definió como un bus serial de comunicación que constaba de 4 cables correspondientes con las señales que lo conforman. Actualmente su arquitectura se ha extendido para numerosos sistemas digitales [33], [34] y es una base fundamental de las comunicaciones a nivel HW.

Una de las principales ventajas del uso de este protocolo es que cuenta con la capacidad de tener transmisiones de alta velocidad en baja distancia con una interfaz sencilla. También, es posible realizar comunicaciones *full-duplex*, es decir, que se puede enviar y recibir al mismo tiempo información. Esto lo hace uno de los protocolos seriales más rápidos. Sin embargo, al contar con una interfaz sencilla no maneja controles de flujo ya que no existen las señales de verificación de recepción de datos [32] y, al no utilizar direccionamiento para la comunicación, requiere de un mayor número de cables para hacer comunicaciones entre más de dos dispositivos.

3.6.1.1. TOPOLOGÍAS

Cuando se habla de una interfaz SPI es necesario tomar en cuenta el modelo de maestro y esclavo. Esto se debe a que siempre habrá un dispositivo que se encargue de coordinar las comunicaciones y otro que se encargará de obedecer las peticiones de este. Por ello existen 2 topologías (Figura 3.14) que difieren en función del número de dispositivos que estén conectados. Estas son:

- Maestro conectado a un solo esclavo.
- Un maestro conectado a numerosos esclavos. El control de la comunicación dependerá de la activación del SS correspondiente.

Hay que tener en cuenta que no es posible tener varios maestros en una misma línea ya que la conexión solo depende del SS seleccionado. Si ambos maestros habilitan al mismo esclavo existirán conflictos de comunicación. Eso se podría evitar con I2C.

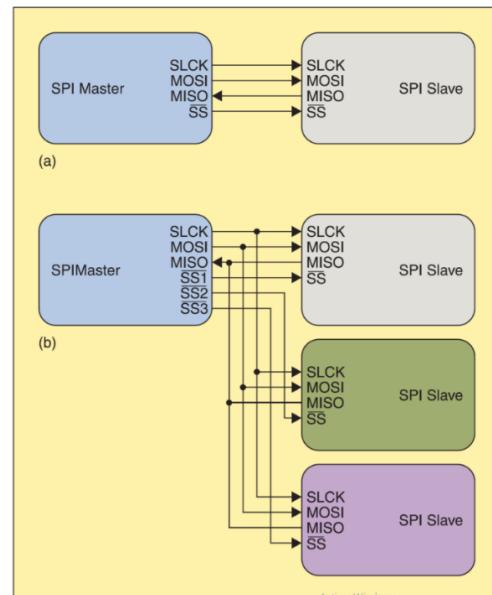


Figura 3.14 Topologías de comunicación SPI [32]

- a) *Un maestro conectado a un esclavo*
- b) *Un maestro conectado a varios esclavos*

3.6.1.2. SEÑALES

Como bien ya se ha visto, en la interfaz SPI están actualmente definidas 4 señales principales:

- Reloj: esta es la salida del dispositivo maestro y entrada de los esclavos. Determina el flanco de reloj que se use para interpretar los bits de datos. Todas las señales que usa el protocolo serán síncronas a este pulso de reloj. Se suele representar como SLCK
- MOSI: es la señal de salida de datos del maestro y por ende la entrada del esclavo seleccionado.
- MISO: corresponde con la señal de salida del esclavo y con la entrada del maestro.
- SS: corresponde con la señal de habilitación de la comunicación con el esclavo correspondiente. Por lo tanto, serán necesarias tantas SS como esclavos se quieran controlar o, en su defecto, utilizar un multiplexor que aumente el número de salidas (Ver Figura 3.14 b). Esta señal suele estar negada.

En un funcionamiento completo de comunicación por SPI existen los canales MOSI y MISO que manejan datos entre el esclavo y el emisor de manera simultánea con un tamaño de 8bits (algunos dispositivos soportan tamaños de palabra de 16 bits). Dichos datos son muestreados en el flanco de bajada del reloj. Además, antes de iniciar la comunicación el SS es puesto a cero para habilitar el canal de transmisión maestro-esclavo (Figura 3.15).

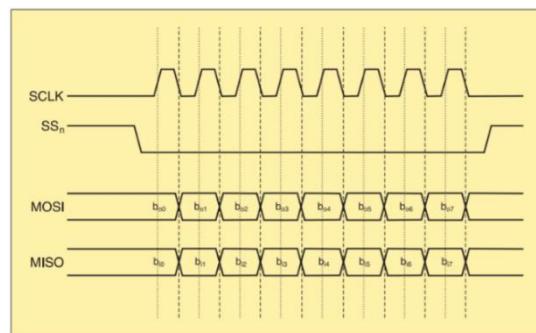


Figura 3.15 Ejemplo comunicación SPI [32]

3.6.1.3. MODOS DE FUNCIONAMIENTO

Es importante tomar en cuenta, antes de empezar a realizar una transmisión SPI, el modo de comunicación que se debe utilizar entre ambos actores de la línea serial. Es posible que el funcionamiento difiera según el componente que se use, por ello, se debe asegurar que ambos dispositivos tengan el mismo modo de comunicación [32] entre los 4 estandarizados actualmente (Figura 3.16).

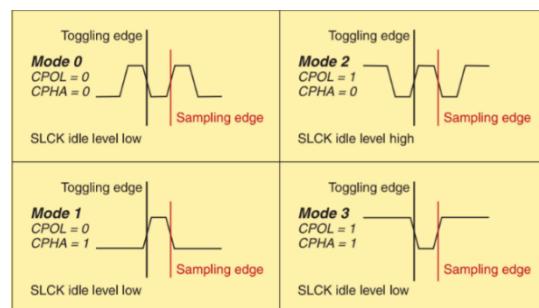


Figura 3.16 Modos SPI [32]

Estos modos definen comportamientos en base a SLCK según los siguientes puntos:

- Flanco de lectura del valor en las líneas de datos: Se determina con la fase del reloj representada con CPHA. También depende de la polaridad del reloj.
- Flanco de cambio de valor en las líneas de datos: Se determina con la fase del reloj representada con CPHA. También depende de la polaridad del reloj.
- Nivel de mantenimiento de la señal SLCK: Se suele llamar polaridad del reloj y se representa como CPOL. Siendo 1 cuando existe polaridad positiva o el nivel SLCK se mantiene en nivel alto y 0 en la situación contraria.

3.6.2. I2C

Este otro protocolo serial se basa en una interfaz de comunicación *multi-master* que consta únicamente de 2 (Figura 3.19) cables a través de los cuales se va a transmitir la información y el reloj que gobernará la comunicación [32]. Para ello se basa en la posibilidad de manejar el direccionamiento de la comunicación entre ambos actores con el uso de un primer byte que cuenta con la dirección correspondiente al módulo receptor (Figura 3.17). De esta forma se establece como maestro el dispositivo que comience la comunicación. Es decir, no hace falta utilizar ningún pin digital para habilitar la transmisión como en SPI. Sin embargo, el gran problema que presenta esta interfaz es que cuenta con una máquina de estados que controla su funcionamiento y por ende necesita una mayor cantidad de datos que tengan la información del proceso comunicativo (Figura 3.17). Todo ello conlleva a que con el I2C no se pueda superar, actualmente, los 400kHz de funcionamiento.

3.6.2.1. TOPOLOGÍAS

Esta comunicación se basa en el esquema básico de maestro y esclavo (Figura 3.18) por ello se pueden conseguir las mismas configuraciones de SPI (Figura 3.14). Además, por sus capacidades de direccionamiento, puede soportar formatos que cuenten con múltiples maestros en las líneas de comunicación.

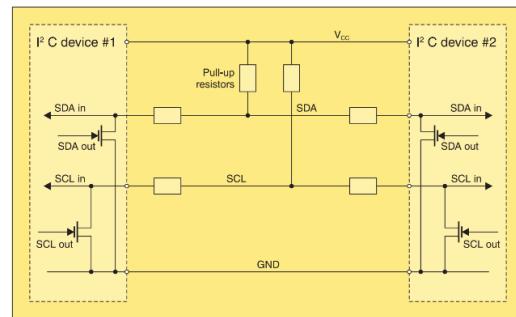


Figura 3.18 Esquema de conexión maestro esclavo I2C [32]

3.6.2.2. SEÑALES

Para poder tener conocimiento del proceso de comunicación y de la información transmitida el I2C cuenta solamente con dos señales (Figura 3.19):

- SDA: línea bidireccional de comunicación de datos en bytes
- SCL: control de reloj de la aplicación

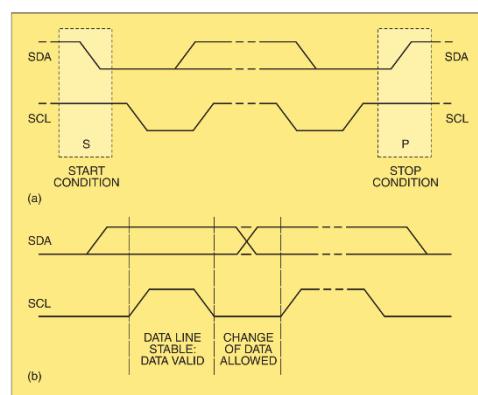


Figura 3.19 Señales I2C

3.7. MODELO CLIENTE-SERVIDOR Y COMUNICACIÓN RED

Este tipo de modelo de comunicación es un sistema distribuido donde el SW está dividido en dos principales actores: el cliente, quien sería el dispositivo que solicita la información o la ejecución de algún servicio, y el servidor, quien es el dispositivo que responde ante las peticiones del cliente. Por esta razón debe existir una separación clara de las responsabilidades entre ambos actores (Tabla 3.2).

Tabla 3.2 Actores en comunicación red

Actor	Funciones
Cliente	- Iniciar diálogo - Enviar peticiones a servidor según protocolo de comunicación - Pedir información o acciones al servidor
Servidor	- Esperar peticiones de los clientes - Responder a las peticiones según las capacidades y política establecida

Para tener el sistema basado en este modelo de comunicación hay que tomar en cuenta todas las capas SW y HW que hay que incluir. Por esto se verán los puntos más relevantes que son necesarios para tener este tipo de comunicación: nivel físico, nivel de transporte y nivel de implementación.

3.7.1. TIPOS DE REDES

Hay que tener la cuenta que puede haber numeroso clientes conectados a un servidor o incluso un cliente conectado a más de un servidor. Este tipo de modelos no tiene límite siempre que el HW y SW sea capaz de soportar las solicitudes. Además, estas conexiones que existan entre ambas partes pueden darse de dos formas [35] (Figura 3.20):

- **Redes locales:** también denominadas LAN son aquellas redes que permiten el intercambio de información entre equipos o dispositivos que estén interconectados en un mismo lugar físico. Normalmente comparten la IP 198.x.x.x.
- **Redes externas:** son aquellas que no se encuentran en una red local sino una WAN. Para su uso cuentan con una IP fija o de un dominio único.

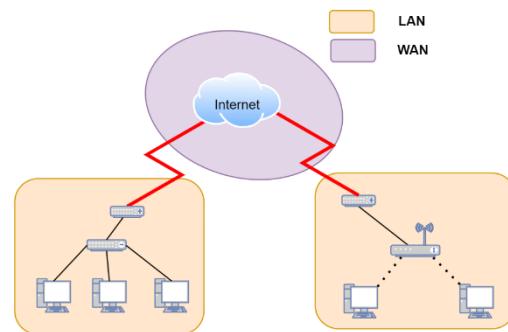


Figura 3.20 LAN-WAN

3.7.2. MODELO OSI

Para que exista una comunicación entre el cliente y el servidor debe haber un medio o canal a través del cual viajen los datos y luego estos sean entendidos y decodificados para ser usados en la aplicación receptora. Todos estos requerimientos vienen reflejados en el modelo OSI [36] que se compone de 7 niveles de abstracción que cuentan con sus propias funciones de forma que se alcance un objetivo común (Figura 3.21) de transmisión de datos.

Por esto es necesario, para contar con un modelo OSI en la aplicación, primero tener un nivel físico que sea el medio de transmisión de todos los datos, luego un tipo de empaquetado para ser enviado a la red y, finalmente, una capa de decodificación del mensaje que el anfitrión pueda utilizar en el mantenimiento de la sesión si es necesario o para recibir los datos que se requieran [36]. Sin embargo, las capas a utilizar van a depender del tipo de protocolo de comunicación que se utilice y los requerimientos de la aplicación final.

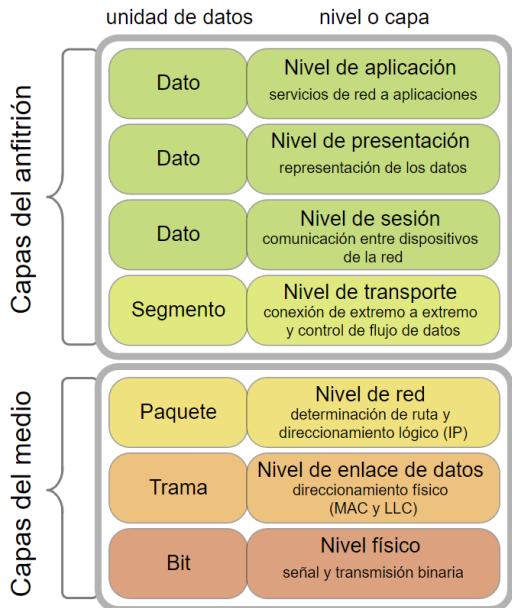


Figura 3.21 Modelo OSI

Imagen tomada de:
es.wikipedia.org/wiki/Modelo_OSI#/media/Archivo:OSI_Model_v1.svg

3.7.3. PROTOCOLO TCP/IP vs UDP/IP

Los datos en las aplicaciones deben ser empaquetados y transportados al servidor que le ofrezca los servicios necesarios para una tarea en específico. Esta capa de transporte, descrita en el nivel 4 del modelo OSI (Figura 3.21), es la responsable de realizar la transmisión de información entre los puntos de comunicación, entiéndase cliente y servidor. Para este propósito existen protocolos de comunicación dedicados para la capa de red IP que se encarga de hacer el enrutado de las direcciones de comunicación[37]. Entre los protocolos más populares están el TCP y el UDP. Sin embargo, cada aplicación puede tener distintos requerimientos y, en base a estos, se puede escoger el protocolo más indicado. Por ello se verán las principales diferencias que existen entre ambos protocolos. Según [37] se tienen:

- El TCP requiere de realizar el establecimiento y fin de la conexión. Por ende, necesita de un mayor número de transacciones para comenzar o terminar una comunicación. En UDP la comunicación se hace de forma asíncrona.
- El TCP requiere de un reconocimiento de mensaje. De esta forma puede saber si la conexión sigue activa y además si se ha transferido la información de forma correcta. Por otro lado, UDP no requiere del reconocimiento del mensaje.

- TCP cuenta con control de flujo, por ende, emite los datos según el orden de envío. El UDP no utiliza control de flujo por lo que envía los datos en el momento en que los recibe.
- TCP tiene un mayor tamaño de mensaje ya que necesita más información en el control de conexión. Esto hace que el tamaño de datos a enviar sea mayor y, por lo tanto, a que la comunicación sea un poco más lenta que con UDP.

En general se puede establecer que la diferencia entre uno y otro protocolo recae fundamentalmente en la fiabilidad. Es evidente que el protocolo TCP cuenta con unas mayores prestaciones en cuanto a la seguridad entre las comunicaciones. Sin embargo, si se requiere una latencia baja para una gran cantidad de datos a procesar puede ser más ventajoso el uso de UDP si no es necesario un control de sesión. Esto, como bien se dijo con anterioridad, va a depender del requerimiento de la aplicación.

3.7.4. SOCKETS EN UNIX

Para establecer las comunicaciones a través de IP se suele seguir el modelo de socket de UNIX (Figura 3.22) que se basa en la filosofía cliente-servidor. Esta implementación se hace normalmente con el uso de una API que abstrae todos los métodos y servicios de UNIX y que permite tener la funcionalidad correspondiente de una forma sencilla sin necesidad de ir a bajo nivel para poder tener los sockets funcionales. Es decir, que los sockets suelen estar cubriendo lo que sería la capa de sesión del modelo OSI (Figura 3.21) que sirve para decodificar los paquetes recibidos en formatos UDP o TCP para ser utilizados por los niveles superiores. Por esto es que, dependiendo del protocolo que se utilice, puede que no sea necesario el uso de algunas de sus funciones. Por ejemplo, si se está usando UDP no será necesario establecer la conexión antes de enviar algún paquete de datos. Es decir, que por parte del servidor no hace falta esperar conexiones sino solamente esperar alguna recepción en el socket y desde el cliente no hace falta hacer la conexión, sino simplemente empezar a enviar datos al socket correspondiente.

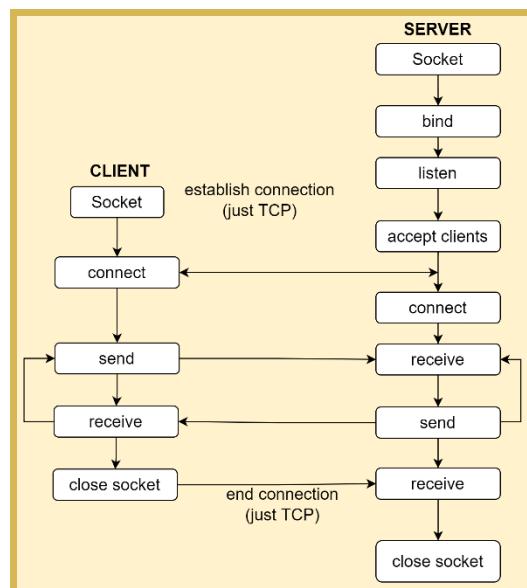


Figura 3.22 Cliente Servidor UNIX

Los dos tipos de socket en el dominio UNIX más usados son:

- Orientados a conexión: dependen de las conexiones que se establezcan. Es decir, que existe un canal dedicado para la comunicación entre cliente y servidor. Usa TCP/IP.
- Orientados a datagramas: no existe un canal dedicado entre cliente y servidor. Usa UDP/IP.

3.7.5. PAQUETE JSON

Este empaquetado de datos se presenta como uno de los más populares formatos de envío de mensajes en comunicaciones seriales y de red ya que es muy ligero y cuenta con una estructura jerárquica que le aporta más información a la trama enviada. Actualmente es uno de los medios de intercambio de información más utilizados en WWW y de los sistemas de bases de datos [38]-[40]. Por esta razón se ha escogido como paquete de información que contendrá los datos que quieran ser enviados en la comunicación entre los módulos SW que conformar la aplicación del SCARA (5.5).

3.7.5.1. FORMATO

Hay que resaltar que el formato JSON es una extensión de documento y como tal cuenta característica clave para poder efectuar su lectura [38]. Por ello es que para construir un mensaje en formato JSON se deben cumplir los siguientes puntos:

- Para empezar un mensaje o nivel de jerarquía se usa el carácter { y para terminarlo el }
- Las palabras clave de los atributos van entre comillas
- Los valores de los atributos pueden ser:
 - o Enteros: no necesitan comillas
 - o Strings: deben ir entre comillas
 - o Vectores: deben ir entre [], separado por comas, y seguir las mismas reglas anteriores

Ejemplo: se quiere construir un mensaje JSON que cuente con la información de la ubicación actual por GPS (Figura 3.23)

```
{  
    "type": "coordinates",  
    "data": {  
        "country": "Spain",  
        "city": "Madrid",  
        "street": "Ronda de Valencia",  
        "number": 3,  
        "GPS": {  
            "lat": "40.405514",  
            "lon": "-3.700013"  
        }  
    }  
}
```

Figura 3.23 Ejemplo paquete JSON

3.7.5.2. VENTAJAS

- Facilidad de organización de datos
- Posibilidad de expansión de forma sencilla
- Soporte de numerosas aplicaciones de comunicación
- Facilidad de interpretación de datos
- Básico control de flujo

Capítulo 4. DESARROLLO MECÁNICO DE ROBOT SCAR

En este capítulo existen 2 puntos principales a tratar: diseño CAD y Requerimientos HW. Primero se evaluará la evolución del diseño en CAD y el porqué de las elecciones tanto en materiales como en cambios estructurales. Posteriormente, con los requerimientos mecánicos establecidos, se verán los componentes eléctricos y electrónicos que se han usado para la construcción del robot.

4.1. DISEÑO CAD Y REQUERIMIENTOS MECÁNICOS

Se estudiarán las 2 partes fundamentales que conforman el robot: SG y cuerpo de SCARA. En ambos casos se detallarán los cambios principales que se realizaron y que fueron más relevantes para tener un robot funcional. Para elaborar todos los modelos que se verán a continuación se ha hecho uso del SW de CAD Fusion360 de Autodesk¹.

4.1.1. PROTOTIPO DE GRIPPER

Con base en el efecto del Fin Ray [9], [10], [41] se realizó una prueba de los diseños principales que se han visto para contar con el funcionamiento requerido. Es por lo que se verá cada una de las versiones del prototipo que se hizo hasta conseguir la que cubría los requerimientos de funcionamiento. Estas pruebas se realizaron en una escala de 2:3 para ahorrar material de impresión y tiempo de fabricación.

- **Primera versión:** Se elaboró una primera versión del SG como prototipo para ver el modo de funcionamiento y sus limitaciones (Figura 4.1). El problema con este diseño fue su rango de movimiento. Al ser la primera aproximación no permitía sujetar objetos de más de 20 mm de diámetro.

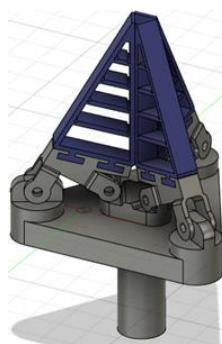


Figura 4.1 Primera versión SG

- **Segunda versión:** Teniendo en cuenta los problemas anteriores se hacen los cambios necesarios para cubrir un mayor rango de movimiento. Principalmente el problema estaba en el ángulo que formaban las partes móviles (Figura 4.2) del SG. Esto provocaba un bloqueo en un punto máximo del movimiento normal de la pieza.

¹ <https://www.autodesk.es/products/fusion-360/overview>

Con este simple cambio se logró un mayor rango de actuación. Sin embargo, aún se notaba que su movimiento no era del todo completo.

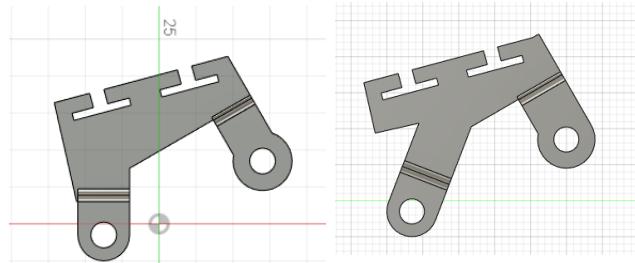


Figura 4.2 Comparación V1 y V2 SG

A la izquierda está la versión 1 del SG con un ángulo de acción de 90 grados. A la izquierda se encuentra la versión 2 con un ángulo de 45 grados en el punto de acción.

- **Tercera versión:** en el nuevo planteamiento se incluye un nuevo punto de pivote que aporta mayor movilidad a las partes rígidas del mecanismo (Figura 4.3). Con ello es posible lograr un movimiento completo de la pieza para poder hacer sujeción de objetos de mayor tamaño. Sin embargo, la base estaba siendo un problema para el movimiento ya que impedía que las partes flexibles se cerraran completamente. Con este prototipo (Figura 4.3) se lograron hacer las pruebas de funcionamiento primarias para evaluar la sujeción y el tipo de actuador que se requería para este caso (Figura A3. 1).

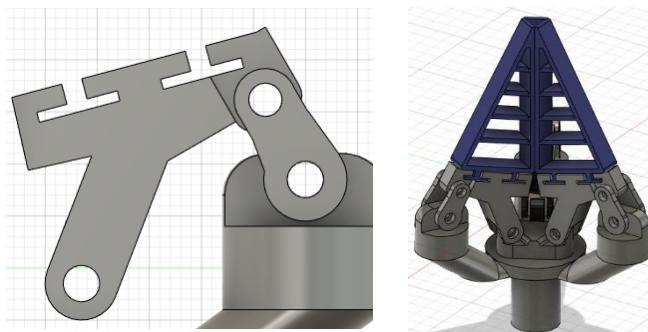


Figura 4.3 versión 3 de SG

A la izquierda se muestra el sistema de sujeción con 2 articulaciones. A la derecha se ve el SG de la tercera versión montado completamente

4.1.2. SOFT GRIPPER DEFINITIVO

Una vez se comprendió el modo de funcionamiento que tiene el SG se puede llegar a un diseño más completo y simplificado del mismo. Para ello se usó la última versión de lo que se denominará pieza flexible (SG_FLX) del mecanismo. Esta pieza con una estructura básica que permite contar con un funcionamiento completo del efecto FinRay. Por otro lado, se procede a eliminar el material y mecanismos de sujeción que presentaron problemas en la última versión del prototipo del SG para tomar únicamente las partes de mayor interés de la base y trabajar sobre ella para llegar a un diseño que no requiera de tornillos.

En ambos casos se realizaron múltiples iteraciones que fueron descartadas por problemas evidentes de funcionamiento pero que poco a poco fueron dándole forma a un modelo bien definido y que al final se planteó como definitivo (Figura 4.4). Este último diseño está dividido en 2 partes: desarrollo de pieza flexible (SG_FLX) y desarrollo de base de fijación (SG_BS). Esto es debido a que, a pesar de que



Figura 4.4 SG definitivo

ambas piezas debían estar integradas, sus funcionalidades y materiales de fabricación son bastante diferentes.

4.1.2.1. BASE DE SG

El problema principal del prototipo (Figura 4.3) es que cuenta con un gran número de partes móviles que requieren de mayor cantidad de tornillos y piezas separadas. Esto se traduce en un mayor peso de la estructura y también un mayor número de posibles fallos por aflojamiento de las partes móviles que pueden ser bastante perjudiciales para el funcionamiento. Es por ello por lo que la base del SG fue planteada como una estructura única que constara con la posibilidad de sujetar las partes flexibles. De esta forma el pivote del mecanismo sería efectuado aprovechando la flexibilidad de las piezas de sujeción [9].

Para este propósito se dividió SG_BS en partes:

- **Actuador (SG_BS_ACT):** móvil que ejercerá la fuerza requerida para la sujeción y que está conectada al motor del SG
- **Base fija (SG_BS):** encargada de sujetar los extremos no móviles de la pieza flexible

Una vez aclarado estos puntos se puede estudiar las 4 versiones principales de dicha base:

- **Primera versión:** se desecha el mecanismo de movimiento del prototipo (Figura 4.5). Se mantiene todo el funcionamiento conocido anteriormente con las correspondientes adaptaciones de las partes flexibles nuevas (Figura 4.5). Con esta primera versión se encontró un funcionamiento bastante bueno de sujeción (Figura A3. 2). Además, era evidente que, al contar con menos piezas y no tener tornillos, el movimiento era mucho más fluido a la hora de sujetar objetos de distintas formas.

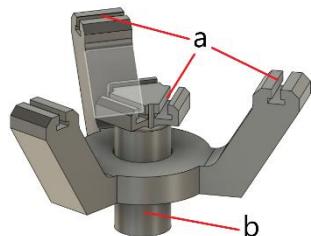


Figura 4.5 Primera versión de base definitiva para SG

En la imagen se representa:

- a) Entrada de las piezas SG_FLX
- b) Pieza que realiza la acción SG_BS_ACT

- **Segunda versión:** al intentar incluir la primera versión en el cuerpo del robot no se encontró un método viable de sujeción entre ambas partes que no requiriera un cambio en el diseño. Esto fue de la mano con el desarrollo del tercer eslabón (4.1.4.4) ya que debía de integrarse con esta pieza de una forma coherente y segura. Es por lo que, en un primer lugar, se optó por incluir un sistema de fijación basado en una rosca (Figura 4.6) que permitiera el fácil ajuste de la base del SG al tercer eslabón de una forma sencilla y rápida. Además, no requiere de tornillos ni pegamentos para fijar ambas partes.

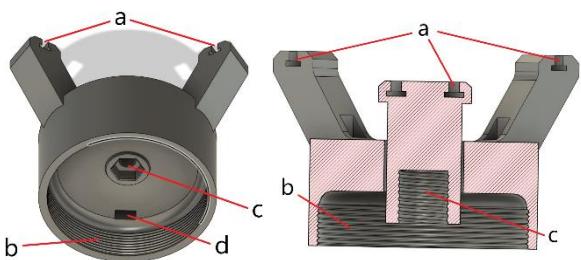


Figura 4.6 Versión 2 de SG

A la izquierda se puede ver la base con más material para rosca. A la derecha se ve un corte transversal de las dos piezas de la base. Se aprecian:

- a) Sistema de sujeción de SG_FLX
- b) Rosca para sujeción con ESB3
- c) Rosca para sujeción con el motor del SG
- d) Canal para cableado de FSR

- **Tercera versión:** con la versión anterior se tuvieron buenos resultados de funcionamiento y además se consiguió integrar la base con el cuerpo completo del robot. Sin embargo, surgió un problema a la hora de instalar los FSR (3.4) en el SG. Debido a que los sensores van alojados en las partes de contacto de las piezas flexibles y para su uso requieren de un cable que recorra el interior del sistema hasta llegar al MCU. El diseño ya constaba de agujeros específicos para este propósito (Figura 4.6) pero cuando se intentó hacer la instalación completa era evidente que al utilizar el sistema de rosca los cables se enredarían y podrían perjudicar el desempeño del actuador. Por esta razón se eliminó el sistema anterior y se cambió por uno de 3 tornillos para su fijación (Figura 4.7).

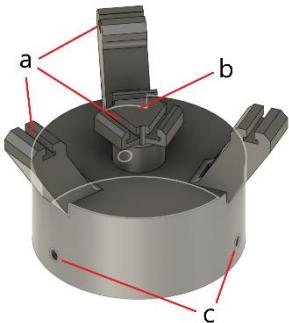


Figura 4.7 Base de SG con sujeción por tornillos

En la imagen se muestra:

- a) Sistema de sujeción de SG_FLX
- b) Pieza SG_BS_ACT
- c) Sistema de sujeción por 3 tornillo M4 con desfases de 120°.

- **Cuarta versión:** en este punto ya se cuenta con un diseño claro y un funcionamiento fluido del mecanismo. Con ello se pueden hacer modificaciones de optimización para reducir el material usado y, por ende, el tiempo de impresión (4.2.1). También se agrega un mecanismo de ajuste de la base con el eslabón 3, de forma que se acople alineadamente con los tornillos, lo que facilita en gran medida el proceso de instalación y producción del SG (Figura 4.8).

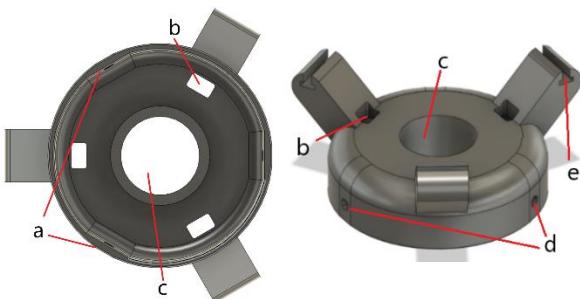


Figura 4.8 Cuarta versión de SG

En la imagen se puede ver la pieza optimizada para la base del SG. Se representa:

- a) Patrón de fácil encaje con ESB3
- b) Canal para cables de FSR
- c) Canal de SG_BS_ACT
- d) Entrada de tornillos M4

4.1.2.2. PIEZA FLEXIBLE DE SUJECIÓN (SG_FLX)

En el prototipado del SG se ha planteado un diseño sencillo y básico que sirvió para estudiar el mecanismo de acción de este tipo de sujetadores. En ese caso se hizo evidente que el material necesario para la fabricación de esta pieza iba a ser un punto muy importante a tomar en cuenta para su correcto funcionamiento (4.2.1.1). Por otro lado, es indispensable que el diseño cuente con las características correspondientes que permitan realizar el efecto Fin Ray

que ya se ha mencionado [9], [10], [41]. Por esta razón es que se tuvieron que evaluar distintas estructuras para esta pieza. Así se pudo comprobar cuál de ellas presentaba las mejores prestaciones. Entre las principales versiones se encuentran:

- **Primera versión:** en este caso la propia estructura flexible cuenta con una forma de triángulo equilátero (Figura 4.9) con membranas internas que ayudan a causar el efecto esperado de sujeción.

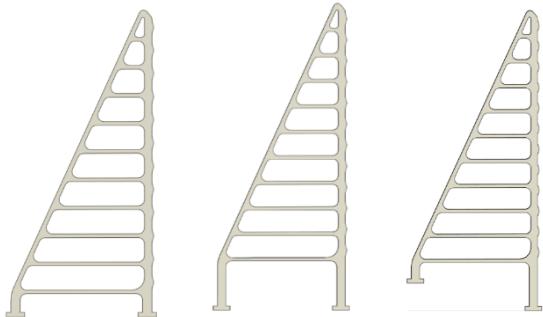


Figura 4.9 Primera versión de Pieza flexible

A la izquierda se ve la versión base. En el medio se ve el alargamiento de puntos de sujeción. En la imagen de la izquierda se tiene una combinación de las anteriores

- **Segunda Versión:** la primera versión presentó problemas de soporte ante tensiones. Por lo que se plantea la inclusión de una base más rígida (Figura 4.10) en la sujeción de la pieza con la base para que la estructura presentara mejor resistencia ante deformaciones.

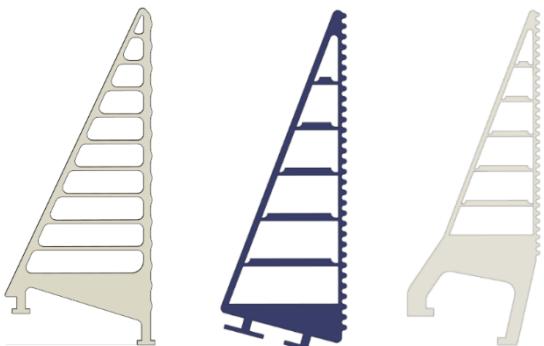


Figura 4.10 Segunda versión de pieza flexible con base rígida

A la izquierda aparece el diseño probado con la última versión de la base. Las otras dos fueron probadas con el prototipo de base.

- **Tercera versión:** debido a que se ganó rigidez se perdió flexibilidad porque el ángulo de acción provocaba que el efecto que se busca no se produzca de forma sencilla. Esto dificulta la acción del motor ya que este tiene que vencer dicha fuerza de tensión que no aporta beneficios de funcionamiento. Además, esta resistencia de la pieza podría llegar a ser contraproducente en la sujeción de objetos delicados. El objetivo no es transmitir fuerza sino simplemente sujetar de forma segura para poder desplazar dicho objeto. Por esta razón se simplificó el diseño y se siguieron los mismos ejemplos vistos en estudios anteriores [9], [41] para generar un dedo de Fin Ray que cuenta con una forma de triángulo isósceles y con una superficie rugosa menos pronunciada (Figura 4.11).
- **Cuarta versión:** después de varias pruebas de densidades y pequeños cambios en la versión 3 se obtuvo una solución que cumplía con los requisitos principales que debía tener esta pieza: baja resistencia ante deformaciones y estructura sólida para la sujeción. Para ello se redujo, siguiendo las recomendaciones de [10], las paredes de la estructura y el número de membranas internas (Figura 4.12). Posteriormente a las pruebas de la estructura básica,

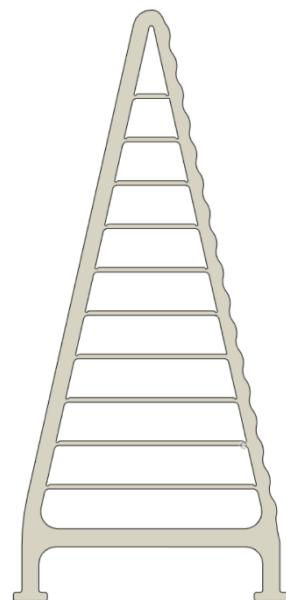


Figura 4.11 Pieza flexible versión 3

se incluyeron las modificaciones que hacen posible la implementación de un FSR junto con este SG. De esta forma se cuenta con un diseño genérico que da la posibilidad de tener 3 niveles de instalación para los FSR y así gestionar la presión ejercida sobre el objeto en tres zonas de contacto (Figura 4.12).

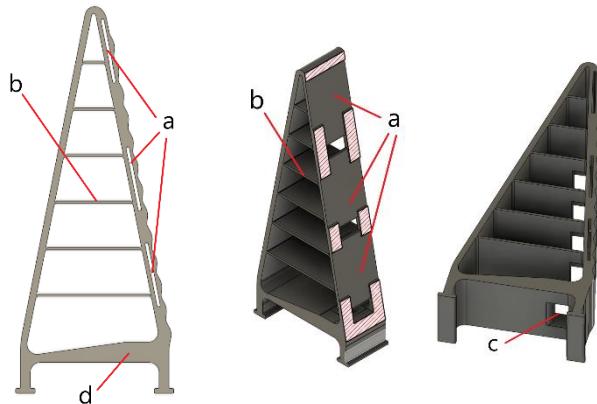


Figura 4.12 Versión definitiva pieza flexible de SG

Se tienen 3 figuras que muestran:

- a) Sitio de alojamiento de FSR
- b) Membrana de estructura
- c) Salida de cables FSR
- d) Refuerzo de estructura

4.1.3. PROTOTIPOS CUERPO SCARA

En este primer acercamiento se tuvo 3 versiones principales que establecieron los requerimientos mecánicos del robot. De esta forma se definieron las longitudes de los eslabones principales y se dio una primera imagen de los problemas a enfrentar para cumplir con el propósito final de sujeción. Estas versiones son las siguientes:

- **Primera versión:** Se define el diseño básico que tendrían estos eslabones y la forma en la que serían instalados en la base del robot (Figura 4.13). Para el primer diseño se establecieron los siguientes aspectos:
 - El eslabón 1 con una longitud de 35 cm y para el segundo una de 25 cm. Es decir, el robot cuenta con un rango de acción de 60 cm.
 - Uso de SM Nema 26 para que realizaran los movimientos en el ESB1 y el ESB2. Se incluye sistema de sujeción con tornillos M4.

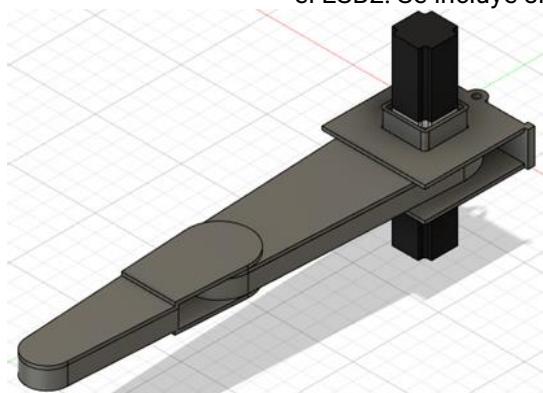


Figura 4.13 Segunda versión de prototipo SCARA

En la imagen se muestran los motores NEMA 26 coaxiales. Las geometrías son muy básicas y conforman la estructura general del diseño.

- **Segunda versión:** el problema inicial que se detectó en la primera versión es que el primer eslabón cuenta con una longitud bastante grande ya que si se toma en cuenta los radios exteriores esta pieza mediría 47 cm. Dicha longitud supera la mayoría de las posibilidades de impresión con tecnología FDM. Además, podría presentar problemas de flexión o incluso ruptura. Por esta razón se incluyeron los siguientes cambios:
 - Material para los eslabones pasa a ser con planchas de metacrilato de 3mm. Se dispusieron una sobre otra para que formarán los primeros eslabones y se tuviera así una estructura rígida (Figura 4.14)
 - Polea de transmisión para correas T2 en sujeción entre eslabón 1 y 2

- Sistema de tensión de correas en eslabón 1 (Figura 4.14)

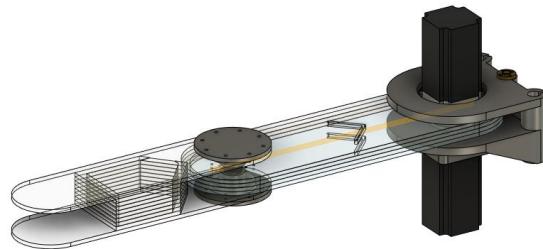


Figura 4.14 Segunda versión de cuerpo SCARA

En la base se puede ver el agujero correspondiente a la instalación de tuerca T8 para tornillo de avance. Además, en el medio del eslabón 1 se puede ver unas guías para tornillos M3 que tienen la función de tensar la correa de transmisión

- **Tercera versión:** era evidente que la solución con metacrilato traía numerosos inconvenientes, sobre todo en el caso del eslabón 2 debido a que éste requiere de una mayor versatilidad en el diseño. Por ello se optó por producirla con impresión FFF ya que sus dimensiones sí son abarcables para esta tecnología y, como no soporta mucha tensión, podría ser impresa en dos partes. Entre otros cambios realizados se encuentra:
 - Inclusión de soporte para rodamientos planos para movimientos de rotación (Figura 4.15)
 - Primeras pruebas de sistema de sujeción con tornillos y tuercas internas M3 (Figura 4.15)
 - Implementación de sistema de poleas de fácil instalación para correas T5 (Figura 4.15)

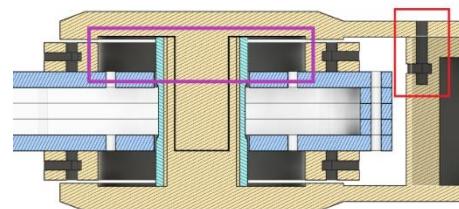
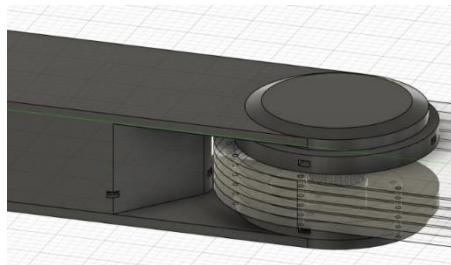


Figura 4.15 ESB2 Versión 2

En rojo se muestra el sistema de sujeción con tornillos M3 y ranura interna para tuerca. En morado se refleja el espacio para rodamiento plano de diámetro externo 68 mm e interno 40 mm. Se ve internamente el sistema de instalación por ajuste de la polea para correa T2.

- **Cuarta versión:** se tenía un diseño claro en cuanto a los primeros eslabones. Sin embargo, era notable el problema que surgiría con el desplazamiento del eje Z. Un tornillo de avance no contaría con la suficiente estabilidad para poder soportar todo el robot y además realizar los correspondientes movimientos. Por ende, se realizaron los siguientes cambios:
 - Sistema de desplazamiento en eje Z con rieles lineales Hiwin HGW15CC¹ (Figura 4.16)
 - Refuerzo en la base de soporte ya que esta pieza debe aguantar toda la torsión del robot (Figura 4.16)
 - Cambio de poleas a GT5 para contar con mayor capacidad de tensión y transmisión de fuerza (Figura A3. 7).

¹ www.hiwin.com

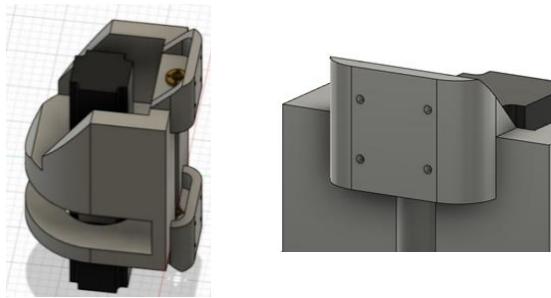


Figura 4.16 Cuarta versión cuerpo SCARA

A la izquierda se puede ver el refuerzo que se incluyó en la base junto con el sistema de sujeción para los carros del riel HIWI que se ve a la derecha.

- **Quinta versión:** el problema principal que se quería resolver en esta versión fue la que existía con el posicionamiento de los motores. Al ser coaxiales requerían de un gran espacio para su instalación. Es por ello por lo que se hicieron los siguientes cambios:
 - o Transmisión de fuerza con sistema de poleas en la base para ambos motores. De esta forma se gana un mayor control de los movimientos y también se puede ordenar los motores en ejes paralelos. Así, el centro de masa pasa a estar concentrado en la base (Figura 4.19)
 - o Cambio de motores a NEMA 23 ya que los requerimientos del robot no necesitaban una mayor potencia
 - o El Segundo eslabón pasa a tener 2 piezas que lo conforman para facilitar su fabricación. También cuenta con inclusión de soporte para el tercer eslabón y los ajustes correspondientes para los motores
 - o En el extremo del segundo eslabón se agrega motor NEMA 11 para desplazamiento en eje z. Se conforma así el tercer eslabón (ESB3) en el extremo del robot (Figura 4.17). Este cuenta con:
 - 2 ejes lineales guiados por rodamientos IGUS¹
 - Soporte de NEMA 11 no cautivo para SG
 - o Inclusión de camino interno para cableado en eslabón 1 (Figura 4.18)

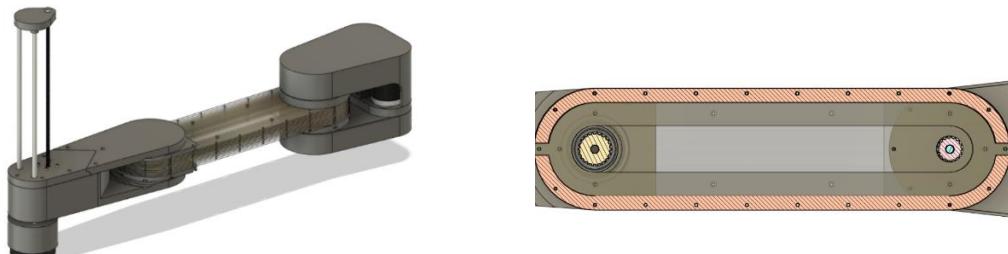


Figura 4.17 Quinta versión cuerpo SCARA

A la izquierda se tiene una vista en perspectiva de esta versión. A la derecha se encuentra un corte medio del ESB1 para entender el sistema interno de dicha pieza.

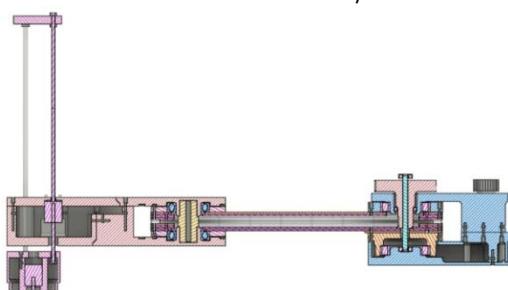


Figura 4.18 Corte lateral de cuerpo SCARA versión 5

Se observa el camino entre los laterales para que el cableado proveniente de los demás eslabones pueda pasar a la base. También se cuenta con agujeros para tornillos M3 en este camino para asegurar dicho cableado

¹ www.igus.es

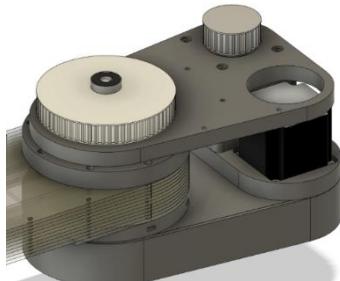


Figura 4.19 Sistema de poleas en base de quinta versión de cuerpo SCARA

En la parte descubierta se ve el sistema de transmisión para correas GT5 de 15 mm de ancho. Este sistema es simétrico en la parte inferior cubierta. El que se observa corresponde a la primera transmisión del eslabón 2.

4.1.4. CUERPO SCARA DEFINITIVO

Al igual que en el diseño del SG la producción del cuerpo del robot, recordando el planteamiento de su modelo como robot SCARA (Figura 3.1), se pueden dividir en 5 módulos:

1. **Base (BS)**: conjunto de piezas que resguardan los motores de los eslabones 1 y 2.
2. **Eslabón 1 (ESB1)**: módulo encargado de realizar movimientos rotativos sobre el eje de la base y con un rango máximo definido. Además, se encarga de transmitir el movimiento del eslabón 2 con un sistema de poleas y correas.
3. **Eslabón 2 (ESB2)**: se encarga de realizar movimiento de rotación sobre el eje final del primer eslabón y con rangos definidos límite. También cuenta con un sistema de sujeción del eslabón 3.
4. **Eslabón 3 (ESB3)**: última parte del cuerpo principal. Se encarga de realizar movimientos lineales con un rango establecido por diseño. Cuenta para ello con sistemas de guía y tuercas T8 para tornillo de avance. Su principal función es la de móvilizar el eje Z y de sujetar el SG para su uso.
5. **PCBOX**: en ella se encuentra la PCB principal unida a BS.

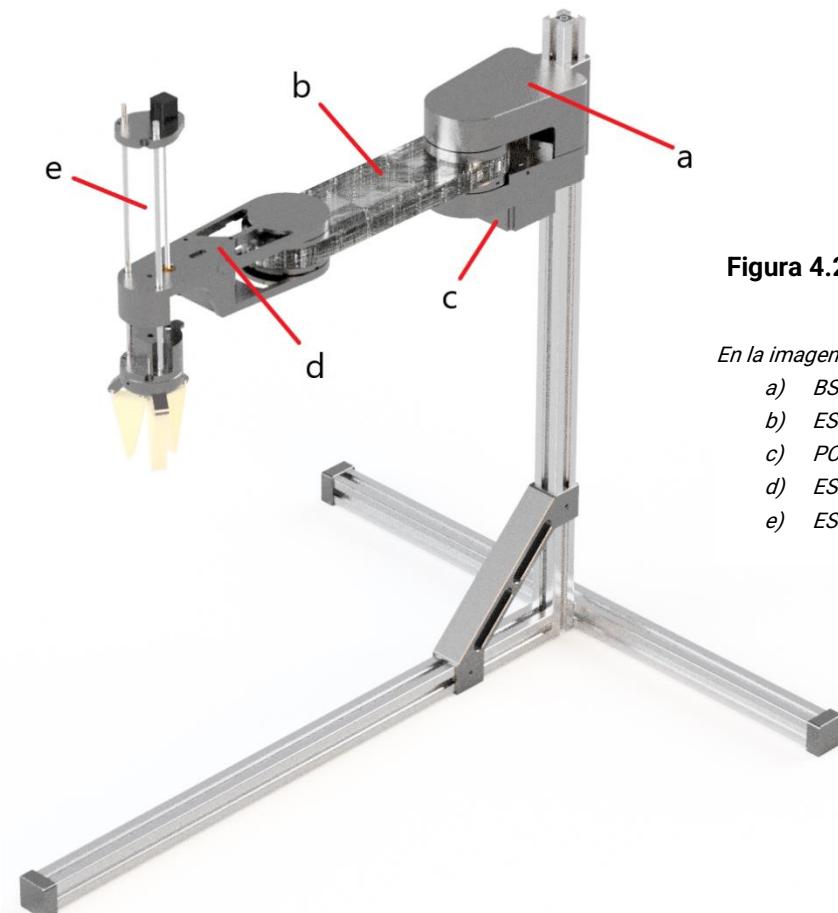


Figura 4.20 Diseño final montado SCARA

En la imagen se muestra:

- a) BS
- b) ESB1
- c) PCBOX
- d) ESB2
- e) ESB3

Después de numerosos intentos se ha conseguido un diseño base para poder trabajar (Figura 4.17). A partir de este se llega a lo que se llamará diseño final que se ha probado como robot de pick and place (Figura 4.20). Por esta razón la explicación del diseño del cuerpo irá enfocada en ver las partes que conforman cada uno de estos módulos que, a diferencia de capítulos anteriores, se verán más detalladamente. Para ello se estudiarán las partes diseñadas en base a los requerimientos mecánicos que se han establecido en las fases finales del diseño inicial (4.1.3) y las últimas soluciones adoptadas. En cuanto a los componentes mecánicos se ha llegado a las siguientes conclusiones:

- La aplicación requiere de articulaciones que soporten trabajos de compresión y de corte, por ello los rodamientos planos no son los más indicados para este propósito [42]. Por esta razón se opta por rodamientos de bolas rígidos [43] que prestan un buen funcionamiento ante fuerzas axiales y de corte. Además, suelen tener un menor peso que las de rodamiento plano [43] y esto es una gran ventaja para reducir el peso del sistema
- Correa de transmisión de 5 mm de paso y un ancho de 15 mm [44]. De esta forma se evitan posibles rupturas por las altas tensiones que pueden existir en el sistema de poleas.
- Uso de motores NEMA 23 para los primeros eslabones, NEMA 11 para el tercer eslabón y NEMA 11 no cautivo para SG (4.3.2.5).
- Para las posibles ensamblajes que se tengan se hará uso del sistema de atornillado interno visto anteriormente (Figura 4.18). De esta forma cada uno de los módulos del cuerpo es un bloque que puede ser ensamblado de forma separada del resto.

4.1.4.1. MÓDULO DE BASE (BS)

Este módulo necesita tener un soporte rígido para poder sostener todo el peso del robot y la carga (peso del robot en el rango de 6 Kg). Por esta razón cuenta con un conjunto de partes que tienen la principal funcionalidad de construir un cuerpo sólido que sea fácil de ensamblar y que tenga la menor cantidad de material necesaria para su fabricación (Figura 4.21).

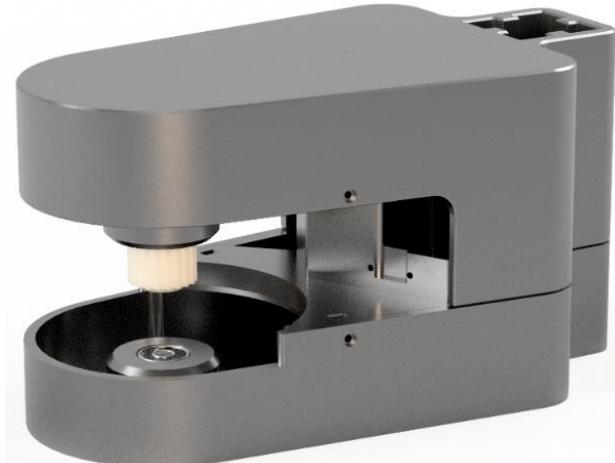


Figura 4.21 Base de cuerpo final SCARA

Este módulo se divide en 3 partes principales:

- **Parte superior (BS_TOP)**: conformadas por 3 piezas que cumplen con la función de hacer la sujeción del motor encargado de mover el segundo eslabón. Estas piezas son:
 - o BS_TOP_MOT2: esta pieza se encarga de la sujeción del motor 2 y del soporte del rodamiento 6805-2RS [43] para polea de transmisión (Figura 4.23).
 - o BS_TOP_COV: pieza de cubrimiento externo superior (Figura 4.24). Cuenta con sujeción a perfil XCBM 1X44 [45].
 - o BS_TOP_TENS: Sistema de tensión de correa con barra de 8 mm y rodamiento 608-2RS [43] (Figura 4.22).

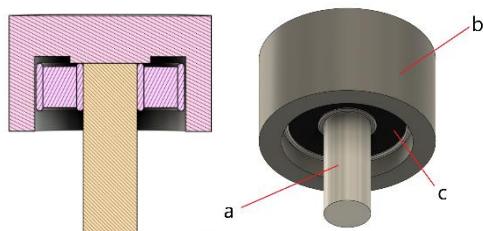


Figura 4.22 BS_BOT_TENS y BS_TOP_TENS

A la izquierda se ve un corte del mecanismo. A la derecha se tiene:

- a) Barra sólida de aluminio de 8 mm y 12 cm de largo
- b) Cobertura de tensor adaptada para rodamiento
- c) Rodamiento 608-2RS

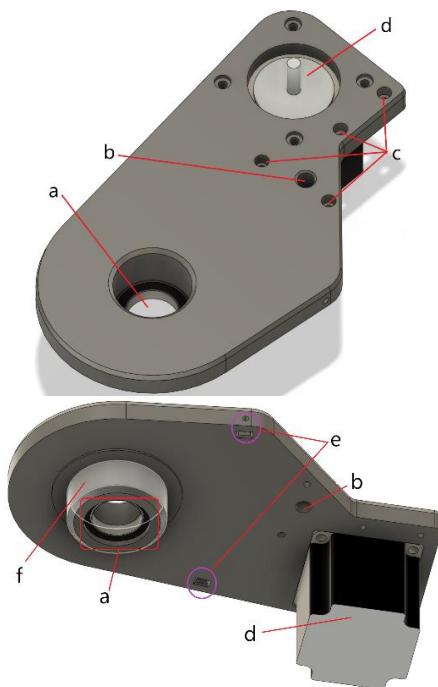


Figura 4.23 BS_TOP_MOT2

- a) Soporte de rodamiento 6805-2RS
- b) Entrada de sistema de tensión para barra de 8mm
- c) Entrada de tornillos M3 para sujeción con parte inferior de base
- d) Sujeción de motor Nema 23 con tornillos M4
- e) Sujeción con pieza de recubrimiento superior de base para tornillos M3
- f) Entrada de rodamientos 6010-2RS

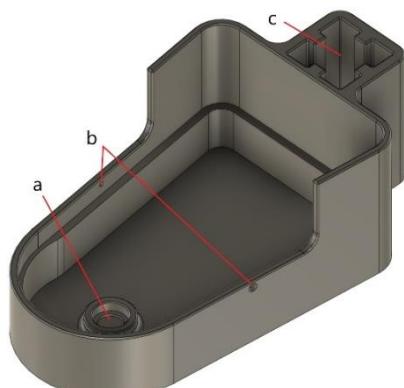


Figura 4.24 BS_TOP_COV

- a) Agujero para instalación de rodamiento 608-2RS para sujeción de barra de BS_PULL
- b) Entrada para perfil de aluminio de 44mm x 44mm y ranura de 10 mm
- c) Entrada de tornillos M3 para fijación con pieza de sujeción de motor 2

- **Parte Inferior (BS_BOT):** conformada por 3 piezas encargadas de dar soporte al motor del eslabón 1 y servir de punto principal de apoyo del peso del robot. Se tiene:
 - o BS_BOT_COV: pieza de soporte de eslabón 1 y cubrimiento inferior externo de base (Figura 4.26)
 - o BS_BOT_MOT1: pieza de sujeción de motor 1 (Figura 4.25)
 - o BS_BOT_TENS: sistema de tensión de correa con barra de 8 mm y rodamiento 608-2RS [43] (Figura 4.22)

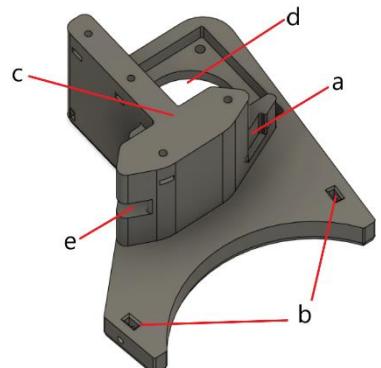


Figura 4.25 BS_BOT_MOT1

- Sistema de sujeción de LS para eslabón 1 con tornillos M3
- Entradas de tornillos M3 para unión con cubrimiento inferior de base
- Entradas de tornillos M3 para unión con pieza de sujeción de motor 2 de base
- Sujeción de motor 1 con entradas de tornillos M4

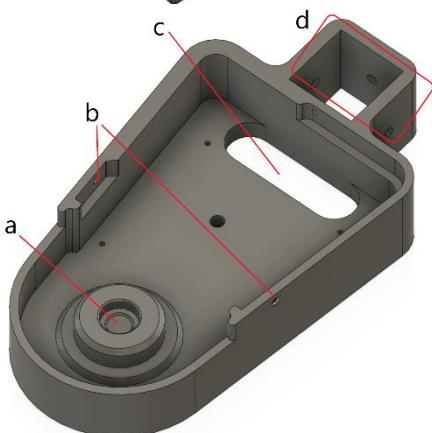


Figura 4.26 BS_BOT_COV

- Sujeción para rodamiento de bolas con Φ interno de 8 mm y externo de 25 mm
- Entradas de tornillos M3 para unión BS_BOT_M1.
- Salida de cableado a la caja de la PCB
- Entrada para perfil de aluminio de 44 mm x 44 mm con agujero M6 para sujeción con tuercas planas

- **Sistema de poleas (BS_PULL):** está formada por 3 piezas (Figura 4.28). Existen 2 que se ensamblan por presión en la parte interna de la base (Figura 4.29). Su objetivo es transmitir la fuerza desde el motor 2 hasta el segundo eslabón. Para ello se cuenta con:
 - o BS_PULL_P62: polea GT5 de 62 dientes elevadora de transmisión inicial
 - o BS_PULL_P20: polea GT5 de 20 dientes coaxial a BS_PULL_P62. Reduce la relación de transmisión al segundo eslabón
 - o BS_PULL_P27: polea de sujeción de motores GT5 de 27 dientes (Figura 4.27).

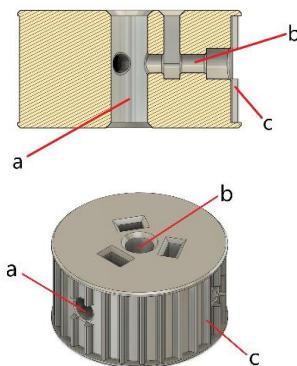


Figura 4.27 BS_PULL_P27

En la imagen de arriba se muestra un corte transversal de la pieza donde se ve el lugar de asentamiento del eje del motor de Φ 6 mm y un largo de 21mm. Se muestra:

- Entrada de eje de motor
- Sistema de sujeción con tornillos M3
- Polea T5 para correa de máximo 17 mm de ancho

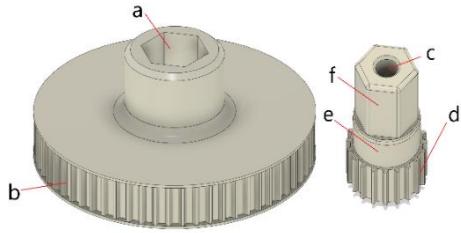


Figura 4.28 BS_PULL_P62 y BS_PULL_P20

- a) Ajuste hexagonal para conexión con BS_PULL_P20.
- b) Polea GT5 de 62 dientes
- c) Canal para barra de aluminio sólida de Ø 8mm de soporte de poleas
- d) Polea GT5 de 20 dientes
- e) Encaje con rodamiento de 25mm Ø
- f) Ajuste hexagonal para conexión con BS_PULL_P62

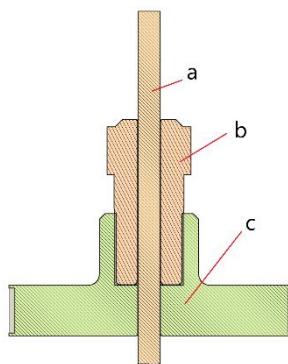


Figura 4.29 Corte transversal en sistema de poleas de base SCARA

- a) Barra de aluminio sólida de 12,5 cm.
- b) BS_PULL_P20
- c) BS_PULL_P62

4.1.4.2. MÓDULO DE ESLABÓN 1 (ESB1)

Con la base montada (Figura 4.21) se cuenta con el soporte y sistemas de transmisión necesarios para instalar el primer eslabón (Figura 4.30).

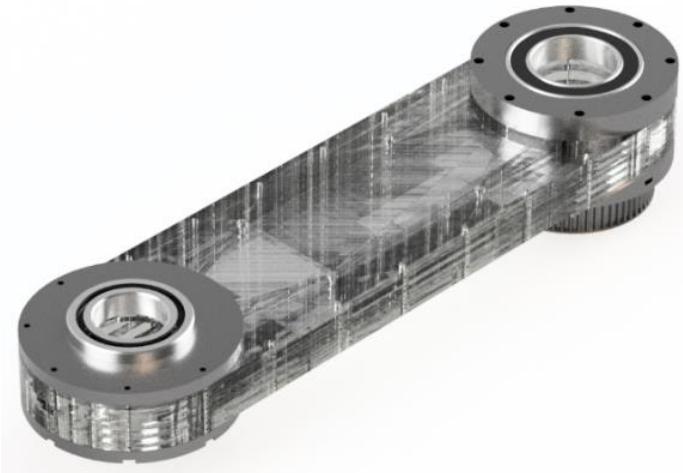


Figura 4.30 Eslabón 1 definitivo para cuerpo SCARA

Para estudiar el eslabón 1 se separará este módulo en 3 partes principales:

- **Sujeción con base (ESB1_SJBS):** esta parte está conformada por 2 rodamientos de bolas 6010-2RS [43] que están sujetas por dos piezas que cuentan con la sujeción de tornillos M3 de al menos 50mm de longitud. Estas piezas son:
 - o ESB1_SJBS_TOP: pieza superior encargada de la sujeción de rodamiento de bolas y conexión con parte superior de base (Figura 4.32)
 - o ESB1_SJBS_PUL62: polea inferior GT5 que cuenta con 62 dientes para transmisión de fuerza a eslabón 1 y soporte de peso con rodamiento de bolas (Figura 4.31)

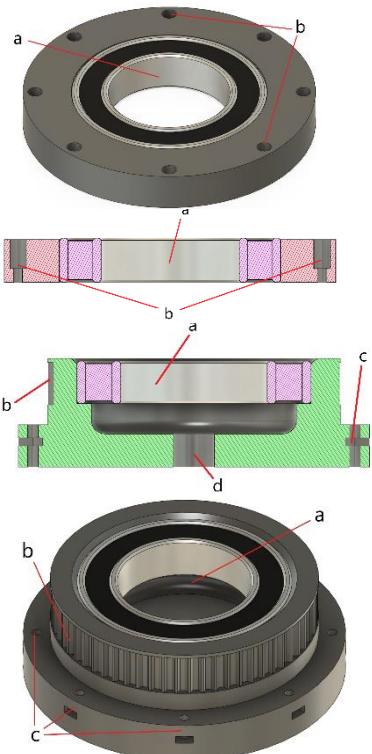


Figura 4.32 ESB1_SJBS_TOP

En la imagen superior se tiene una vista en perspectiva de la pieza. En la inferior se detalla su interior con un corte transversal. Se tiene:

- a) Agujero de sujeción de rodamiento 6010-2RS
- b) Agujeros de entrada para tornillos M3

Figura 4.31 ESB1_SJBS_PUL62

En la imagen superior se tiene una vista en perspectiva de la pieza. En la inferior se detalla su interior con un corte transversal. Se tiene:

- a) Agujero de sujeción de rodamiento 6010-2RS
- b) Polea GT5 de 62 dientes para correas de un ancho máximo de 17 mm
- c) Sistema de sujeción de tornillo M3 con tuercas internas
- d) Entrada de barra sólida de aluminio para permitir sujeción con

- **Sujección con ESB2 (ESB1_SJ2):** para hacer la unión entre el eslabón 1 y 2 se siguió un diseño parecido para la sujeción del eslabón 1 con la base. En este caso se necesitan rodamientos de menor tamaño ya que esta articulación no soportará tanto peso como la primera. Los rodamientos usados son los 6008-2RS [43]. Estos se sujetan con las dos piezas siguientes:

- o ESB1_SJ2_TOP: es la pieza superior de sujeción que se encarga de soportar el rodamiento y cuenta con entradas de tornillos M3 de 45 mm (Figura 4.34).
- o ESB1_SJ2_BOT: pieza de sujeción inferior encargada de soportar el rodamiento inferior y cuenta con salida de tornillos y soporte de tuercas (Figura 4.33).

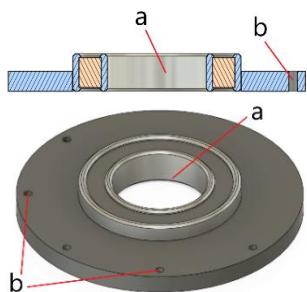


Figura 4.33 ESB1_SJ2_TOP

En la imagen superior se muestra la vista en perspectiva de la pieza. En la inferior se encuentra un corte transversal que servirá de ayuda para ver los puntos más relevantes de este diseño. Se tiene:

- a) Soporte de rodamiento 6008-2RS
- b) Entrada para tornillos M3 de 45mm

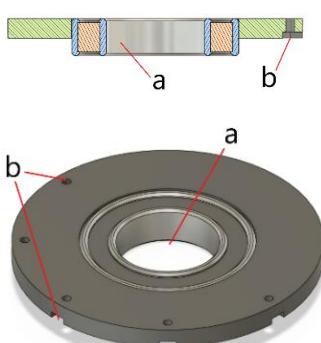


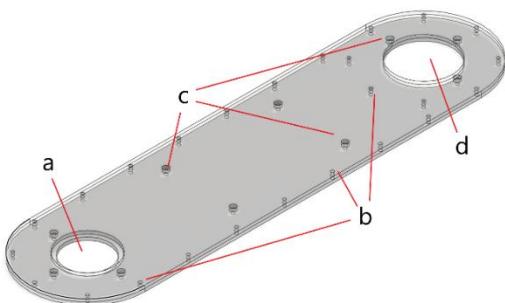
Figura 4.34 ESB1_SJ2_BOT

En la imagen superior se muestra la vista en perspectiva de la pieza. En la inferior se puede ver un corte transversal que servirá de ayuda para ver los puntos más relevantes de este diseño. Se tiene:

- a) Soporte de rodamiento 6008-2RS
- b) Salida para tornillos M3 con soporte de tuerca

- **Parte principal de módulo (ESB1_PRC):** esta pieza es la encargada de realizar los movimientos del primer eslabón, de crear el camino para la transmisión de la fuerza al segundo eslabón y de organizar el cableado proveniente de las demás partes del robot (sensores y actuadores, ver 4.2). Está formado por:
 - o ESB1_PRC_TOP: pieza superior compuesta por dos láminas de metacrilato para poder acceder a su interior (Figura 4.35).
 - o ESB1_PRC_BOT: pieza base compuesta por 8 capas de metacrilato para poder resistir las altas tensiones (Figura 4.36).

Figura 4.35 ESB1_PRC_TOP



En la imagen se muestra la pieza en perspectiva y en la inferior la pieza en vista superior. Se tiene:

- Soporte de rodamiento para parte de sujeción con ESB2
- Entradas de tornillos 3M para soporte de todo el módulo.
- Entradas de tornillos M3 con sujeción efectiva en ESB1_PRC_BOT. Estos tornillos aseguran el correcto camino del cableado interno.

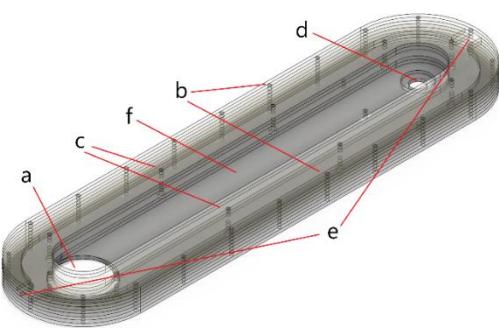


Figura 4.36 ESB1_PRC_BOT

Se muestra la vista en perspectiva de la pieza. Se tiene:

- Soporte de rodamiento para parte de sujeción con ESB2
- Entradas de tornillos 3M para soporte de todo el módulo
- Entradas de tornillos M3 con sujeción efectiva en ESB1_PRC_BOT. Estos tornillos aseguran el correcto camino del cableado interno
- Salida de barra sólida de 8 mm
- Entrada y salida de cableado

4.1.4.3. MÓDULO DE ESLABÓN 2 (ESB2)

Este módulo conforma el segundo eslabón del robot SCARA. Para su funcionamiento cuenta con las siguientes características básicas:

- Polea final de sistema de transmisión.
- Sistema de sujeción para desplazamiento líneas de módulo de ESB3.
- Sujeción para LS correspondientes a ESB2 y ESB3.
- Aberturas para cableado del ESB2 y ESB3.
- Conexión con ESB1.

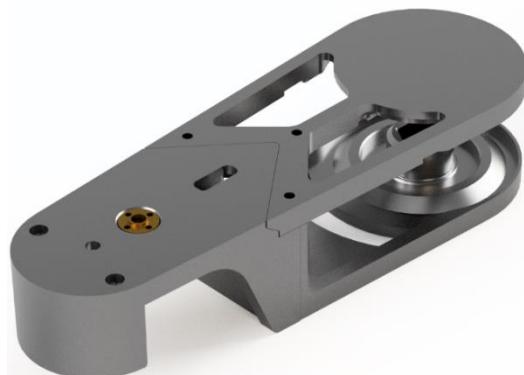


Figura 4.37 Modulo ESB2 definitivo

En este módulo (Figura 4.37) hay 2 partes principales:

- **Pieza de sujeción con ESB3 (ESB2_SJ3):** esta es una pieza única (Figura 4.38) que cuenta con las siguientes características:
 - o Soporte de LS para ESB3
 - o Soporte de rosca T8 de 4 hilos para tornillo de desplazamiento de ESB3
 - o Sistema de sujeción de guías lineales IGUS
 - o Sistema de sujeción con ESB2_SJ1
 - o Entradas y salidas de cableado

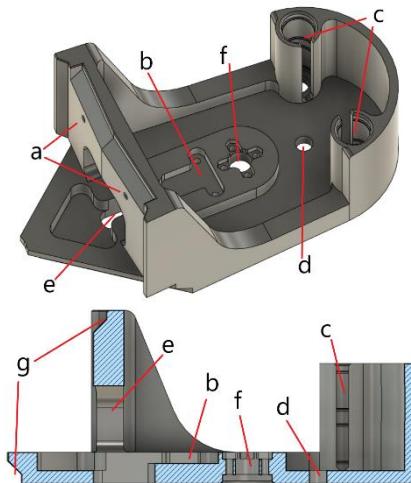


Figura 4.38 ESB2_SJ3

En la imagen superior se muestra la vista en perspectiva de la pieza. En la inferior se puede ver un corte transversal que servirá para ver los puntos más relevantes de este diseño. Se tiene:

- a) Entrada de tornillos M3 de 40 mm para ensamblaje con ESB2_SJ1_BOT
- b) Ranura de soporte de LS para ESB3
- c) Soporte de guías lineales IGUS
- d) Agujero de entrada de tornillo de avance de SG
- e) Entrada y salida de cableado
- f) Soporte de rosca para tornillo de avance T8
- g) Conexión con ESB2_SJ1

- **Sistema de sujeción con ESB1 (ESB2_SJ1):** Esta parte del módulo está formado por 3 piezas:
 - o ESB2_SJ1_PUL21: polea GT5 de 21 dientes que cuenta con una estructura simétrica que tiene a sus extremos un ajuste hexagonal para fácil instalación (Figura 4.39).
 - o ESB2_SJ1_TOP: tapa superior de sujeción de rodamiento que se encarga de soportar la tensión superior del eslabón que es transmitida por ESB2_SJ1_PUL21. También sujetta el rodamiento superior y cuenta con 3 agujeros de fijación para tornillos M3 de 25 mm de largo (Figura 4.40).
 - o ESB2_SJ1_BOT: base de sujeción de rodamiento. Se encarga de transmitir la fuerza que recibe ESB2_SJ1_PUL21 hacia la parte inferior del módulo. Cuenta con agujeros de recepción de tornillos M3 y con las correspondientes entradas de tuerca para sujeción de ESB2_SJ1_TOP (Figura 4.41).

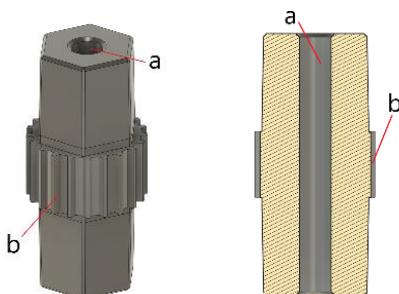


Figura 4.39 ESB2_SJ1_PUL21

A la izquierda se tiene una vista en perspectiva de la pieza. A la derecha un corte transversal de la misma. Se resalta:

- a) Entrada de barra sólida de 8mm para refuerzo de polea
- b) Sistema de poleas GT5 con anchura máxima de 17 mm

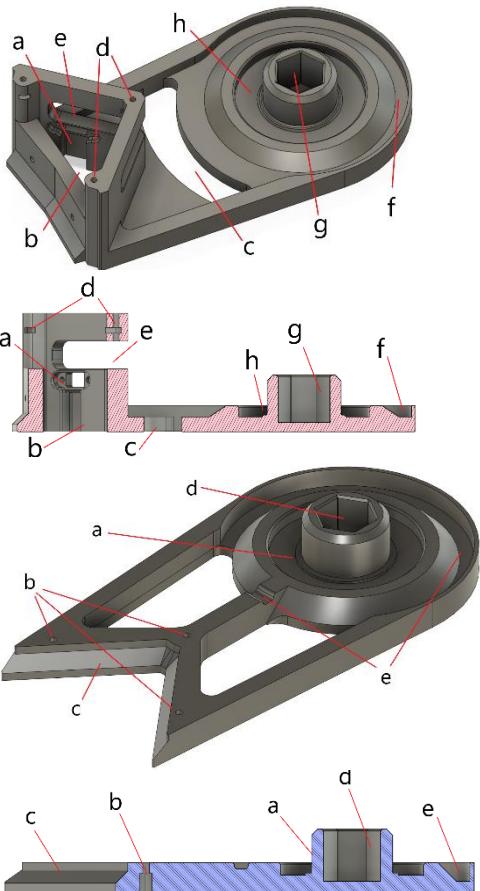


Figura 4.41 ESB2_SJ1_BOT

En la imagen superior se muestra la vista en perspectiva de la pieza. En la inferior se puede ver un corte transversal de la pieza que será de ayuda para ver los puntos más relevantes de este diseño. Se tiene:

- a) Sujeción de LS para ESB2_SJ1_TOP
- b) Salida de cables
- c) Entrada de cableado
- d) Entrada de tornillos M3 para ensamblaje con ESB1
- e) Salida de cableado
- f) Espacio de libre movimiento de extremo de tornillos
- g) Entrada hexagonal para ESB2_SJ1_PUL21

Figura 4.40 ESB2_SJ1_TOP

En la imagen superior se muestra la vista en perspectiva de la pieza. En la inferior se encuentra un corte transversal que servirá de ayuda para ver los puntos más relevantes de este diseño. Se tiene:

- a) Entrada de rodamiento ESB1_SJ2_TOP
- b) Entrada de tornillos M3 para ensamblaje con ESB2_SJ1_BOT
- c) Ajuste de conexión con ESB2_SJ3
- d) Entrada de ESB2_SJ1_POL20 con ajuste hexagonal
- e) Espacio libre para cabezas de tornillos

4.1.4.4. MÓDULO DE ESLABÓN 3 (ESB3)

Para tener el cuerpo completo del SCARA queda estudiar uno de los eslabones más importantes. Este va a ser el módulo denominado ESB3 (Figura 4.42) que cuenta con la finalidad de ejecutar los desplazamientos en el eje Z del extremo del robot. Para cumplir con los requisitos de funcionamiento ESB3 debe contar con las siguientes características principales:

- **Desplazamiento lineal:** cuenta con soporte de motor para efectuar desplazamiento en su rango máximo de 28,5 cm provistos por el tornillo de avance de diámetro de 8 mm (tiene 4 hilos de 2 mm de avance). Finalmente, este tornillo se desplaza a través de la rosca T8 presente en ESB2_SJ3.
- **Soporte de motor para SG:** a través de este motor se desplazará linealmente un tornillo de avance que aporta la fuerza al SG.
- **Sistema de alineación y desplazamiento:** cuenta con dos barras huecas de aluminio que guían el movimiento teniendo la menor fricción y peso posible.

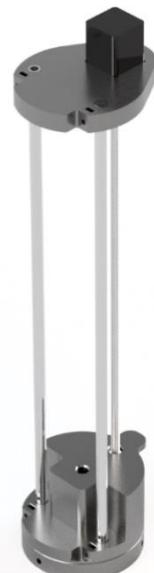


Figura 4.42 Módulo ESB3

- **Ligero:** tanto piezas impresas como mecánicas deben tener el menor peso posible que permita facilitar el desplazamiento
- **Ensamblaje sencillo:** sistema de sujeción con tornillos M3 y M4 para distintos propósitos de ensamblaje como son del propio módulo y uniones con SG.

El módulo cuenta con 2 partes unidas a las barras huecas de aluminio. Ambas piezas también tienen los requisitos mínimos de diseño para que sean funcionales y requieran de poco material para su fabricación. Estas partes son:

- **ESB3_MOT3:** pieza (Figura 4.44) que por sus funciones cuenta con:
 - o Soporte y sujeción con tornillos M2.5 de motor 3 (NEMA 11) para movimientos lineales
 - o Sistema de sujeción de barras huecas de aluminio de Φ 8mm con tornillos 3M
- **ESB3_MGRP:** pieza (Figura 4.43) inferior que cuenta con:
 - o Base para sostener el motor 4 (NEMA 11 no cautivo) con tornillos M2.5
 - o Sistema de sujeción de barras huecas de aluminio de Φ 8mm con tornillos 3M
 - o Sistema de Sujeción con GRPR con tornillos M4

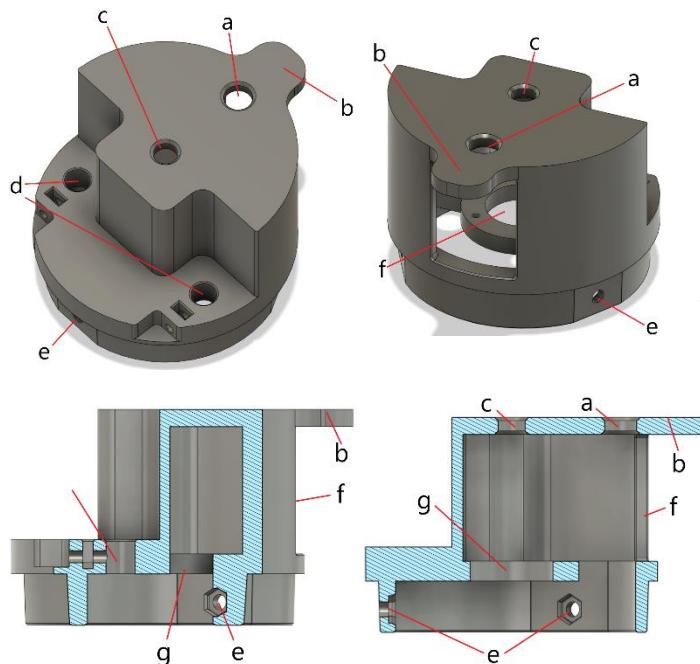


Figura 4.43 ESB3_MGRP

En las imágenes superiores se muestran dos vistas en perspectiva de la pieza: a la derecha una frontal y a la izquierda una de la parte posterior. Debajo se tienen 2 imágenes de cortes transversales de la misma pieza en planos de interés. En ellas se muestra:

- a) Entrada de tornillo de avance T8.
- b) Superficie de contacto con LS para ESB3
- c) Salida de tornillo de avance para SG
- d) Sistema de sujeción para barra de Φ 8mm con tornillos M3
- e) Sistema de sujeción con SG con tornillos M4 y tuercas internas
- f) Entrada y salida de cableado
- g) Base de soporte para Motor 4

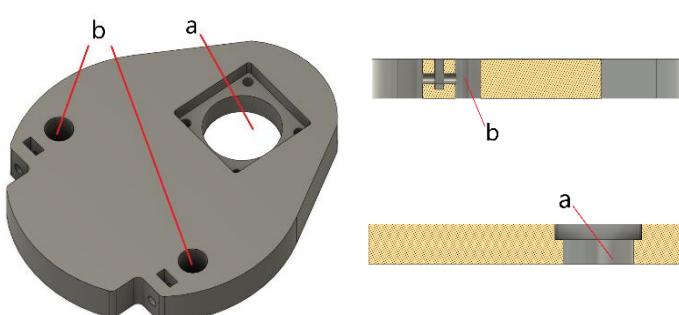


Figura 4.44 ESB3_MOT3

En la imagen de la derecha se muestra la vista en perspectiva de la pieza. En las de la izquierda se muestran dos cortes transversales que servirán para ver los puntos más relevantes de este diseño. Se tiene:

- a) Soporte para motor 3 con entrada de tornillos M2.5
- b) Sistema de sujeción para barra de Φ 8mm con tornillos M3

4.1.4.5. CAJA DE PCB (PCBOX)

Este módulo se encarga de sostener la PCB que se usó para controlar todos los motores y periféricos (4.3.3). Es un diseño simple que cuenta con la finalidad de:

- Sostener firmemente la PCB con 4 tornillos M3
- Tener sistema de sujeción con BS
- Contar con salida de pantalla OLED (4.3.2.4)
- Contar con salida y entrada de cableado y conexión USB para alimentación de placa

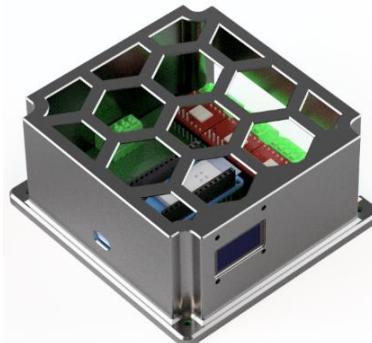


Figura 4.45 PCBOX

Con estas especificaciones se diseñó un módulo (Figura 4.45) que cuenta con 2 piezas:

- **PCBOX_TOP**: tapa de cobertura de PCB. Cuenta con sujeción a PCBOX_TOP con 4 tornillos M3.
- **PCBOX_BOT**: pieza base de PCB (Figura 4.47) que permite dejar una separación entre las soldaduras y BS_BOT_COV. También cuenta con fijaciones a la base con tornillos M3.

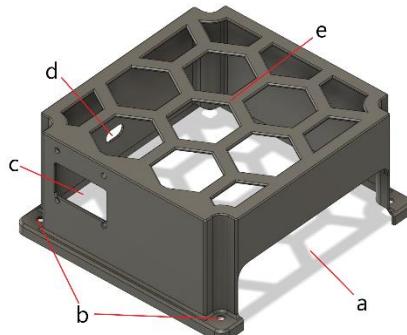


Figura 4.46 PCBOX_TOP

Vista en perspectiva de pieza. Se resalta:

- a) Entrada y salida de cableado
- b) Entradas de tornillos M3 para ensamblaje con PCBOX_BOT
- c) Salida de pantalla OLED con entradas para tornillos M2
- d) Salida de puerto USB tipo C
- e) Patrón hexagonal para ventilación y fácil acceso a PCB

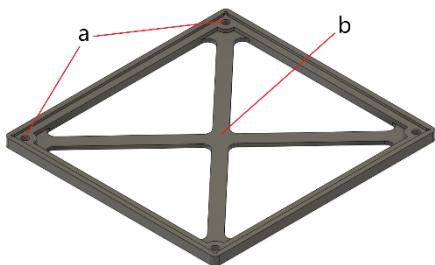


Figura 4.47 PCB_BOT

Vista en perspectiva de pieza. Se tiene:

- a) Entrada de tornillos M3 para sujeción con BS_BOT_COV
- b) Membranas de base para rigidez y con espaciado para soldaduras

4.2. PRODUCCIÓN DE PIEZAS

Para la fabricación de las piezas del modelo definitivo (4.1.4 y 4.1.2) se debe primero entender que existen distintos requerimientos en cada una de ellas. Por esta razón van a separarse según los requerimientos que tenga. Se dividen en:

- Piezas rígidas que necesitan tener una buena resistencia a altas tensiones cortantes. En el robot existen:
 - o BS
 - o EBS2
 - o ESB1
- Piezas rígidas de cobertura que no requieren soportar tensiones. En el modelo se pueden encontrar:
 - o PCBOX
- Piezas rígidas y ligeras que no soportarán tensiones cortantes. Entre ellas:
 - o ESB3
 - o SG_BS
- Piezas flexibles con una estructura estable y con baja resistencia ante fuerza de compresión:
 - o SG_FLX

Una vez categorizada cada una de las piezas según sus requerimientos mecánicos es posible estudiar el proceso de fabricación escogido. Para ello se verán las tecnologías implementadas y con ello los materiales y SW utilizados en cada caso.

4.2.1. FABRICACIÓN CON TECNOLOGÍA FDM

En el diseño se ha visto (4.1) que se necesitan numerosas piezas que cuentan con geometrías compleja. Por esta razón se optó por el uso de la tecnología FFF o FDM [8], [46] ya que es posible crear modelos impresos con numerosos tipos de materiales y con distintas propiedades. Además, los resultados son bastante buenos incluso con geometrías complicadas (por ejemplo, Figura 4.43) que, con cualquier otro método de fabricación, serían difíciles de lograr. Por otro lado, es un tipo de tecnología que permite múltiples configuraciones de fabricación como pueden ser: densidad, rigidez de las paredes y resolución de capas de impresión (Tabla 4.2).

4.2.1.1. IMPRESORAS UTILIZADAS

Para la producción de las piezas se han utilizado dos impresoras de Creality¹ para acelerar el proceso de fabricación. Estas son:

- **Ender 3D Pro:** con esta impresora se realizaron impresiones de las piezas más pequeñas. Esto se debe a que cuenta con una dimensión reducida de impresión y además, cuenta con una tecnología de impresión que no permite velocidades altas (Tabla 4.1). Es decir, que necesita mayor tiempo de trabajo que impresoras con tecnologías más avanzadas. Entre las piezas que se imprimieron están: ESB3_MGRP, ESB3_MOT3, ESB2_SJ1_PUL21, ESB1_SJ2_BOT, ESB1_SJ2_TOP,

¹ www.creality.com

- ESB1_SJBS_TOP, BS_PULL_P27, BS_PULL_P62, BS_PULL_P20, PCBOX, BS_BOT_TENS y BS_TOP_TENS.
- **CR10S-Pro V2:** con esta segunda impresora se realizaron impresiones de un mayor volumen gracias a sus características (Tabla 4.1). Además, de su gran capacidad volumétrica tiene una mayor velocidad de impresión acompañada de un sistema de protección ante ruptura de filamento. Por estas razones se fabricaron las piezas: ESB2_SJ3, ESB2_SJ1_BOT, ESB2_SJ1_TOP, BS_BOT_MOT1, BS_BOT_COV, BS_TOP_COV y BS_TOP_MOT2. Por otro lado, es una impresora con capacidad de imprimir materiales flexibles sin necesidad de modificaciones, por ende, se fabricaron también las piezas SG_FLX.

Tabla 4.1 Comparación entre impresoras 3D utilizadas

	Ender 3D Pro	CR10S-Pro V2
Tamaño de impresión (mm³)	220x220x250	300x300x400
Velocidad de impresión (mm/s)	≤ 80	≤ 120
Precisión de impresión (mm)	0.1	0.1
Temperatura de cama (°C)	≤ 100	
Temperatura de Boquilla	≤ 240	≤ 260
Tipo de extrusor	Bowden ¹ con 1 engranaje	Bowden con 2 engranajes
Filamentos aceptados	PLA, ABS, TPU (con modificación), TPE (con modificación)	PLA, ABS, TPU, TPE, madera, aleaciones de cobre
Tecnología de nivelación	Mecánica	Automatizada con sensor de proximidad.
Protección ante ruptura de filamento	No	Si

4.2.1.2. FLUJO DE TRABAJO Y HERRAMIENTAS USADAS

Para poder hacer uso de las impresoras 3D se debe tener el diseño en un formato que estas máquinas puedan entender para fabricar la pieza. Además, se debe seguir un procedimiento de fabricación que sea acorde con la tecnología FDM. Por ello, se ha seguido el flujo de trabajo propuesto en [47] de tal forma que para cada pieza se han realizado los siguientes pasos:

1. **Fase de boceto:** se genera un esquema básico de la pieza a partir de requerimientos y funcionalidades.
2. **Fase CAD:** Modelización del diseño con SW para tener las dimensiones específicas. Se ha optado por Fusion360 (4.1).
3. **Fase CAM:** Generación de archivo STL² de cada pieza con Fusion360. De esta forma es posible cargar dicho archivo en el SW de *slicing*³, que es el encargado de generar el fichero GCODE que corresponde con el patrón de fabricación que debe seguir la máquina de impresión. En este caso se optó por el SW de Ultimaker⁴ Cura 4.12.1.

¹ Motor de extrusor alejado de parte móvil de extrusión con cable bowden por el cual circula el filamento.

² Formato de estereolitografía.

³ No se ha encontrado una buena traducción para denominar a este tipo de SW. Suelen llamarse slicers.

⁴ www.ultimaker.com

4. **Fase de fabricación:** se carga el GCODE dese CURA y se guarda en una SD para luego ser leída por máquina a utilizar. En esta fase se deben seguir los pasos recomendados por el proveedor de material de impresión para tener la mejor calidad posible en el acabado final.
5. **Fase de post-procesado:** se separa la pieza de la base de impresión y se eliminan los soportes de esta en caso de existir. Algunas piezas requieren de mecanizados posteriores para mejorar acabados, normalmente lijado para suavizar extremos y mejorar ajustes.

4.2.1.3. PARÁMETROS DE IMPRESIÓN

En este apartado se hará énfasis en las fases de CAM y de fabricación (4.2.1.2) ya que, teniendo un diseño completo, son las partes más cruciales de la fabricación y dependen, como ya se verá, de los requerimientos de la pieza y del material que se vaya a utilizar. Esto es, principalmente, porque las piezas impresas por FDM presentan anisotropía por las propias características de la tecnología [46]. Por esta razón, se separarán las configuraciones que fueron necesarias en el SW Cura y en la impresora de acuerdo con el tipo de pieza y material que se necesita para su fabricación (pág. 66). Se utilizaron los siguientes filamentos:

- **Piezas impresas con PLA:** para tener las configuraciones de los 3 primeros tipos de piezas a imprimir se ha optado por el PLA. Esto se debe a que es uno de los polímeros más populares en el mundo de las impresoras 3D por su bajo costo y buenos resultados de fabricación [46]. Además, no requiere de temperaturas muy elevadas para la fusión como otros materiales (por ejemplo, ABS). Se siguieron las recomendaciones presentadas en [8], [46] para la configuración en cada caso (Tabla 4.2).

Tabla 4.2 parámetros impresión piezas con PLA

Parámetro	Piezas rígidas con resistencia a tensiones	Piezas rígidas de cobertura	Piezas rígidas y ligeras
Densidad (%)	40	20	30
Altura de capa (mm)	0.2	0.2	0.2
Grosor de pared (mm)	2	0.8	1.6
Velocidad de impresión (mm/s)	80	100	80

- **Piezas impresas con TPU:** en los inicios del diseño del SG_FLX se utilizó TPE con una dureza de A93. Sin embargo, el TPE no tenía soluciones de filamento con menor dureza y tampoco presentaba características técnicas adecuadas para el funcionamiento que se busca en el SG. Por esta razón se optó por el uso de TPU ya que, principalmente, este polímero cuenta con mejores propiedades técnicas: mayor resistencia a la abrasión, mayor rango de tipo de dureza, mayor rigidez, entre otros. En concreto, a partir de los estudios de [10], se usó eFil TPU 85A [48] debido a que es uno de los filamentos

más suaves para impresión FDM que existen en el mercado actualmente. A partir de estos estudios, las recomendaciones del proveedor de filamento y numerosas pruebas se llegó a la configuración perfecta para imprimir con TPU piezas flexibles (Tabla 4.3).

Tabla 4.3 Parámetros de impresión con TPU A85¹

Parámetro	Piezas flexibles con una estructura estable
Densidad (%)	5
Altura de capa (mm)	0.2
Grosor de pared (mm)	0.8
Velocidad de impresión (mm/s)	20
Flujo de impresión (%)	115

Cabe destacar que hubo varios problemas a la hora de imprimir debido a que al comienzo se utilizó la impresora Ender 3D. Sin embargo, esta máquina requiere de algunas modificaciones en la tensión del extrusor lo cual trae numerosos problemas típicos de imprimir con filamentos flexibles: obstrucción en extrusor y bloqueo de filamento por alta tensión de extrusión (Figura A3. 12). Con la CR10S se solucionaron dichos problemas iniciales, pero existió el inconveniente de tener un flujo de filamento mucho menor del requerido (Figura A3. 11). Después de varios intentos y pruebas de flujo se consiguió el parámetro correcto para tener una capa perfectamente unida.

4.2.2. MECANIZADO LÁSER

Para la fabricación de ESB1_PRC (Figura 4.36 y Figura 4.35) se ha optado por el uso de metacrilato (PMMA) por tener unas mejores propiedades mecánicas que eran necesarias para estas piezas. Dicho material suele venir en planchas rectangulares que pueden ser mecanizadas con cortadoras láser. De esta forma se cortaron las 10 piezas que requería el diseño en planchas de 3 mm de grosor. Para ello, se han seguido el mismo flujo de trabajo propuesto en [47]. Sin embargo, a diferencia de la fabricación con FDM, se realizaron los siguientes cambios en las fases que afectaban a la tecnología utilizada:

- **Fase CAM:** a diferencia del uso de impresoras 3D es necesario utilizar únicamente el contorno del área que se quiere cortar. En este caso se ha optado con utilizar extensiones DXF que cuentan con la vectorización del perímetro de la pieza. Dicho fichero puede ser generado fácilmente con Fusion 360 para luego ser cargado directamente en la máquina de corte.
- **Fase de Fabricación:** se ha usado la cortadora Gesmain Co2 5080 (Figura 4.48) para hacer el corte de las planchas de metacrilato. Esto se debe a que cuenta con las características dimensionales y de potencia (Tabla 4.4) requeridas para estas 10 piezas (Figura A3. 10). Hay que tomar en cuenta que la potencia puede afectar a la capacidad de corte de la máquina ya que, como utilizamos un material con mejores prestaciones

¹ Cada impresora y filamento puede presentar ciertas diferencias.

mecánicas como es el PMMA, mientras más grosor tenga la plancha más potencia y número de pasadas necesitará la cortadora láser.

- **Fase de post-procesado:** se requiere hacer la unión de las piezas cortadas correspondientes a las dos partes de ESB1_PRC. Para este propósito se optó por el uso de acetona al 100% ya que este compuesto químico es capaz de actuar como disolvente del metacrilato. Es decir, que al colocar este compuesto sobre la superficie de una pieza se disuelve una fina capa de PMMA que puede conseguir una unión permanente al ponerse en contacto con otra pieza. Se comprobó que este método tenía mejores resultados que pegamentos de alta resistencia o específicos para plástico. Sin embargo, a pesar de que puede dejar ciertos residuos de aire, los resultados fueron los que se buscaban para que este módulo del robot tuviera su funcionalidad correspondiente (Figura 6.6).

Tabla 4.4 Características CO2 3050¹

Área de trabajo (mm ²)	500x800
Velocidad máxima(mm/s)	1000
Potencia(W)	60
Materiales aceptados	Todo tipo de materiales duros no metálicos



Figura 4.48 Cortadora láser Gesmain Co2 5080

Esta cortadora láser se encuentra en el laboratorio de Electrónica y robótica de la ETSIDI

4.3. DESARROLLO ELÉCTRICO Y ELECTRÓNICO

En este apartado se verán todos los componentes eléctricos y electrónicos que fueron necesarios para tener las funcionalidades principales del robot y cómo se han configurado sus conexiones. Es esto, se ha en dos partes principales:

¹ Imagen y tabla recogidas de FabLab ETSIDI. Disponible en: <https://fablabetsidi.com/index.php/installaciones/>

1. **Unidad de control (CU):** es un módulo electrónico que cuenta con la capacidad de analizar y manejar múltiples entradas y salidas que dependen de sus características y limitaciones. Esta unidad debe contar con:
 - a. Control y monitorización de entradas y salidas analógicas y digitales. Se requiere:
 - i. 3 señales digitales para LS
 - ii. 3 señales analógicas para FSR
 - b. Soporte para distintas conexiones seriales (3.6):
 - i. USB
 - ii. I2C
 - iii. SPI
 - iv. UART
 - c. Capacidad de conexión con red WIFI para los distintos protocolos existentes (3.7.3).
2. **Periféricos:** pueden ser tanto CI como componente eléctricos y electrónicos que responden a las necesidades del MCU. Es decir, van a ser los esclavos de la unidad de control. Para el diseño es necesario:
 - a. Módulo de control de SM con posibilidad de comunicación serial con CU. Se necesitan
 - i. 4 unidades de control para cada uno de los motores utilizados.
 - b. Sensores de señales analógicas y digitales. Se necesitan:
 - i. 3 LS para tener la señal de fin de recorrido de cada eslabón
 - ii. 3 FSR para lectura de presión en SG_FLX
 - c. Salida de mensajes por pantalla. Para ello se usa:
 - i. 1 pantalla OLED con comunicación serial

4.3.1. UNIDAD DE CONTROL

De acuerdo con los requisitos que son necesarios solucionar se escogió el Arduino¹ MKR1010 (Figura 4.49) como la controladora principal de todo el HW eléctrico y electrónico del robot. De esta forma se puede hacer uso de las librerías propias de Arduino para abstraer las funcionalidades que se necesitan (aunque, se puede incluir código de bajo nivel directamente con los registros propios del MCU). Así, es posible controlar los demás IC y periféricos de la misma placa de una forma sencilla. Esto se debe a que el módulo incluye:

1. **SAMD21:** este es el MCU encargado de proveer toda la funcionalidad a la CU. Entre sus principales características [49] están:
 - o CPU: Cortex-M0+
 - o Reloj: interno y externo de 48MHz
 - o RAM: 32KB.
 - o FLASH: 256KB.
 - o Entradas y salidas: Hasta 52 I/O programables.

¹ www.arduino.cl

- 1 ADC de 12 bits de 20 canales y 1 DAC de 10 bits.
 - Controladores seriales: I2C, SPI, USB, UART
 - Temporizadores: 5 TC de 16 bits, 4 TCC de 24 bits, RTC de 32 bits
2. **Módulo de radio NINA-W102:** este CI es uno de los componentes más importantes y que marcaron la diferencia a la hora de elegir este Arduino. Esto se debe a que es el encargado de establecer la conexión y comunicación con la red WiFi. Entre sus principales características [50] están:
- a. Entradas y salidas de propósito general.
 - b. Interfaz serial: UART, SPI, I2C, CAN.
 - c. Interfaz de antena RF: este módulo soporta comunicaciones con red WiFi y Bluetooth LE¹.
3. **Pines de entradas y salidas digitales y analógicas:** estas están ruteadas desde el MCU hacia las salidas correspondientes a cada pin.
4. **Interfaces de comunicación:** no todas las interfaces de comunicación del MCU están disponibles ya que unas son usadas para comunicarse con los módulos incluidos en la placa. Sin embargo, algunas están ruteadas a pines de salida y entrada para poder utilizarse con el exterior (Figura 4.49).

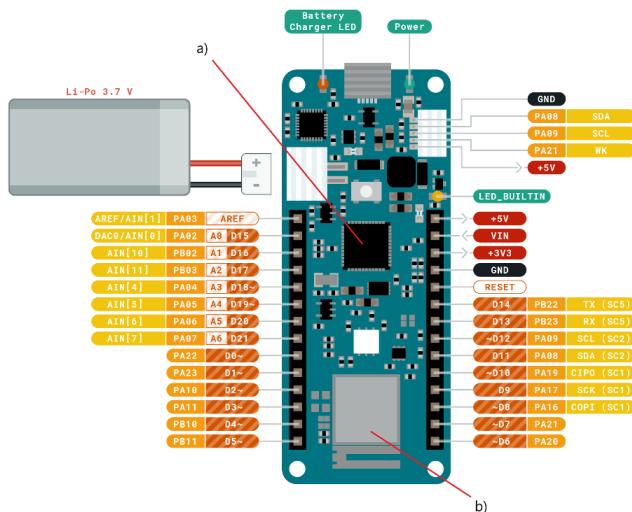


Figura 4.49 Pines de Arduino MKR1010

Se muestran las distintas entradas y salidas de la placa. Se refleja también:

- a) SAMD21
- b) NINA-W102

4.3.2. PERIFÉRICOS

Ya contando con la unidad de control se debe evaluar los periféricos que la CU se encargará de manejar. A partir de los requerimientos principales (4.3) se escogieron los periféricos que se verán a continuación.

4.3.2.1. CONTROLADOR TMC5160

Este es uno de los pilares fundamentales del funcionamiento del robot ya que permitirá un control de alta potencia sobre los SM. El TMC5160, de Trinamic², es un IC que cuenta con

¹ Para futuros proyectos se pueden hacer pruebas con comunicación Bluetooth ya que cuenta con una menor latencia.

² www.trinamic.com

numerosas interfaces de comunicación y que provee de funciones avanzadas para el control de los SM [51]. Entre las principales características [51] están:

- Control de motor de 2 fases
- Soporte de hasta 20 A
- Rango de voltaje 8V – 60V
- Posibilidad de MicroStepping (ver 3.5.2) de hasta 256 micropasos por paso de motor
- Interfaces de comunicación: SPI, UART de un cable
- Funcionalidades de control avanzadas: Control de movimientos suaves, rueda libre, detección de carga, control de energía

Este CI está disponible con un módulo llamado TMC5160 SilentStepStick [52] (Figura 4.50) que cuenta con lo necesario para poder controlar este periférico de forma completa como puede ser: mosfets de protección de 40V, configuraciones para comunicación serial (UART o SPI), reguladores de voltaje y pines de salidas y entradas principales para funcionamiento (

Tabla 4.5) De esta forma se cuenta con las funcionalidades de control del TMC5160 de una forma sencilla y de manera independiente de los demás módulos de la PCB completa. Es

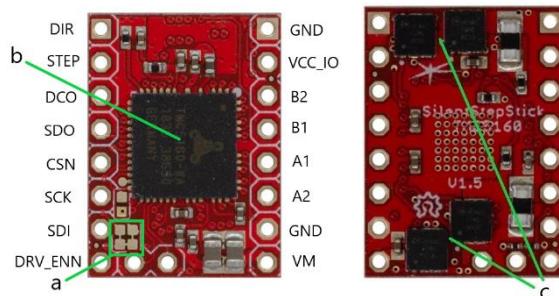


Figura 4.50 TMC5160 SilentStepStick [52]

A la izquierda se tiene una vista frontal del módulo con las señales correspondientes. A la derecha una vista posterior. Se muestra:

- a) Conexiones para comunicación serial:
 - a. SPI: corte de conexión izquierda
 - b. UART: corte de conexión derecha
- b) TMC5160
- c) Mosfet de protección

dicir, que es posible hacer cambios de módulos de control sin necesidad de grandes modificaciones.

Tabla 4.5 Señales de TMC5160 SilentStepStick [52]

Para más información de ubicación de señales ir a Figura 4.50

Señal	Descripción
DIR	Dirección de giro PWM
STEP	Señal PWM
DCO	Entrada de Oscilador para control digital
SDO	Salida de datos seriales
CSN	Selector de módulo para comunicación serial (activa en nivel bajo)
SCK	Señal de reloj de comunicación serial
SDI	Entrada de datos seriales
DRV_ENN	Señal de habilitación de módulo activa en nivel bajo
GND	Tierra
VCC_IO	Salida de voltaje configurable para 3.3V o 5V
B2	Fase B del motor
B1	Fase B del motor

Señal	Descripción
A1	Fase A del motor
A2	Fase A del motor
VM	Voltaje de alimentación del SM: 10-35 V

Ya conociendo el HW y funcionamiento básico del módulo se puede explicar de qué forma se hizo la comunicación con el mismo. Ya se ha visto que existen múltiples formas de utilizarlo, ya sea por comunicación serial o control PWM. Para esta aplicación se optó por hacer una comunicación serial utilizando el protocolo SPI. Esto se debe a que la CU (4.3.1) solo cuenta con una salida de control serial para UART, pero es necesario establecer comunicación con hasta 4 controladores. Como ya se ha estudiado, el protocolo SPI (3.6.1) cuenta con la señal CSN (Figura 3.15) que permite establecer con cuál módulo se quiere hacer comunicación solamente con el uso de una señal digital para cada controlador y las 3 señales de comunicación: MISO, MOSI, CLK. Es decir, que es posible hacer comunicaciones SPI con todos los controladores prácticamente de manera simultánea gracias a la alta velocidad de transmisión que se puede obtener utilizando esta interfaz. Para este propósito se ha seguido la topología de un maestro y múltiples esclavos (Figura 3.14 y Figura 4.51) para que el sistema de control sea totalmente funcional.

A partir de esto es posible, según [51], realizar comunicaciones seriales SPI mediante el uso del modo 3 (Figura 3.16) de este protocolo. De esta forma se puede enviar y recibir datos de una manera relativamente sencilla con un paquete de 5 bytes: el primer byte enviado cuenta con la dirección de acceso al registro (Tabla 4.6) y los siguientes 4 serían la palabra que contiene la información transmitida. Toda esta funcionalidad queda solucionada a través de un SW (5.3.3.2) encargado de codificar y decodificar los paquetes entre el maestro y el esclavo.

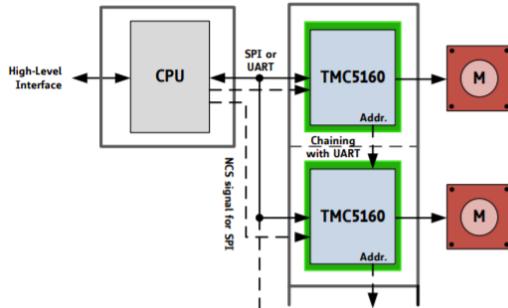


Figura 4.51 Modelo un maestro y múltiples esclavos con TMC5160 [51]

Tabla 4.6 Registros más relevantes de TMC5160 [51]

Nombre de registro	Descripción
GCONF	Configuración general
CHOPCONF	Configuración de chopper para microstepping
IHOLD_IRUN	Límites de corriente de mantenimiento y de desplazamiento de SM
TPOWERDOWN	Tiempo de espera que mantiene el motor con IRUN. Una vez culminado se pone a IHOLD.
VSTART	Velocidad de arranque del motor sin signo
A1	Aceleración de arranque
V1	Velocidad objetivo para A1
AMAX	Segunda aceleración para llegar a VMAX
VMAX	Velocidad objetivo durante movimiento

DMAX	Primera desaceleración entre VMAX y D1
D1	Desaceleración final que termina en VSTOP
VSTOP	Velocidad de detenimiento del motor.
RAMPmode	Tipo de modo de control: control por posición o control por velocidad.
XACTUAL	Posición del motor. Si este registro se cambia se provoca movimiento
XTARGET	Posición objetivo para modo de generador de rampa de velocidad.

A partir de estos registros es posible configurar el controlador tal como se requiere para cada uno de los SM y así tener una rampa de velocidades que el motor pueda seguir (Figura 4.52). Si ya está establecida la configuración general y las corrientes de funcionamiento se puede hacer uso 2 tipos de control:

1. **Control por posicionamiento (RAMPmode = 0):** se establece la posición objetivo (XTARGET) y la velocidad a la cual se quiere hacer el desplazamiento (VMAX) a dicha posición. Para ello se deben configurar todos los registros V y D correspondientes.
2. **Control por velocidad:** se hace el establecimiento de la velocidad que se quiere alcanzar (VMAX) y se establece dirección positiva (RAMPmode = 1) o negativa (RAMPmode = 2) según requerimiento.

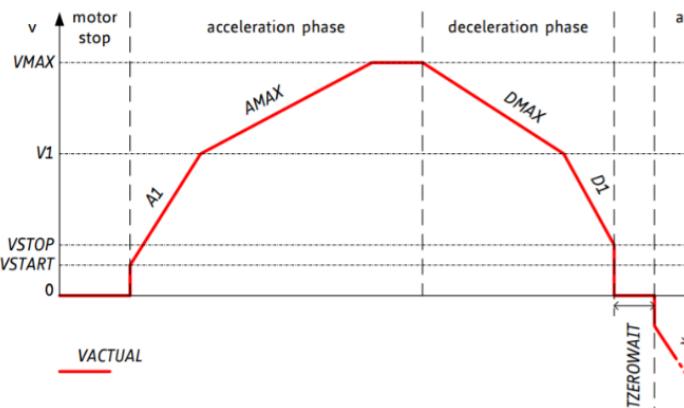


Figura 4.52 Rampa de velocidades TMC5160 [51]

En la imagen se refleja la velocidad que es enviada al motor en función del tiempo. Se ve de forma clara en qué parte de la rampa de velocidades actúa cada registro. Este funcionamiento se daría en ambos modos de control.

Por otro lado, hay que mencionar uno de los puntos que de mayor interés a la hora de controlar los SM. Esto sería la capacidad de realizar control con *MicroStepping* (3.5.2). Con este módulo de control se puede conseguir hasta 256 micropasos por cada paso real del SM. Esto permite realizar un control mucho más fluido y conseguir movimientos más precisos. Para realizar esta funcionalidad el controlador cuenta con un sistema llamado *MicroPlyer* que se puede activar con el registro CHOPCOF [51]. Con esta información se tiene lo necesario para saber qué valores hay que establecer a los registros para que hagan el determinado movimiento angular. Sin embargo, hay que recordar que estos registros funcionan a partir del reloj central que esté utilizando la CPU. Es decir que las unidades [51] serán las indicadas en la Ecuación 4.1.

$$\frac{\mu S}{t}, \text{ donde } t = \frac{2^{24}}{f_{CLK}}$$

Ecuación 4.1

Por lo tanto, si se quiere hacer que el motor se mueva a una velocidad angular determinada, se debería enviar al registro de velocidad lo presentado en la Ecuación 4.2.

$$VMAX = t \cdot MS \cdot NMS$$

Ecuación 4.2

Dónde: t = constante de tiempo de control. NMS = número de micro paso por paso. MS = pasos reales del SM

Finalmente, se realizaron pruebas de consumo en los motores en función de la corriente que se le solicite de mantenimiento o de movimiento. Los resultados fueron expresados en una tabla (Tabla 4.7) para poder tener una curva de funcionamiento (Figura 4.53) que pueda usarse en el control de corriente de cada motor a nivel SW.

Tabla 4.7 Respuesta de corriente según registro IHOLD_IRUN

Hex	Decimal	Corriente (mA)
0x0A	10	264
0x0C	12	340
0x0D	13	385
0x10	16	550
0x12	18	745
0x15	21	950

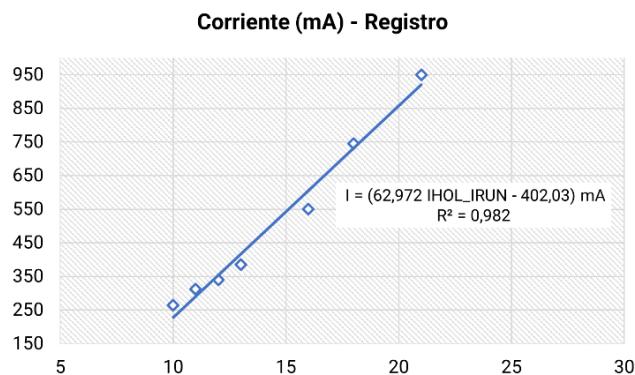


Figura 4.53 Gráfica de corriente en motor en función de registro IHOL_IRUN

4.3.2.2. SENSORES DE PRESIÓN FSR07

Estos sensores serán los encargados de tener las lecturas de presión ejercida en las piezas SG_FLX (4.1.2.2). Como ya se ha mencionado anteriormente (3.4), los sensores FSR pueden variar su resistencia en función de la presión aplicada siguiendo una respuesta conocida (Figura 3.5). Es por ello por lo que son los indicados para obtener la presión que es ejercida sobre el objeto sujetado. Para este propósito se optó por los FSR07 [53] de Ohmite¹ ya que cuentan con las características electrónicas y dimensionales (Figura 4.54) que eran requeridas:

- Área de contacto de circunferencia de 14.70 mm

¹ www.ohmite.com

- Modo shunt (3.4.2)
- Conexión para cable macho y pines para soldadura
- Umbral de comienzo de lectura 15g. Para pesos menores FSR = 10 MΩ

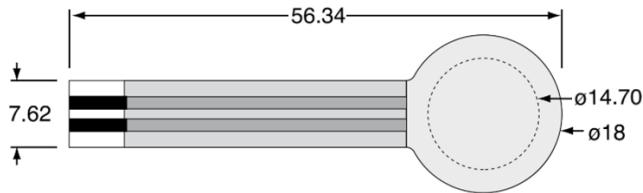


Figura 4.54 FSR07 de Ohmite [53]

Para poder hacer uso de estos sensores es necesario hacer una caracterización de su funcionamiento. Por esto se realizó una prueba de variación de resistencia en función de la fuerza aplicada. Para este procedimiento fue necesario:

- Multímetro: este aparato va a leer la resistencia que tenga el FSR en todo momento durante la prueba. Para ello se hace una conexión en paralelo entre el aparato y el sensor.
- Balanza: este aparato sirve para indicar la fuerza aplicada en todo momento y que va a corresponder con la lectura del multímetro. Cuenta con una resolución de 1g.

Con el sistema de pruebas montado se procedió a ejercer presión sobre la parte activa del sensor apoyado sobre la balanza (Figura 4.55) y se fueron recogiendo los datos correspondientes. Con dichos datos se consiguió, con el uso de Excel, la graficación del comportamiento de la resistencia del sensor en función de la fuerza aplicada.

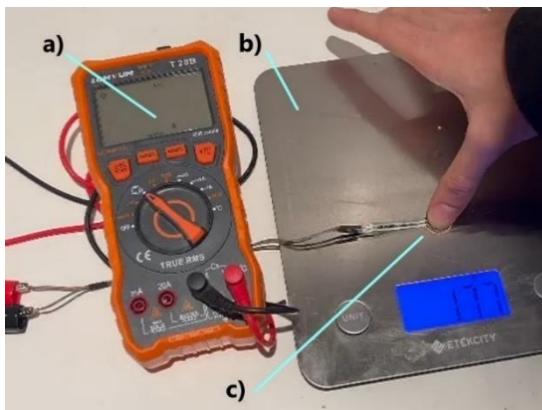


Figura 4.55 Toma de datos para caracterización FSR

Se ve el montaje que se usó para tomar los datos de caracterización del sensor FSR07. En la imagen se muestra:

- a) Multímetro encargado de lectura de resistencia en Ω.
- b) Balanza con capacidad de 5Kg y resolución de 1 g.
- c) Sensor FSR siendo presionado.

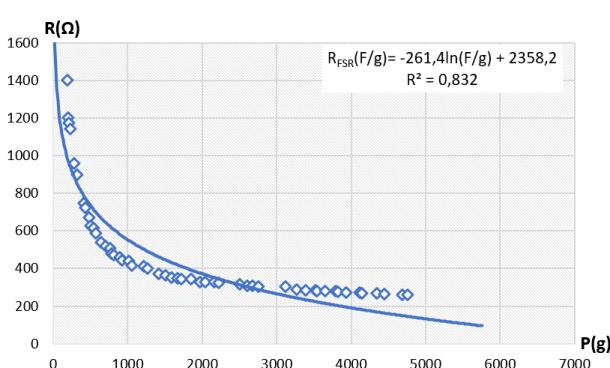


Figura 4.56 Gráfica de resultados de caracterización FSR07 (R - F)

Es evidente que la respuesta del sensor (Figura 4.56) es la que se estaba esperando según el funcionamiento de estos transductores (Figura 3.5). Sin embargo, la aproximación logarítmica

de la función que resulta de la gráfica no es del todo buena. Por esta razón se decidió dividir la curva en 3 tramos corresponde a los 3 mecanismos de funcionamiento con los que estos sensores trabajan y que se estudiaron anteriormente (3.4.1). Se tomaron los siguientes límites:

1. Resistencia de FSR entre 10 MΩ y 750 Ω (Figura 4.59).
2. Resistencia de FSR entre 750 Ω y 300 Ω (Figura 4.58).
3. Resistencias de FSR menores a 300 Ω (Figura 4.57).

La ventaja de hacer la separación en tramos es que se pueden hacer aproximaciones del comportamiento del sensor con otro tipo de funciones como pueden lineales y polinómicas. Esto es una gran ventaja a la hora de traducir estas funciones a código ya que las funciones logarítmicas y exponenciales suelen requerir más recursos del MCU. Además, va a cambiar la relación de la gráfica ya que lo que interesa a nivel funcional va a ser la fuerza que se está aplicando en función de la resistencia que se ha leído con ayuda de los ADC del UC.

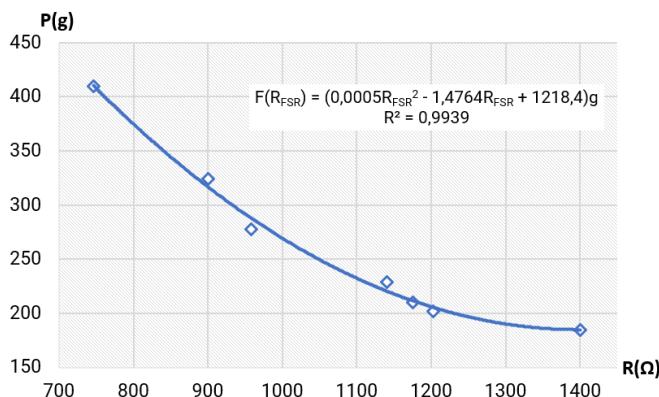


Figura 4.57 Gráfica primer tramo de respuesta FSR07 (Peso - Resistencia)

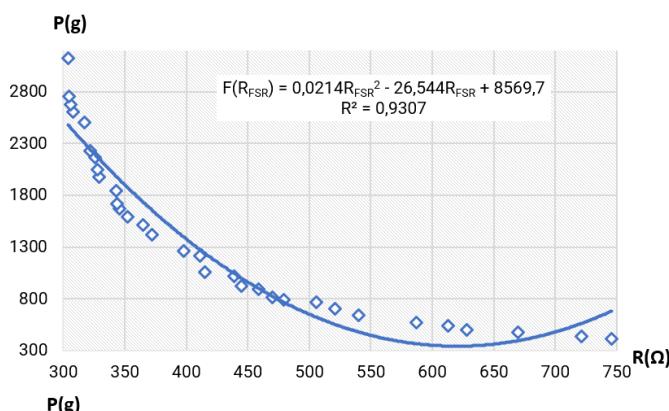


Figura 4.58 Gráfica segundo tramo de respuesta FSR07 (Peso - Resistencia)

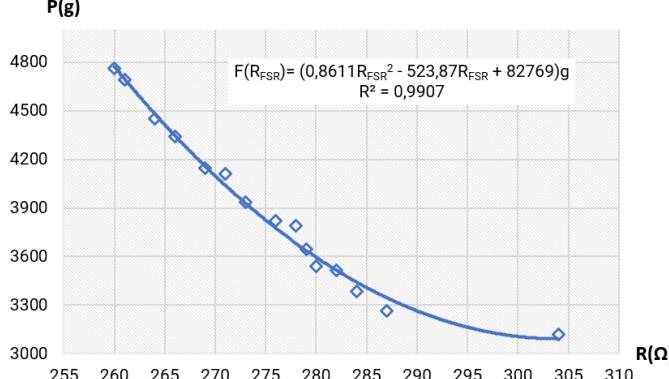


Figura 4.59 Gráfica tercer tramo de respuesta FSR07 (Peso - Resistencia)

Si resumimos los resultados mostrados por los 3 tramos tendríamos la Ecuación 4.3. En estas ecuaciones se representa la multiplicación por la gravedad para tener las unidades de la medida en Newtons.

$$F(R_{FSR}) = \begin{cases} (0.0005R_{FSR}^2 - 1.4764R_{FSR} + 1218.4)g & [N], & 750\Omega \leq R_{FSR} < \infty \\ (0.0214R_{FSR}^2 - 26.544R_{FSR} + 8569.7)g & [N], & 300\Omega \leq R_{FSR} < 750\Omega \\ (0.8611R_{FSR}^2 - 523.87R_{FSR} + 82769)g & [N], & 0 \leq R_{FSR} < 300 \end{cases}$$

Ecuación 4.3

Finalmente, solo se debe escoger la configuración del circuito que servirá de medición de la resistencia R_{FSR} . Se optó por implementar el circuito de divisor de voltaje (Figura 3.7) con una configuración de *pull-up* pero sin el uso de un OPAM. Como lo que interesa es saber el valor de R_{FSR} entonces, a partir de la Ecuación 3.23, se puede deducir fácilmente la Ecuación 4.4 para poder implementarla a nivel SW.

$$R_{FSR} = \frac{R}{\left(\frac{V_{cc}}{V_{FSR}} - 1\right)}$$

Ecuación 4.4

4.3.2.3. SENSORES DE FINAL DE CARRERA(LS)

Como bien dice su nombre estos sensores se usarán para saber cuándo el eslabón donde el sensor esté instalado se encuentra en un punto límite estructural. De esta forma se puede hacer funciones de inicialización de posicionamiento del robot y protección ante errores de control. Por esta razón deben ubicarse en lugares estratégicos del cuerpo del SCARA para que puedan cumplir correctamente con su función.

Se ha optado por los sensores mecánicos verticales de LERDGE¹ ya cuenta con la orientación que se requiere para ser instalarlos en el cuerpo del robot. Además, vienen incluidos los componentes electrónicos necesarios (led de indicación de activación, resistencias y condensador) para tener el sensor funcional y así solamente es necesario hacer la conexión de las 3 señales que el módulo requiere para el funcionamiento (Figura 4.60).

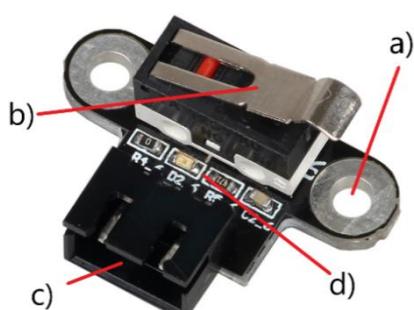


Figura 4.60 LS de LERDGE

- a) Entrada de tornillos M3.
- b) Componentes electrónicos.
- c) Entrada y salida de señales. De izquierda a derecha se ordena: señal de sensor, tierra, alimentación.
- d) Sistema de palanca para activación de sensor por contacto.

¹ www.lerdge.com

4.3.2.4. PANTALLA OLED

Esta pantalla va a ser de gran ayuda en caso de que no se tenga conexión serial para poder saber ciertos valores de interés. Por ello será, además del módulo SW de la GUI, otro sistema de HMI. Se eligió el módulo de pantalla oled de AZ-Delivery¹ (Figura 4.61) que cuenta con las siguientes características:

- Comunicación I2C (3.6.2)
- Pantalla de 168 x 64 pixeles
- Pines de conexión reducidos: Vcc, GND, SDA, SCL
- Consumo bajo de menos de 11 mA

Este módulo tiene como principal CPU el SSD1306 [54] que provee todas las funcionalidades esenciales para imprimir valores en la pantalla oled. Todo ello queda reducido a 4 pines de control a través de los cuales se puede alimentar el módulo y también comunicarnos con este a través del protocolo I2C con el uso de 2 cables. Para esta comunicación es necesario tomar en cuenta la dirección I2C con la que cuenta el módulo. Según [54] esta dirección es 0x3C. Con esta información y la estructura de paquetes del IC se pueden enviar los mensajes correspondientes para el control de las imágenes mostradas por la pantalla. Se verá más adelante que esta comunicación se puede resolver con el uso de librerías específicas para este CI de una forma relativamente sencilla (5.3.3.3).



**Figura 4.61 Pantalla OLED
AZDelivery**

A la izquierda te ve una imagen frontal del módulo con las correspondientes señales a cada pin. A la izquierda se tiene una vista en perspectiva del módulo. Se puede ver las 4 entradas para tornillos M2.

4.3.2.5. MOTORES NEMA

Los últimos componentes eléctricos que hacen falta para construir el robot son los motores. Para esta máquina se optó por el uso de motores SM híbridos (3.5.3) debido a que son los más empleados en robótica para este tipo de sistemas. Estos motores siguen los estándares establecidos por la NEMA², la cual les da nombre. Esto hace que su uso sea bastante aconsejable debido a que son fáciles de conseguir desde distintos proveedores manteniendo las mismas características funcionales y dimensionales. Además, pueden ser manejados con numerosos tipos de controladores, entre ellos el TMC5160 (4.3.2.1).

Estos SM suelen ser referidos como NEMA XX, donde XX es el diámetro del frontal del SM en pulgadas por 10 (por ejemplo, NEMA 17, para un lado de 1.7 pulgadas). Para su

¹ www.az-delivery.de

² www.nema.org

caracterización se debe hacer referencia a [55] en donde se reflejan los estándares de varios sistemas de control industrial. Entre las características que hay que destacar se encuentran:

- Voltaje y corriente nominal
- Número de fases del motor
- Pasos por revolución
- Par de detención
- Par de mantenimiento

Con estas características se puede establecer cuál va a ser el motor que mejor convenga para la aplicación. Normalmente el tamaño del SM es proporcional a sus características por lo que a medida que la numeración del motor aumente, más potencia va a aportar. Sin embargo, también va a conllevar incluir más peso en el sistema. Esto es un factor importante para considerar en la elección, puesto que cuanto más peso tenga la estructura mayor será el par necesario para conseguir que esta se mueva. Por esta razón se observó que los SM que mejor se adaptan al diseño son:

- **NEMA 11**: este es el motor escogido para hacer los desplazamientos lineales de ESB3 (4.1.4.4) y para realizar la acción de sujeción del SG (4.1.2). Esta elección fue motivada principalmente por la búsqueda de un SM liviano que pueda tener la fuerza suficiente requerida para estos sistemas de transmisión. Se escogieron 2 tipos diferentes de este modelo de SM:
 - o Para el ESB3 se optó por un NEMA 11 [56] convencional unido a un tornillo de avance T8 de 30 cm de longitud (Figura 4.42).
 - o Para el caso del SG se escogió un motor NEMA 11 no cautivo [57]. Es decir, que no cuenta con un eje de transmisión de rotación, sino que el rotor realiza la rotación sobre un tornillo de desplazamiento. De esta forma se ve transformado el movimiento rotativo a un movimiento lineal que es necesario para el SG.

Entre las características a destacar de este motor se encuentran:

- o Grados por paso: 1.8°
- o Número de fases: 2
- o Corriente nominal por fase: 0.75 A
- o Peso: 180 g
- o Torque de mantenimiento: 8 N.cm
- o Torque de detención: 0.8 N.cm



Figura 4.62 NEMA 17

En la imagen inferior se ve un NEMA 17 convencional. En las dos imágenes superiores se hace referencia a un NEMA 17 no cautivo a través del cual pasa un tornillo de avance Tr5.56x4.8768mm y 15 cm de longitud. (Imágenes tomadas de: www.omc-stepperonline.com)

- **NEMA 23:** este es el SM (Figura 4.63) escogido para desempeñar la transmisión de movimiento para los módulos ESB1 (4.1.4.2) y ESB2 (4.1.4.3). Esta elección se debió a las exigencias de potencia que ambos eslabones requerían. Además, como ya se ha visto, el diseño cuenta con un sistema de transmisión que ubica a estos dos motores en la base del robot. Por ende, el peso de estos no va a afectar la inercia de la máquina. Entre las características [58] a destacar de este motor se encuentran:
 - Grados por paso: 1.8°
 - Número de fases: 2
 - Corriente nominal por fase: 2 A
 - Peso: 820 g
 - Torque de mantenimiento: 120 N.cm mínimo
 - Torque de detención: 3.5 N.cm máximo



Figura 4.63 Nema 23

Imagen tomada de: www.electronicaembajadores.com

4.3.3. ESQUEMA DE CONEXIONES CON PCB

Con todos los componentes necesarios para el funcionamiento del sistema se verá a continuación cómo se realizaron las conexiones que cada uno de estos componentes. Para ello se ha tomado como base una PCB diseñada en anteriores proyectos (los esquemáticos se pueden encontrar en Anexos 2) y enfocada en el uso del Arduino MKR 1010 como controlador de hasta 4 SM por comunicación SPI. Además, cuenta con bornas conectadas a salidas de interés que servirán para el cableado de cada uno de los periféricos que se han explicado anteriormente (4.3.2).

4.3.3.1. CONEXIÓN DE MOTORES CON TMC5160

Esta conexión se hace en base a lo anteriormente planteado de un maestro y múltiples esclavos (Figura 4.64). Se dispone de una configuración en la que cada esclavo comparte la señal del SPI y las señales del NCS es correspondiente a cada controlador. Hay que resaltar de esta configuración que los pines de NCS pueden variar según la aplicación y los requerimientos. Además, hay que tener presente que las fases de los motores deben estar conectadas de forma ordenada y acorde a la dirección de cada una. Es posible tener problemas en el control al invertir los cables provocando que no se mueva el motor o que este haga movimientos inversos e impredecibles que pueden confundirse con errores de SW.

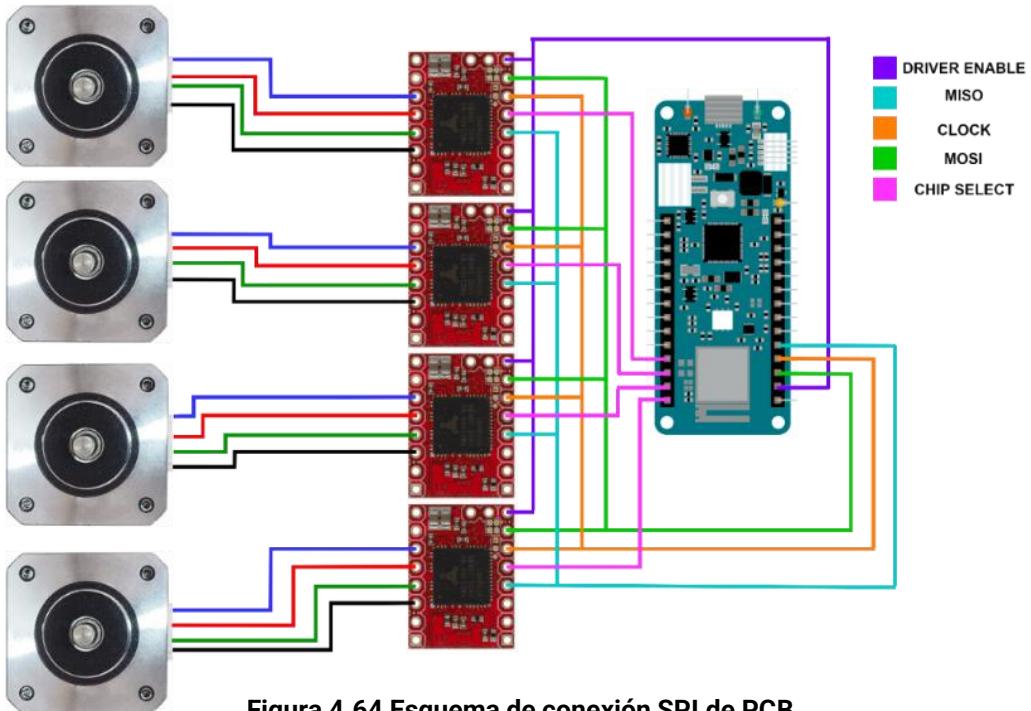


Figura 4.64 Esquema de conexión SPI de PCB

Se ve la topología de un maestro y varios esclavos (Figura 4.56). En este caso en Maestro es el Arduino MKR 1010 el cual control a 4 TMC5160. Se está suponiendo que las alimentaciones de los módulos están conectadas.

4.3.3.2. CONEXIÓN DE SISTEMA DE ENTRADAS ANALÓGICAS Y DIGITALES

Para tener la PCB completamente montada hace falta conectar los demás periféricos de entrada y salida de información del CU. Se verá que algunos sensores serán conectados a entradas analógicas, los FSR (3.4), y otros a entradas digitales, los LS (Ecuación 4.4), según su correspondiente funcionamiento. Además, es necesario hacer la conexión de los 4 pines de la pantalla OLED (4.3.2.4) para tenerla funcionando en conjunto con el sistema (Figura 4.65).

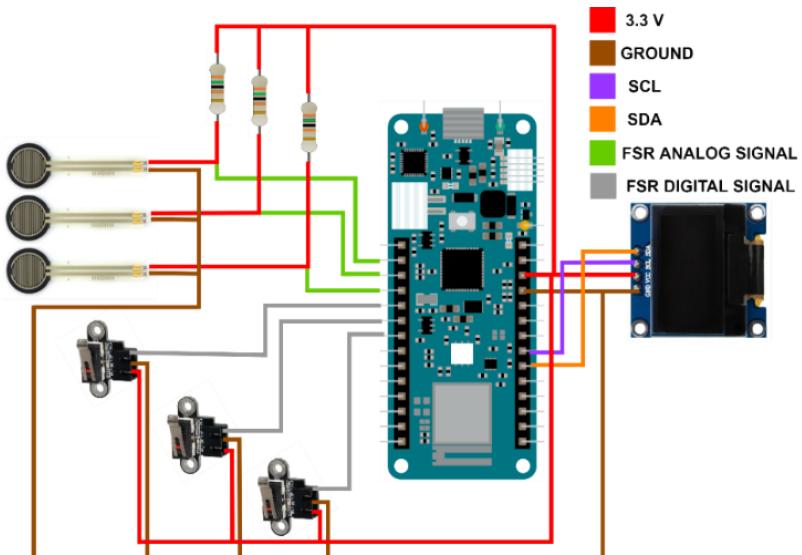


Figura 4.65 Configuración de sensores en PCB

Las resistencias reflejadas no son las que se han usado en el montaje final de la placa.

Capítulo 5. DESARROLLO: SW

En este capítulo se estudiará la arquitectura del SW existente en la aplicación. Para su desarrollo se ha dividido este apartado en 2 módulos principales:

- **Interfaz de usuario (GUI):** es la herramienta HMI de entrada y salida de comando que puede ser controlada desde una máquina o servidor web. Debe contar con:
 - o Entradas y salidas de comandos
 - o Entradas y salidas visuales
 - o Capacidad de decodificación y codificación de paquetes
 - o Herramientas de conexión a través de IP (3.7.3)
- **Aplicación de control:** encargada de hacer la lectura de los comandos de movimiento del robot y que estos sean transmitidos a su correspondiente actuador. Puede ser una aplicación orientada a un sistema físico o que se encargue de simular los comandos a través de una interfaz gráfica. Debe tener:
 - o Conexión a la red por IP para entrada y salida de datos
 - o Capacidad de decodificación y codificación de paquetes
 - o Comunicación y control del robot

Con esto en mente se puede tener una idea clara de cómo hay que empezar la arquitectura general que integrará ambas partes del SW (Figura 5.1). Sin embargo, siempre hay que considerar la importancia de desarrollar módulos SW totalmente desacoplados. Es decir, que, aunque exista comunicación entre ambos, estos puedan funcionar de forma totalmente independiente. Así se asegura que, en caso de requerir alguna modificación en cualquiera de los módulos, no se vea afectado todo el sistema.

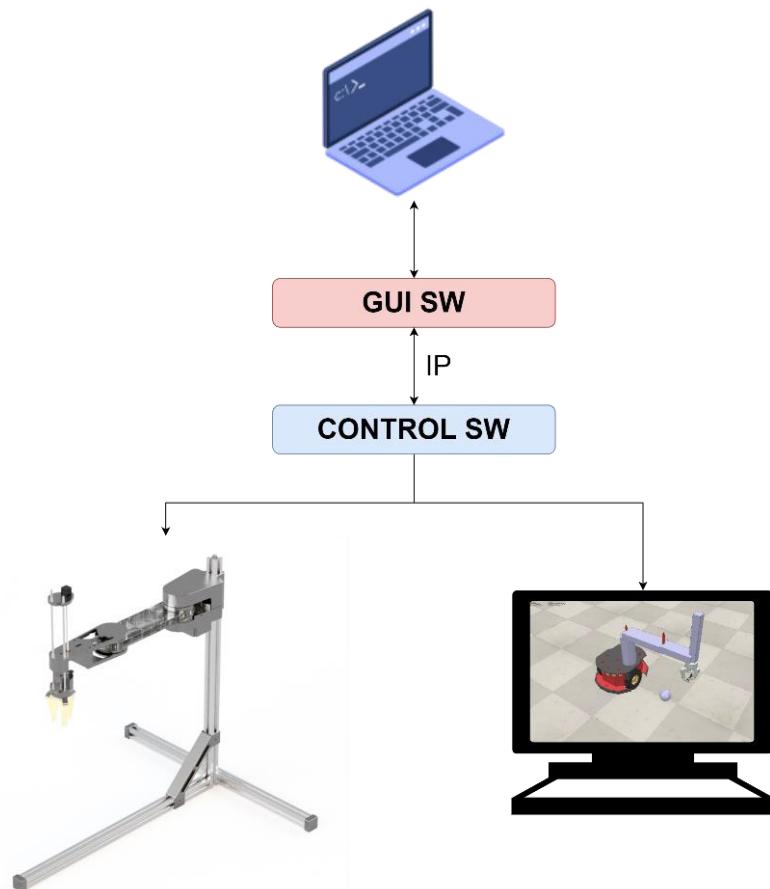


Figura 5.1 Arquitectura SW general

5.1. HERRAMIENTAS UTILIZADAS

Para el desarrollo de ambos módulos del SW se utilizaros 3 herramientas principales para su codificación, depuración y control de versiones. A continuación, se verá un resumen de cada una de las que fueron fundamentales para crear toda la aplicación y mantener un control de versiones.

5.1.1. ARDUINO IDE

Para poder hacer la programación de la UC se hizo uso de esta IDE gratuita provista por la misma compañía que ofrece el módulo MKR 1010, Arduino¹. Se utilizó la versión 2.0 Beta (Figura 5.2) para poder hacer la comunicación con el MCU y aprovechar todas las librerías que ofrece Arduino y otros desarrolladores de FW para hacer un control de los periféricos de los componentes de la placa y módulos externos a la misma. Entre las principales ventajas que ofrece esta aplicación se encuentran:

- Interfaz gráfica sencilla para uso provista de IA para facilitar programación
- Compilador para lenguaje en C++ enfocado en objetos
- Consola para comunicación con puerto serial del módulo MKR 1010
- Programación de placa a través de puerto USB
- Herramientas propietarias y de terceros para facilitar codificación y control de periféricos

Esta IDE se utilizó principalmente para la carga del código en la placa ya que, como es la versión beta, aún tiene algunas herramientas que no están bien optimizadas para facilitar la codificación. Sin embargo, se consiguen todas las librerías necesarias para poder trabajar con cualquier placa ofrecida por esta compañía y con numerosos periféricos.

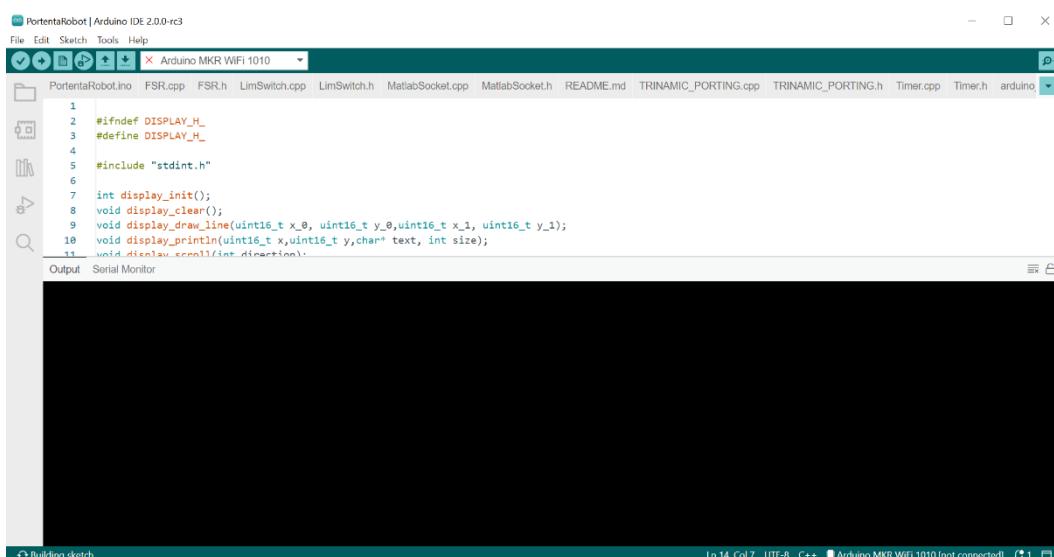


Figura 5.2 Arduino IDE 2.0 Beta

¹ www.arduino.cl

5.1.2. VISUAL STUDIO CODE

Esta es una herramienta de libre acceso ofrecida por Microsoft ¹. Tiene múltiples ventajas que provocaron su elección para la programación. Entre ellas se encuentran:

- Integración con la gran mayoría de lenguajes de programación
- Soporte para múltiples sistemas operativos
- Herramientas avanzadas de IA para programación
- Integración con herramientas de control de versiones git²
- Consola de comandos para ejecución de programas y descarga de paquetes y librerías
- Compiladores y depuración de múltiples lenguajes de alto y bajo nivel

Por esta razón el código fuente de los programas de ambos módulos de la aplicación fueron editados y programados desde esta interfaz gráfica (Figura 5.3). Por un lado, el módulo GUI se ejecuta desde la línea de comandos para hacer pruebas y para tener también una consola en la cual imprimir texto. Por otro lado, los ficheros fuente para la aplicación de control se editaron desde este SW para luego ser utilizados por la IDE de Arduino y cargados en la placa.

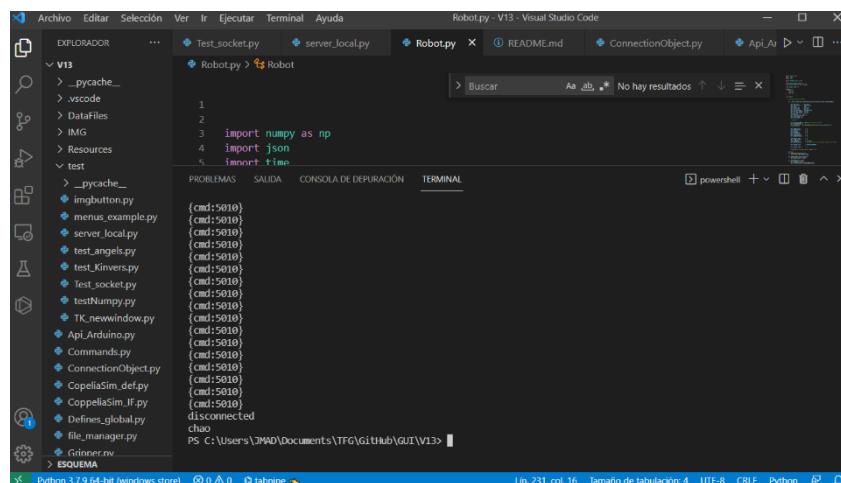


Figura 5.3 Visual Studio Code

En la imagen es posible ver a la izquierda el código fuente de la GUI. A la derecha se tiene el fichero Robot.py abierto y debajo de este la consola de comandos propia de VSC

5.1.3. GITHUB DESKTOP

Esta es una herramienta que permite la conexión con controladores de versiones git. Fue desarrollada por GitHub para poder tener un cliente git en forma de GUI (Figura 5.4) para facilitar el control de versiones del código en los repositorios que se desee. Entre las ventajas de su uso están:

- Conexión con cualquier sistema de control de versiones basado en git

¹ www.code.visualstudio.com

² www.git-scm.com

- Interfaz gráfica de fácil uso e integrada con todas las funcionalidades git: creación de repositorio, descarga de repositorio, subida de cambios en ficheros, creación de rama, etc
- Visualización sencilla de todos los cambios realizados sobre el código que facilita la detección de fallos

```

@@ -109,7 +109,7 @@ class GUI():
    self.powerON()
    else:
        self.powerOFF()
        self.updateFSRValue(json_parser.lookForObjectValue("FSR"))
        self.updateFSRValue(json_parser.lookForObjectValue("FSR_TOP"), json_parser.lookForObjectValue("FSR_BOTTOM"))
        if int(json_parser.lookForObjectValue("LS")) > 0:
            self.turnOnLSRed()
        else:
            self.turnOffLSRed()

@@ -222,28 +222,35 @@ class GUI():
    def createSensorControlCanvas(self):
        self.my_canvasSensor = Canvas(self.root, width=150, height=110, background= BG_CANVAS_MOTORS)
        self.my_canvasSensor.create_text(70,15,text="Sensor Control",font= self.MonserratFont_10, fill= TITTEL_CANVAS)
        self.my_canvasSensor.create_text(70,10, text="Gripper Sensing", font= self.MonserratFont_10, fill= TITTEL_CANVAS)
        self.my_canvasSensor.create_text(50,70, text="Limit Switch", font= self.MonserratFont_10, fill= TITTEL_CANVAS)
        self.my_canvasSensor.create_oval(95,60,115,80, fill= 'black', outline= 'grey')
        self.my_canvasSensor.create_text(40,35, text="FSR_TOP", font= self.MonserratFont_10, fill= TITTEL_CANVAS)
        self.FSR_topVal = Entry(self.my_canvasSensor, width=10, borderwidth=5)

        self.my_canvasSensor.create_text(50,95, text="Power", font= self.MonserratFont_10, fill= TITTEL_CANVAS)

```

Figura 5.4 GitHub Desktop

A la izquierda se puede ver el fichero en el cual se están evaluando los cambios respecto del código actualmente subido en el repositorio correspondiente a la GUI. A la derecha se puede ver los cambios hechos comparados con el código actual. En la parte inferior izquierda el botón para hacer el commit del código actual con algún comentario que haga referencia.

Todo el código fuente del proyecto se encuentra en las siguientes direcciones:

- GUI: https://github.com/juandiz/ArduinoControl_TFG
- Aplicación de control: https://github.com/juandiz/ArduinoControl_TFG

5.2. MÓDULO GUI

Como ya se ha mencionado anteriormente este módulo de SW va a encargarse de ser la primera entrada y salida de comandos y señales fundamentales para el control del robot. Es decir, que, además de encargarse de los efectos visuales, debe ser capaz de hacer gestiones internas de información. Por esta razón debe cumplir con los siguientes requisitos:

- Entrada de múltiples señales de control:
 - o Posición requerida para el robot
 - o Movimiento de la base móvil
 - o Control de funciones de los motores
 - o Visualización de señales analógicas y digitales provenientes de la aplicación de control
- Interfaz de fácil uso y con múltiples funciones:
 - o Establecimiento de parámetros de conexión red
 - o Configuración de comunicación con SW de control
 - o Entradas de botones para generar acciones múltiples
 - o Salidas y entrada de texto

- Visualización de estados de comunicación y control
- Contar con una arquitectura multihilo para control de procesos en paralelo requeridos para un buen funcionamiento de toda la aplicación
- Utilizar el paradigma OOP con un lenguaje de alto nivel que sea flexible y cuente con soporte de librerías avanzadas
- Lógica de codificación y decodificación de comandos propios de comunicación con aplicación de control
- Sistema de ficheros para control de información

Con los principales requisitos establecidos se puede evaluar la estructura utilizada en este módulo del SW (Figura 5.5). A partir de esta configuración es posible estudiar cada una de las partes principales que lo conforman siguiendo la filosofía POO. Así se asegura de que cada una de las partes se encarguen de realizar tareas específicas y que funcionen de manera desacoplada al resto.

Es importante recordar también que, como se propone un funcionamiento con multihilos, es necesario tener herramientas de sincronización y conexión de procesos. Entre los que se requieren están:

- **Colas:** esta herramienta servirá para conectar de forma desincronizada procesos mediante el envío de estructuras de datos de forma ordenada a través de un objeto cola (*queue*).
- **Publicaciones:** esta herramienta permite hacer conexiones entre procesos a través de la ejecución de eventos específicos desde uno de ellos hacia el otro. Soportan también el envío de datos.

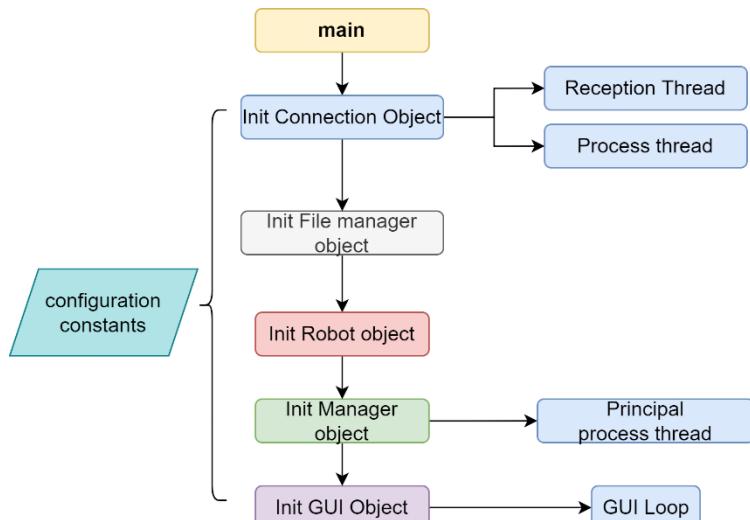


Figura 5.5 Arquitectura general SW GUI

Este es el punto de arranque de toda la aplicación GUI. Se encuentra en el fichero main.py

5.2.1. IMPLEMENTACIÓN DE GUI CON PYTHON

Ya con un esquema establecido y los requerimientos que debe tener este módulo se puede pasar a implementar su funcionalidad. Para este desarrollo se escogió Python 3.7¹ como el

¹ www.python.org

lenguaje de programación de alto nivel utilizado. Esto se debe a que cuenta con las siguientes características:

- Soporte de múltiples librerías fundamentales para el desarrollo
- Lenguaje orientado a objetos de alto nivel
- Lenguaje interpretado para aumentar eficacia en programación
- Es de libre acceso
- Es limpio y fácil de leer
- Está soportado por múltiples SO

Para hacer uso de este lenguaje se utilizó VSC. Desde este se puede hacer la instalación de todos los paquetes y librerías de Python (Tabla 5.1) que son necesarias para poder compilar y ejecutar la GUI. Algunas de estas librerías se descargaron gracias al instalador de paquetes de Python, pip¹, y otras ya vienen integradas con el paquete del lenguaje.

Tabla 5.1 Requerimientos para SW de GUI

Nombre	Comando pip	Descripción
numpy 1.22.4	pip install numpy	Paquete para cálculos matemáticos avanzados
sockets	-	Control de sockets para conexión UDP o TCP
Time	-	Control de tiempo
json	-	Codificación y decodificación de paquetes json
queue	-	Creación y control de colas de datos multicast
theadring	-	Funciones de creación y manejo de hilos
Pypubsub	pip install pypubsub	Manejo de publicaciones
tkinter	-	Construcción de interfaz gráfica con funciones avanzadas de visualización y manejo de eventos

5.2.2. OBJETOS DE SW GUI

A partir de la arquitectura general del SW del GUI (Figura 5.5) se evaluarán a un alto nivel cada uno de sus componentes.

5.2.2.1. OBJETO DE CONEXIÓN

Este objeto es creado a partir de la clase *ConnectionObject* y se encarga de gestionar la conexión vía red. Para cumplir sus funciones es necesario que utilice hilos de procesos para poder estar evaluando constantemente el estado de conexión y para que el proceso de recepción de datos no presente un problema de bloqueo de ejecución del programa. Además, debe contar con la posibilidad de crear y controlar sockets para hacer comunicaciones vía red con la filosofía cliente servidor (3.7). Es decir, que este objeto es el encargado de la capa 4 y 5 del modelo OSI de comunicación (Figura 3.21). Su objetivo principal es el de mantener 2 procesos activos:

¹ www.pypi.org/project/pip/

- **Proceso de recepción:** (Figura 5.6) hace tareas de gestión de la conexión IP del SW de la GUI con la aplicación correspondiente de control. Sus principales funciones son:
 - o Esperar una recepción de información proveniente del socket en el cual se haya establecido conexión
 - o Ejecutarse únicamente si hay una conexión abierta
 - o Decodificar el paquete JSON recibido para almacenarlo en la memoria del programa

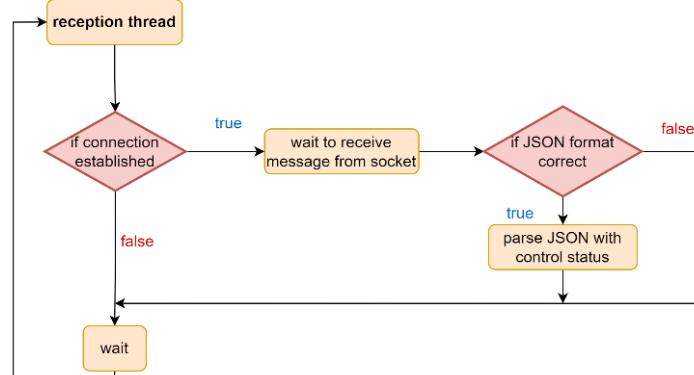


Figura 5.6 Hilo de recepción de Objeto de conexión

- **Proceso principal:** este proceso es el encargado de:
 - o Establecer tipo de conexión: IP/UDP o IP/TCP (3.7.3)
 - o Gestionar los pedidos de conexión y desconexión con la red
 - o Enviar los mensajes nuevos recibidos desde una cola destinada al envío de paquetes JSON con protocolo de aplicación (5.5.1)

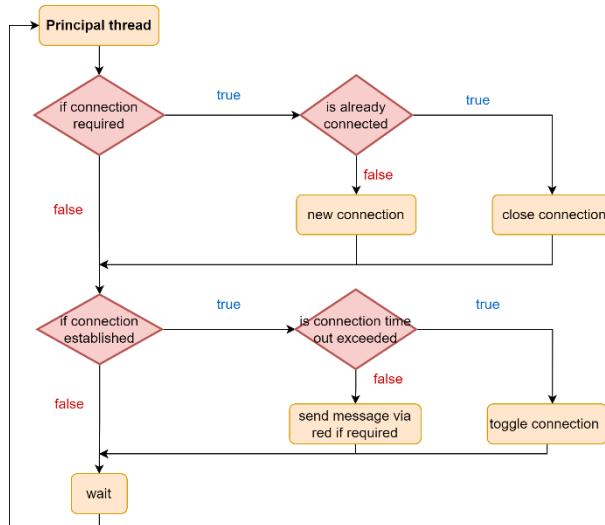


Figura 5.7 Hilo principal de Objeto de conexión

Lo más importante de este objeto es que sirve para abstraer la funcionalidad de la conexión entre la aplicación GUI y la de control. Para ello se utiliza el concepto de polimorfismo de clases. Gracias a este paradigma de programación es posible gestionar las principales funciones de conexión con un mismo objeto y así soportar 2 tipos diferentes de aplicaciones de control: API para control desde CU y API para control de simulación con CoppeliaSim (5.4).

Finalmente, se debe resaltar que este objeto es capaz de hacer comunicaciones con el objeto GUI (5.2.2.5) gracias a un sistema de publicaciones que ayuda a tener distintos eventos y señales que provocan la ejecución de funciones específicas en la interfaz. Así es posible tener

conocimiento visual de cómo ha ido el proceso de conexión y qué valores tienen los sensores presentes en el robot (4.3.2.3 y 4.3.2.2). Además, requiere del uso de una cola de cadena de caracteres para sincronizar el envío de datos a través del socket de conexión. Esta cola es pública y puede ser accedida desde cualquier proceso de la aplicación.

5.2.2.2. OBJETO DE ADMINISTRACIÓN DE FICHEROS

Este objeto cuenta con la tarea principal de brindar soporte de memoria a la aplicación GUI. Para ello se encarga de abstraer las tareas de lectura y escritura de ficheros en donde se quiera guardar información relevante de la aplicación. Para esta funcionalidad el objeto es creado a partir de la clase *data_file*⁷ que se encarga de organizar las distintas funciones de manejo de ficheros: open, read, write, seek, etc. De esta forma se puede aportar una mayor funcionalidad en el manejo de archivos de texto y que cuenten con un formato personalizado. Por esta razón el objeto requiere para su creación:

- Ubicación del archivo global o relativa al fichero donde se encuentre la clase *data_file*. En caso de no existir lo crea.
- Nombre o ID de los atributos que se quieren almacenar en formato de *String*
- Caracteres de inicio y fin de valor del atributo para establecer el formato al fichero

Una vez inicializado el objeto se pueden realizar escrituras y lecturas de cualquiera de los atributos almacenados en el fichero desde cualquier otro proceso que incluya el objeto de administración de ficheros (Figura 5.8). Únicamente hace falta indicar el ID que se quiera manejar y, en caso de escritura, especificar el valor que se quiere actualizar.

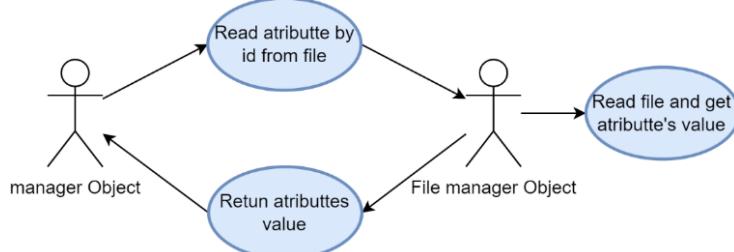


Figura 5.8 Caso de uso de lectura de valor desde el objeto mánager

En resumen, la principal funcionalidad de este objeto es la de aportar un almacenamiento de datos de la aplicación de forma continua, para que, en caso de que la aplicación se reinicie, no sea necesario volver a cambiar todos los parámetros de conexión red o del robot. Así se facilita en gran medida la interacción del usuario con la interfaz gráfica.

5.2.2.3. OBJETO ROBOT

Este objeto es creado a partir de la clase *Robot*. Para su inicialización requiere:

- Nombre del robot
- Limitación de las articulaciones
- Posición inicial

¹ Esta clase se encuentra en fichero file_manager.py

- Máximo rango de funcionamiento de los eslabones

Su principal función es la de almacenar los datos principales del robot para que estos puedan ser usados por otros procesos. Por esta razón requiere de todos los valores principales de la máquina y sirven para:

- Dibujar rango de cobertura del robot en el plano cartesiano de la interfaz gráfica
- Contar con los últimos valores de los sensores que se encuentren en el robot
- Tener la posición actual de cada uno de los motores

Por estos requerimientos es que el robot se va a actualizar desde la recepción del JSON de estado que el objeto de conexión actualiza constantemente a una frecuencia indicada por la aplicación de control (en caso de usar un simulador no se recibiría ningún estado). Y, por lo tanto, este objeto únicamente va a tener funciones de lectura y escritura (Figura 5.9) de variables¹.

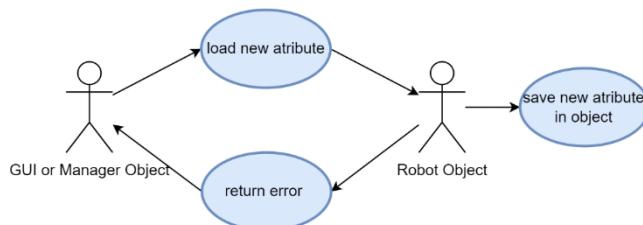


Figura 5.9 Caso de uso en escritura de valor de atributo del objeto Robot

Además de lo anterior mencionado, el objeto robot puede ser el encargado de hacer el cálculo de la trayectoria que se va a ejecutar. Es decir, que, si es necesario, se pueden habilitar las funciones matemáticas con el uso de *numpy*² para calcular la cinemática inversa y directa del robot. Esto puede ser de gran ayuda, en caso de que el MCU no cuente con la potencia suficiente para realizar estos cálculos. Sin embargo, pueden existir problemas de mayor latencia en el envío de los paquetes ya que debe enviar cada una de las posiciones que el robot debe seguir para realizar la trayectoria completa de manera fluida. Es decir, que el tamaño del paquete sería más grande y por ende el MCU tardaría más en hacer la decodificación del mensaje JSON (5.5).

5.2.2.4. OBJETO MÁNGER

Para que exista una coordinación y comunicación entre los eventos de los procesos involucrados en el módulo de aplicación GUI se ha optado por el uso de una tarea dedicada exclusivamente a la evaluación de los múltiples eventos existentes en la aplicación. Esta tarea es efectuada con el objeto mánager que se crea a partir de la clase *man_control*³ que hereda todas las cualidades del objeto hilo (*Thread.threading*) de Python gracias al uso del módulo *threading*⁴ que está integrada en el paquete del lenguaje.

¹ Este objeto puede ser creado a partir de un objeto fichero para contar con la misma finalidad. Se utiliza para tener un código más descriptivo.

² www.numpy.org

³ Clase declarada en manager.py

⁴ www.docs.python.org/3/library/threading.html

Para inicializar el mánager hace falta el conocimiento del objeto robot (5.2.2.3), el objeto de conexión (5.2.2.1) y un objeto administrador de ficheros (5.2.2.2). Una vez inicializado es posible arrancar su proceso principal (Figura 5.10) de forma que se mantiene a la espera de cualquier comando recibido desde una cola de mensajes dedicada para este objeto.

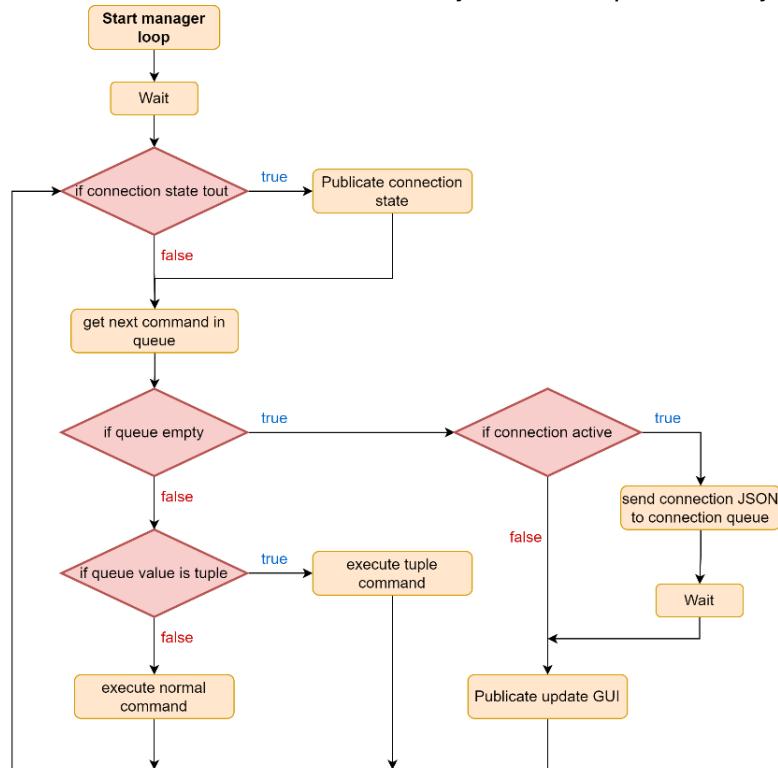


Figura 5.10 Proceso principal de mánager

En esta tarea de organización se utilizan 2 herramientas para hacer comunicaciones con el exterior:

- Publicaciones para ejecutar funciones específicas en el objeto GUI:
 - o Publicación de actualización de interfaz gráfica con valores recientes de robot
 - o Publicación de estado de ejecución de la aplicación
- Uso de cola de envío de comandos JSON para comunicación con aplicación de control. En este caso se envía comando de mantenimiento de conexión (5.5.1) hacia el objeto de conexión que se encarga de su posterior envío.

Por otro lado, la cola que se utiliza en el objeto mánager se encarga de recibir comandos específicos que van a desencadenar acciones que pueden afectar tanto al estado del mismo objeto mánager como a otros procesos. Las colas pueden manejar 2 tipos de comandos:

1. **Cola con estructura de valores:** estos son los que se reciben desde la GUI para realizar algún cambio en los demás procesos y objetos de la aplicación (Tabla 5.2).

Tabla 5.2 Comandos de Objeto mánager con estructura de datos

Nombre de comando	ID	Descripción
SAVE_NEW_IP	6001	Comando con la información almacenada de configuración red que se recibe desde el menú del objeto GUI dedicado para este propósito. Hace luego el

		guardado en el fichero de memoria y la actualización en los parámetros de conexión
NEW_POS	1005	Cuenta con las nuevas coordenadas, recibidas desde el objeto GUI, para ser enviadas al robot. Por ello desencadenan un envío de comando de control hacia el módulo SW de control
SET_ROBOT_PARAMS	6002	Comando que agrega nuevos valores de configuración en los parámetros del robot: <ul style="list-style-type: none"> - Velocidad de movimiento - Tipo de sujeción de SG Estos modifican al objeto robot y al fichero de memoria.
SET_CONNECTION_MODE	6003	Viene acompañado del tipo de conexión que se quiere realizar. Puede ser: <ul style="list-style-type: none"> - Simulación con CopeliaSim (5.4) - Conexión con MCU - Depuración de código
SET_COMMUNICATION_SETTINGS	6004	Este comando cuenta con los valores nuevos de configuración de esta comunicación en caso de ser activado el control desde el módulo GUI. Se puede configurar: <ul style="list-style-type: none"> - Tipo de control: por velocidad o por posición - Número de posiciones a enviar por paquetes. Así se evita un paquete muy largo de comunicación - Número de puntos a calcular por cm de desplazamiento. Esto se requiere para realizar el control de trayectoria lineal (5.6)

2. **Colas con comandos individuales:** estos no cuentan con valores nuevos para actualización, sino que simplemente efectúan un evento específico ante la recepción del comando (Tabla 5.3).

Tabla 5.3 Comandos de ejecución de funciones en objeto Mánager

Nombre de comando	ID	Descripción
RESET_MOTORS	1013	Ejecuta el envío de comando de reinicio para los motores especificados
SEND_PCKG	1010	Se encarga de sincronizar el envío de los comandos de control del robot con la disponibilidad de espacio en la recepción y manejo de este en la aplicación de control
GO_HOME	1004	Ejecuta el comando de inicialización de todos los eslabones del robot de forma de tener una calibración de la posición de todo el sistema
OPEN_GRIPPER	1008	Ejecuta el comando de apertura del gripper del robot
CLOSE_GRIPPER	1009	Ejecuta el comando de cierre del gripper del robot
TOGGLE_ROBOT_EN	1013	Alberna la habilitación de los motores para configurar las corrientes de funcionamiento principal (4.3.2.1)
CONNECT	2001	Alberna el estado de la conexión para controlar el socket del objeto de conexión

Con toda esta funcionalidad es posible hacer la organización de toda la aplicación y de tener una conexión asíncrona entre la interfaz gráfica y los demás objetos que conforman el

módulo GUI siempre que este último tengan conocimiento de estos comandos y de la cola que se usa en el objeto mánager.

5.2.2.5. OBJETO GUI

La GUI es la interfaz gráfica que va a ser visible para el usuario y desde la cual se van a introducir los datos de control y se van a monitorizar todas las señales existentes en el robot. Por esta razón las principales funcionalidades van a venir sujetas al uso de un módulo Python que cuente con las distintas herramientas visuales y de gestión de eventos necesarias para integrar este objeto. Es por lo que se ha optado por el uso del módulo gráfico Tkinter que funciona con un nivel más alto de abstracción de las librerías para aplicación GUI tcl/tk¹.

Con la librería Tkinter se tienen las siguientes ventajas:

- Creación de botones con generación de eventos que apunten a funciones
- Herramientas avanzadas de maquetación de interfaz
- Obtención de información de texto de entrada y posición de ratón en pantalla
- Creación de formas complicadas a partir de figuras básicas
- Inserción de imágenes
- Inserción de estilos de texto
- Creación de ventanas gráficas con posibilidad de anidación multinivel

Ya contando con el uso de una librería gráfica se pueden establecer los requisitos que debe incluir la ventana de interacción principal con el usuario (Figura 5.11). En cuanto a estos requerimientos se tienen:

- Inclusión de botones con funcionalidades específicas:
 - o Botón de conexión y desconexión red
 - o Botón de carga de valores ingresados de coordenadas para mandar al robot
 - o Botón de reinicio de motores de robot
 - o Botón de activación y desactivación de motores
 - o Botones de control de apertura y cierre de SG
 - o Botones de control de movimiento para robot móvil integrado con SCARA
- Inclusión de cuadros de texto para ingresar información:
 - o Cuadros de coordenadas destino que se quieren enviar al robot
 - o Cuadro de salida de valores de posiciones actuales de cada motor
 - o Cuadro de valores de los FSR presentes en SG
- Visualización de rango de cobertura de robot y toma de posición de control a partir de este plano
- Menú superior desplegable con múltiples funciones:
 - o Configuración de conexión: IP, puerto y tipo de protocolo IP
 - o Configuración de tipo de simulación:
 - SIM_ARDUINO: se configura comunicación UDP con placa Arduino para hacer comunicaciones de paquetes JSON (3.7.5)
 - SIM_COPPELIA: se configura conexión TCP para comunicación con SW CoppeliaSim Edu (5.4)
 - SIM_DEBUG: el código no se ve afectado por conexiones de desconexión e imprime en terminal todos los estados que está evaluando. Funcionalidad de depuración.

¹ www.tcl.tk

- Configuración de parámetros de robot generales:
 - Velocidad lineal
 - Nombre de robot
 - Tipo de control: posición o velocidad
 - Configuración de corrientes
- Mensaje de ayuda para explicación de interfaz
- Visualización de señales analógicas con leds de encendido:
 - Indicación de robot alimentado
 - Indicación de estado lógico de cada uno de los LS correspondiente a cada motor.

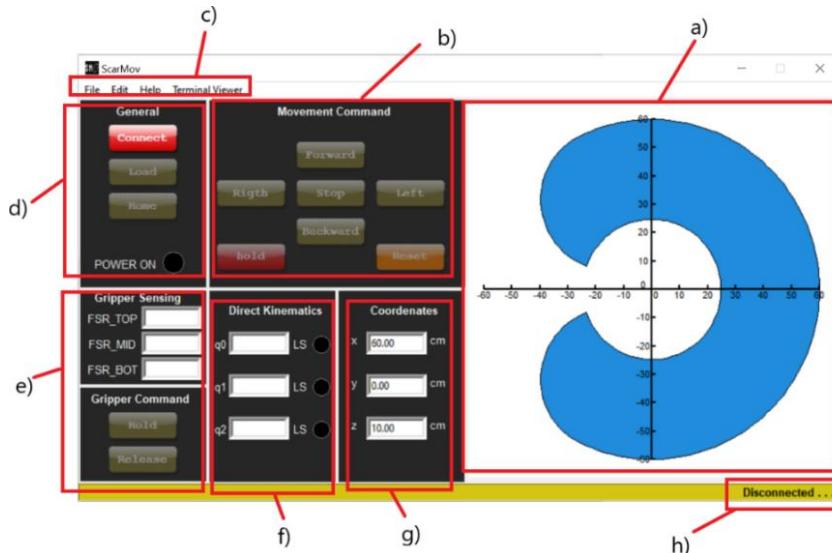


Figura 5.11 Estructura de Interfaz gráfica

En la imagen se muestran los distintos sectores que tiene la interfaz gráfica:

- a) Plano cartesiano con vista superior del robot SCARA para recepción de coordenadas
- b) Botones de control para movimientos básicos de robot móvil
- c) Menú con distintas funcionalidades de configuración de conexión y control del SCARA
- d) Control general de conexión con botones y señal de alimentación de robot
- e) Control de SG y lectura de sensores FSR
- f) Posición actual de los motores y señal de salida de sus correspondientes LS
- g) Entrada manual de coordenadas a enviar
- h) Control de estado de aplicación

Este objeto debe ser primero declarado (Figura 5.12) conociendo un objeto robot (5.2.2.3), un objeto conexión (5.2.2.1) y un objeto de administración de fichero (5.2.2.2). De esta forma se puede hacer el inicio de la interfaz gráfica. Posteriormente se debe llamar a la función *loop* del objeto GUI para mantener la ventana de interacción constantemente abierta y a la espera de peticiones programadas. Esta función debe llamarse de último a la inicialización de los demás procesos ya que es una función bloqueante y, hasta que no se cierre la pantalla, el código posterior a esta que se encuentre en el script no se ejecutará.

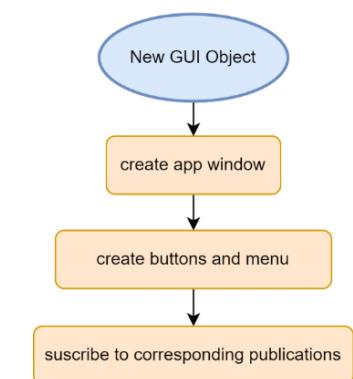


Figura 5.12 Inicio de objeto GUI

Una vez se hace uso de la función *loop*, el objeto GUI da soporte a 3 procesos (Figura 5.13) principales:

- Proceso de publicaciones:
 - o Mantener una espera continua de publicaciones provenientes de otros procesos que son recibidas como eventos. Existen las siguientes:
 - Publicación de actualización de valores en interfaz
 - Publicación de estado de ejecución actual de la aplicación
 - Publicación de actualización de estado de conexiones
 - o Ejecutar la correspondiente función suscrita con la publicación
- Mantener activa la ventana de usuario principal para que pueda recibir eventos
- Recibir los eventos realizados por el usuario y que provocan el envío de un comando, a través de una cola de datos, hacia el objeto mánager (5.2.2.4).

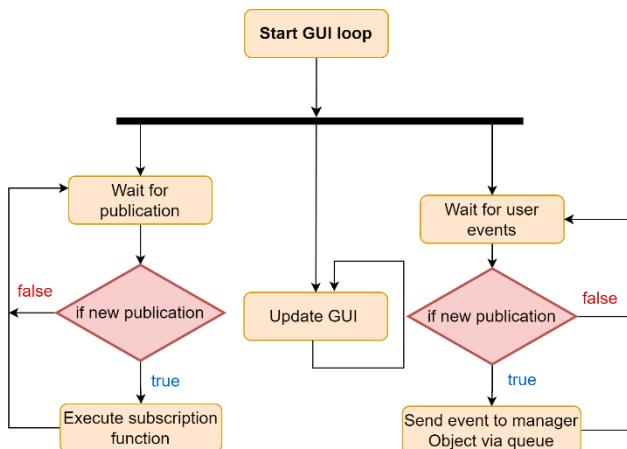


Figura 5.13 Diagrama de procesos activos con GUI loop

Finalmente, es importante resaltar que la interfaz gráfica, al igual que toda la aplicación SW de GUI, pretende servir como una herramienta escalable para distintos tipos de SCARA. Es decir, que a partir del establecimiento de una conexión con la aplicación de control se pueda conocer cuáles van a ser los requerimientos de la interfaz para aplicar los cambios necesarios correspondientes al robot conectado¹.

5.3. MÓDULO SW DE CONTROL

El segundo módulo SW va a ser el encargado de hacer el control del HW dedicado a los periféricos existentes en el robot (Figura 5.1). Es decir, que esta será la aplicación enfocada en incluir la funcionalidad específica de cada uno de los motores y periféricos. Por esta razón este es el código que será cargado en la UC (4.3.1) escogida.

Antes de poder hacer cualquier codificación es necesario cumplir ciertos requisitos en cuanto a las dependencias que tiene el módulo de control. Para cada una de sus partes fueron necesarias ciertas librerías (Tabla 5.4) de terceros que facilitan en gran medida el uso de estos

¹ Aplica únicamente con la comunicación a MCU

periféricos. Por esta razón, como se harán comunicaciones con los registros del CU, es evidente que este módulo SW va a contar con un lenguaje de menor nivel como es C y C++.

Tabla 5.4 Dependencias de módulo de control

Nombre	Proveedor	Descripción
WiFiNINA 1.8.13	Arduino	Librería para manejo de funciones de red WIFI soportadas por el módulo NINA-W102 [50]
SAMD Boards 1.8.11	Arduino	Librería de soporte de placas SAMD para contar con las funciones del SAMD21 [49]
Adafruit SSD1306 2.5.0	Adafruit ¹	Controlador de SSD1306 [54] para manejo de pantalla Oled (4.3.2.4)
Adafruit GFX Library ²	Adafruit	Librería para facilitar uso de controladores específicos de pantallas gráficas. Es usada en conjunto con Adafruit SSD1306 para implementar funciones visuales de alto nivel [59]
TMC5160	Proyecto anterior para control de SM	Manejo de funcionalidades de TMC5160 [51] a través de SPI para configuración completa de registros (Tabla 4.6) y monitorización.

Gracias a estas herramientas SW se pueden cubrir los requerimientos que debe incluir la arquitectura general del módulo de control (Figura 5.14). Entre ellas están:

- Lectura de señales analógicas y digitales
- Control de temporizadores de al menos 1 ms
- Control de interrupciones
- Mantenimiento de procesos secuenciales
- Manejo de interfaces seriales I2C, SPI y UART
- Uso personalizado de pantalla OLED
- Control de los motores con una interfaz sencilla

5.3.1. IMPLEMENTACIÓN DE ESQUEMA SECUENCIAL

A partir de estos requerimientos es importante resaltar el hecho de que se busca un funcionamiento secuencial (Figura 5.14) entre las tareas de la aplicación de control. Por esta razón se ha optado por el uso de 2 temporizadores³ disponibles en el SAMD21:

- Temporizador TC5 de 5ms para utilizar como servicio de temporización y control
- Temporizador TC3 de 1ms para manejo de tareas críticas y que requieren de mayor frecuencia de operación

De esta forma es posible crear un servicio de *tick* global que mantenga un conteo constante por ms y con un tamaño de palabra de 32 bits (50 días de temporización sin problemas

¹ www.adafruit.com

² Disponible en versiones superiores a Arduino IDE 1.8. Se puede encontrar en el siguiente repositorio: <https://github.com/adafruit/Adafruit-GFX-Library>

³ Fichero *timer.cpp*

de sobrecarga). Así se puede contar con un servicio de planificación de tareas¹ que se encarga de ejecutar cada proceso una vez su temporizador se ha vencido (Figura 5.14).

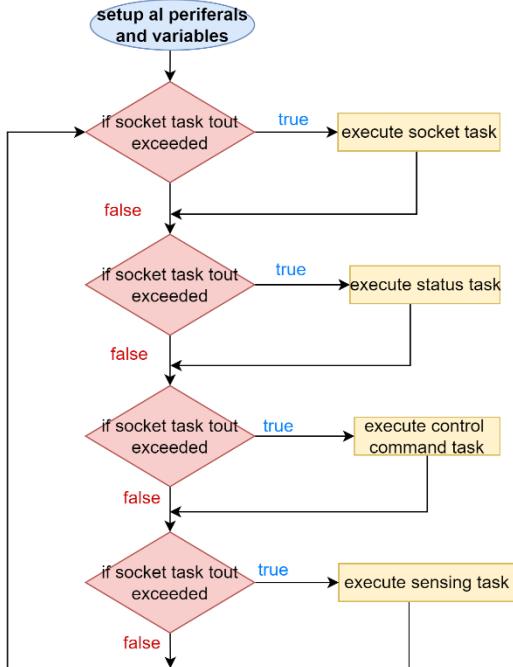


Figura 5.14 Bucle principal de módulo de control

5.3.2. CONFIGURACIÓN DE MÓDULO DE CONTROL

Para hacer modificaciones generales del módulo de control se puede utilizar el fichero *config_file.h* que cuenta con todos los valores de configuración necesarios para la implementación del código en el Arduino MKR 1010. Se pueden hacer los cambios de:

- Frecuencia de ejecución de los procesos principales
- Configuración de los parámetros red:
 - o IP
 - o Puerto
 - o Tiempo de espera de conexión
 - o Tamaño máximo de datos de entrada en bytes
 - o Nombre y credenciales de red. Se pueden incluir varias redes a las cuales se intentará establecer conexión
- Elección de pines analógicos y digitales: LS, NCS de motores, FSR
- Configuración de motores:
 - o Corriente de mantenimiento y de desplazamiento para cada motor
 - o *MicroStepping*
 - o Parámetros de relaciones con el sistema de poleas en articulaciones de revolución y avance por vuelta para articulaciones lineales
 - o Establecer el tipo de articulación para cada motor: lineales y de revolución
- Configuración de circuito FSR:
 - o Valor de resistencia R_M (Figura 3.7)
 - o Tipo de configuración de circuito: *PullUp*, *PullDown*

¹ El SAMD21 soporta un SO, como puede ser FreeRTOS. Sin embargo, limitaba el espacio disponible para el código de aplicación y, además, no se requieren herramientas de SO avanzadas.

- Configuración de macros usadas en el gestor de comandos
- Nivel de depuración para control de flujo de aplicación mediante impresiones en consola por puerto serial

Gracias a este fichero se puede tener de forma centralizada los valores más importantes que van a ser necesarios en la aplicación y que pueden ser modificados según sea conveniente de una forma rápida y sencilla.

5.3.3. SERVICIOS IMPLEMENTADOS

Para que los procesos principales del módulo cuenten con una funcionalidad completa hace falta incluir ciertas herramientas que permitan sincronización y comunicación entre ellos. Los casos en los que estas condiciones se necesitan son:

- Generación de eventos entre procesos distintos y que ejecuten funciones específicas
- Sincronización y gestión de cadenas de caracteres a ser enviados por el gestor de conexión UDP (5.3.4.2)
- Abstracción de comunicación con controladores TCM5160
- Abstracción de llamadas a funciones propias de MCU

Por estas razones se ha optado por la creación de código dedicado para cada una de esas tareas y que puede ser implementado de forma transparente a la aplicación. De esta forma se puede mantener la vigencia de la aplicación y se asegura que puede ser escalable en caso de ser necesario.

5.3.3.1. GESTOR DE TEXTO

Esta librería, que corresponde con los ficheros de nombre *frameManager*, se desarrolló para cumplir dos objetivos:

1. **Decodificación de cadenas de caracteres en formato JSON:** se ha implementado un algoritmo que cuenta con una máquina de estados capaz de hacer una lectura constante de uno o varios bytes que le sean cargados para verificar si se encuentra algún formato de texto plano que cumpla con las especificaciones de JSON (3.7.5). Una vez encuentra un nuevo paquete JSON correcto lanza un evento que puede ser capturado por cualquier otro proceso para saber cuándo se cuenta con un mensaje que posiblemente tenga información de interés para la aplicación (5.3.3.1).
2. **Gestión de cadena de caracteres para ser enviada por red:** cuenta con un buffer circular que puede ser cargado con una cadena de caracteres de longitud máxima establecida (5.3.2) para que, cuando el gestor de conexiones esté disponible y haya transcurrido la temporización de envío de estado de aplicación, el paquete

sea enviado a través del socket abierto hacia el cliente que se encuentre conectado (Figura 5.15).

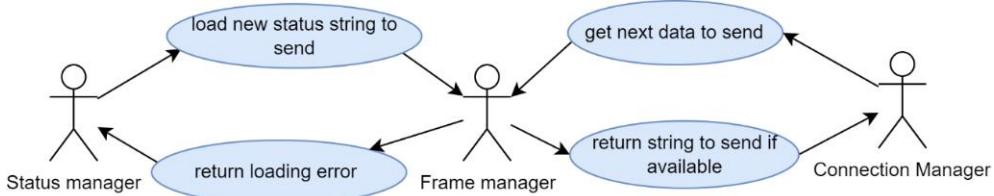


Figura 5.15 Caso de uso de gestor de texto para sincronización entre gestor de estados y gestor de conexión

5.3.3.2. INTERFAZ DE CONTROL DE MOTORES

Es una herramienta que sirve para llamar a las librerías de comunicación por SPI con el controlador TMC5160 (4.3.2.1). De esta forma se cuenta con un desacoplo entre el tipo de comunicación serial que se utilice y la aplicación de control desarrollada por el gestor de comandos (5.3.4.1). Entre las funciones que aporta este gestor se encuentra:

- Configuración general de *microStepping*
- Establecimiento de corrientes funcionales de cada motor
- Configuración de control de movimientos por posición o velocidad
- Lectura de registro de posición actual del motor. Este se puede utilizar como valor de realimentación del bucle de control (Figura 3.12)
- Verificación de alimentación en motores: para ella se utiliza un registro de o bits que cuente con el estado lógico de alimentación de cada motor. Ejemplos:
 - Alimentación = 0b00001111. Todos los motores alimentados
 - Alimentación = 0b00000010. Solo motor 1 alimentado
 - Alimentación = 0b00000110. Solo motor 1 y 2 alimentados

Esta librería está pensada para funcionar de forma escalable a medida que se agreguen más motores al sistema (siempre que puedan ser soportados por el Arduino). Estos parámetros deben ser coherentes dimensionalmente y estar definidos según lo especificado en el fichero de configuración (5.3.2).

5.3.3.3. INTERFAZ CON PANTALLA OLED

Se encarga de hacer las conexiones correspondientes con las librerías de comunicación I2C para el control de la pantalla oled (4.3.2.4). Cuenta con las siguientes funciones disponibles:

- Inicio de comunicación con pantalla
- Limpieza de pantalla
- Escritura de cadenas de caracteres
- Dibujo de línea e imágenes en formato de mapa de bits
- Desplazamiento de imágenes y texto en distintas direcciones

Esta interfaz va a servir para la salida de información de gran importancia a través de la pantalla OLED. Así, en caso de no haber comunicación serial, existe una salida de datos que pueden ser fundamentales para la conexión entre el módulo GUI y el módulo de control (por ejemplo, IP y puerto de conexión red).

5.3.3.4. INTERFAZ DE SENSORES

Estas librerías cuentan con la implementación específica con la que van a contar los sensores analógicos y digitales utilizados. En cada caso de monitorización de señales se va a contar con la posibilidad de configuración completa del número de sensores activos y cuales pines van a ser usados para dicho propósito (consultar hoja de características de CU para verificar posibilidad de uso de pines, 4.3.1) y por ende hace uso de la interfaz con MCU (5.3.3.5). Estas librerías se dividen en:

- **Librería FSR:** cuenta con la lógica del circuito utilizado para el sensor de presión (Figura 3.7). Implementa también todos los tramos evaluados (4.3.2.2) y probados para los modelos FSR07 de forma de contar con una medición más precisa. Para su configuración se pueden agregar identificadores que facilitan su lectura a partir de las funciones de interfaz (Por ejemplo, que los FSR_TOP tiene id = 0, configurable desde *config_file.h*).
- **Librería LS:** cuenta con las lecturas de cada sensor vinculado a su motor correspondiente. El estado del sensor puede ser accedido de forma individual con su identificador o en conjunto con la lectura del registro de 8 bits que cuenta con los estados de todos los sensores activados (desde el fichero de configuración, 5.3.2). El orden en este registro se efectúa igual que el de los motores, es decir, que el bit menos significativo corresponde al sensor cero y así sucesivamente (ejemplos se pueden ver con el registro de alimentación de motores, 5.3.3.2)

5.3.3.5. INTERFAZ CON MCU

Ya se ha explicado que se utilizan varias librerías elaboradas por Arduino que son fundamentales para la aplicación. Por esta razón se añadió una capa de abstracción que permite acceder a ciertas funciones de la librería de Arduino a través de un envoltorio o *wrapper* (Figura 5.16). Así se asegura que haya un mayor desacople entre las funciones propias del CU y la aplicación. En caso de cambiar de MCU se requeriría solamente de modificar este fichero de dependencias con Arduino y adecuarla al nuevo HW.

Entre las funcionalidades con que cuenta se tiene:

- Inicialización y control de comunicaciones seriales
- Lectura y escritura de pines digitales
- Lectura de pines analógicos
- Acceso a temporizadores

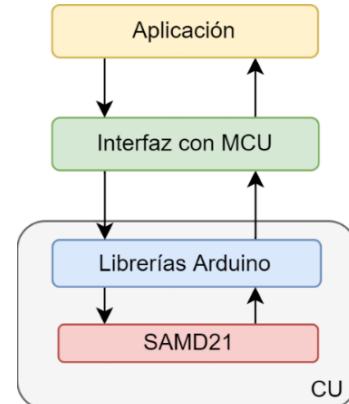


Figura 5.16 Esquema de capas de módulo SW con interfaz MCU

5.3.4. PROCESOS PRINCIPALES EN MÓDULO DE CONTROL

Se verán a continuación los 4 procesos que se ejecutan de manera secuencial gracias al esquema multiprocesos (Figura 5.14) implementado para la arquitectura del módulo de control en busca de cubrir los requerimientos especificados para esta.

5.3.4.1. GESTOR DE COMANDOS Y CONTROL

Como ya se ha visto (Figura 5.1) la aplicación SW requiere de un protocolo de comunicación entre la aplicación de GUI y la de control (5.5). Por esta razón es necesario una tarea encargada de interpretar los comandos recibidos (Figura 5.17) para manejar la información de control que se requiera y ejecutar la función correspondiente a este comando (Figura 5.21). Esta funcionalidad se encuentra en los ficheros *control_cmd* con el cual se puede realizar:

- Recepción de comandos recibidos por el gestor de texto al recibir el evento de JSON correctamente (5.3.3.1)
- Verificación de errores en comando
- Ejecución y control de buffer de recepción de comandos
- Estructura de datos enfocada en control de motores y gestión de buffer de comandos
- Interfaz de acceso a variables de control
- Uso de temporizadores para seguridad en ejecución de comandos

Es importante resaltar que esta tarea, debido a su importancia, debe de ejecutarse con la segunda mayor prioridad de la aplicación. Su frecuencia de ejecución no debe superar los 5 ms para que se pueda asegurar un muestreo completo y constante del proceso de control en todo momento para cada motor. Esto es posible ya que la comunicación con los motores se hace con SPI (4.3.2.1), lo cual permite hacer comunicaciones de alta velocidad y de manera prácticamente simultánea.

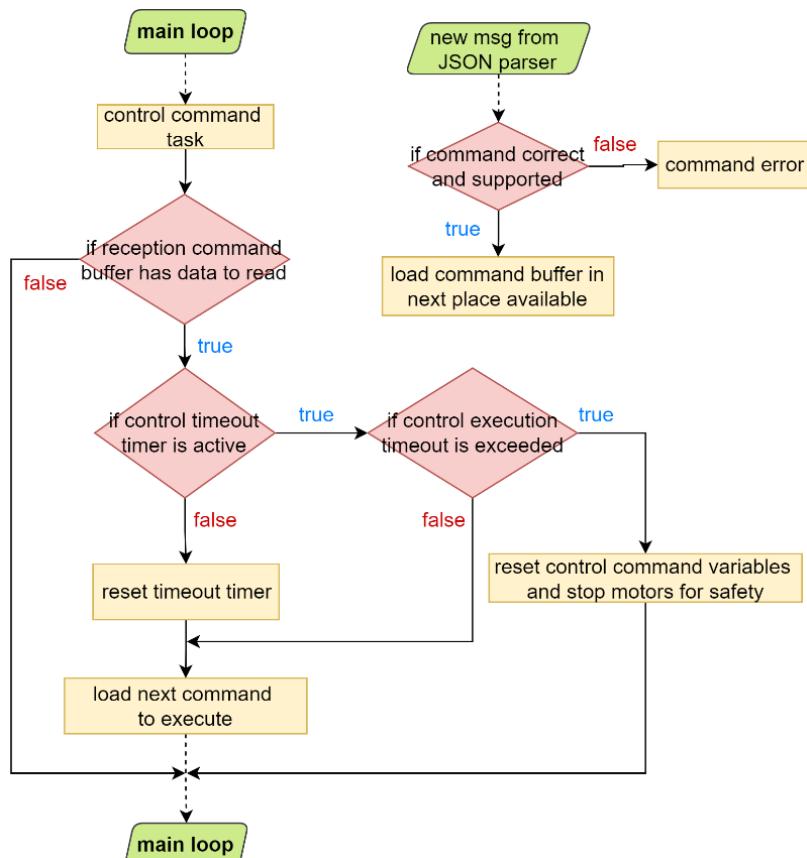


Figura 5.17 Bucle de ejecución para gestor de comandos y control en módulo SW de control

5.3.4.2. GESTOR DE SOCKET PARA CONEXIÓN RED IP/UDP

Este es el que se ejecuta con mayor prioridad debido a que es el responsable de efectuar la recepción y envío de paquetes UDP, es decir, de ser la capa 4 del modelo OSI (Figura 3.21) para la aplicación. En el esquema general de la aplicación de control (Figura 5.14) se incluye un temporizador para establecer su frecuencia de ejecución. Sin embargo, lo más recomendable es ejecutar este proceso con la velocidad de ejecución del bucle principal de la aplicación.

Para este proceso se utilizó la librería WiFiNINA para poder tener la comunicación con el módulo red NINA-W102. A pesar de que esta librería permite montaje de servidores web y comunicaciones con protocolo TCP se ha optado por la implementación del proceso de conexión con el protocolo UDP (Figura 5.18). Esto se debe a que, como ya se explicó (3.7.3), este protocolo ofrece una mayor facilidad en el manejo de la comunicación y, además, cuenta con un menor tamaño de paquete. Esto se traduce en una reducción en errores de comunicación y una mayor rapidez en la transmisión de comandos entre el módulo GUI y el módulo de control.

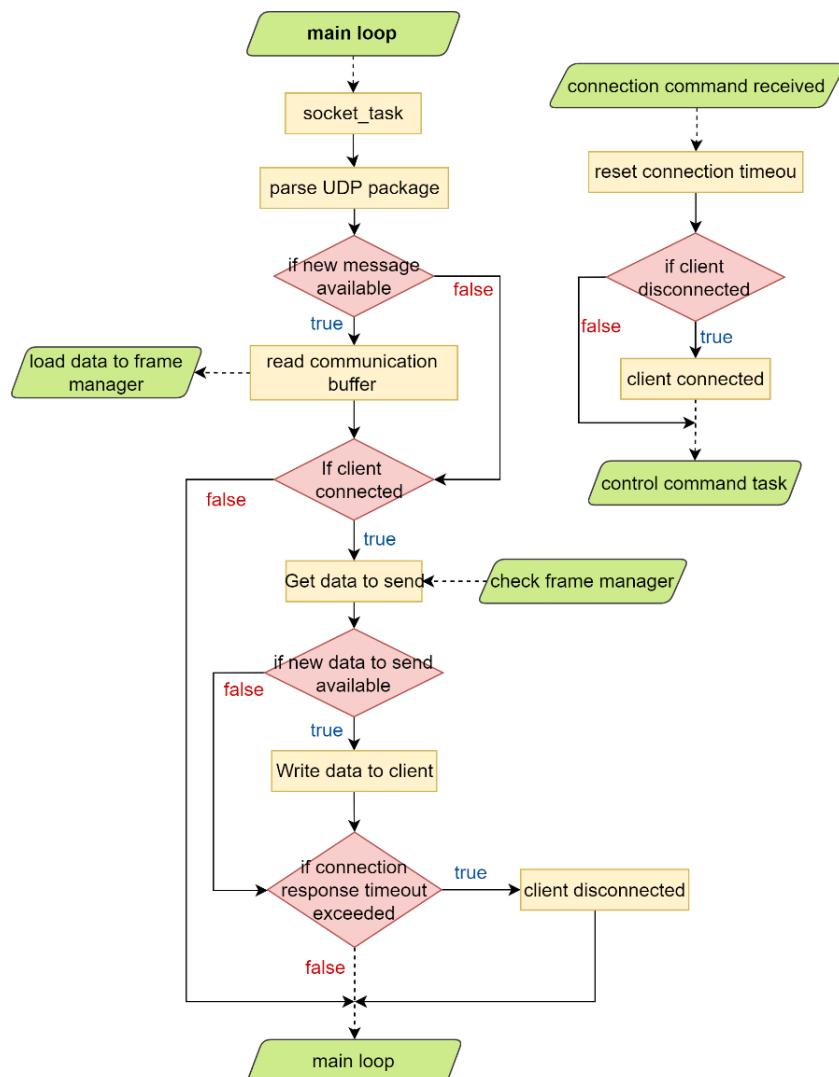


Figura 5.18 Proceso de gestión UDP en módulo SW de control

En verde están representados los eventos generados por o hacia procesos

Una vez inicializada la apertura del socket para datagramas (3.7.4) el proceso cumple las tareas correspondientes al esquema de conexión servidor-cliente para comunicaciones UDP (Figura 3.22). Estas son:

- Decodificación de paquetes UDP recibidos para luego ser enviados al gestor de texto (5.3.3.1)
- Codificación UDP de cadenas de caracteres para ser enviados al cliente
- Gestión de estados de conexión en base a tiempo de espera de recepción de comando de conexión por parte del proceso de control de comandos (5.3.4.1)

5.3.4.3. GESTOR DE ESTADO DE LA APLICACIÓN

Este proceso (Figura 5.19) se encarga de hacer una lectura de los últimos valores registrados en la aplicación y que sean de interés para el cliente que esté conectado. Entre estos datos se encuentran:

- **Presión ejercida en cada uno de los FSR:** existe un sensor FSR para cada SG_FLX ubicados en posiciones distintas: nivel superior, nivel medio y nivel inferior. Así se asegura que se cuente con la lectura de la fuerza ejercida en todos los puntos de apoyo que se hagan sobre el objeto sujetado siempre que este cumpla con las propiedades de sujeción del SG (Figura 3.5).
- **Lectura los valores lógicos de los LS de cada eslabón:** se almacena en un registro de 8 bits el valor lógico de cada LS (4.3.2.3). Este valor depende del número de LS activados en la aplicación y que están relacionados con el motor correspondiente a su límite de rango (5.3.2).
- **Posición de los motores:** se hace la lectura del registro correspondiente a la posición actual de los motores (Tabla 4.6). El número de valores registrados se puede escalar en función del número de motores configurados (5.3.2).
- **Señales de estado de control:** se envía el estado actual del ciclo de control (5.3.4.1) que se esté efectuando para que no haya saturación de comandos. Entre las más importantes:
 - o Estado lógico de ejecución de comandos
 - o Estado lógico de disponibilidad de decodificador JSON
- **Alimentación de los motores:** a pesar de que el Arduino pueda estar conectado con la aplicación GUI es posible que los motores no estén alimentados (4.3.2.5). Por esta razón se evalúa este estado de los motores (5.3.3.2) para tener un valor lógico registrado que sirva para saber, desde la interfaz gráfica, que es necesario alimentar los motores y realizar o bloquear las acciones de control.

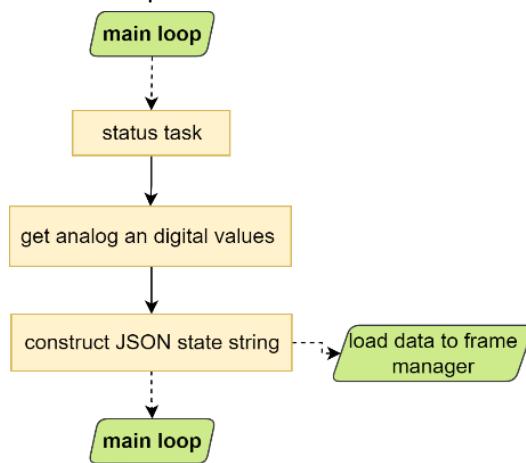


Figura 5.19 Proceso de control de estados de módulo de control

Una vez realizada la lectura de todos los valores lógicos y analógicos se procede a la codificación en formato JSON del mensaje de estado (Figura 5.20). Posteriormente se hace el envío de la cadena de caracteres por la red de comunicación UDP con ayuda del gestor de texto (Figura 5.15).

```
{
    "LS": 2,
    "FSR_TOP": 4,
    "FSR_MID": 2,
    "FSR_BOT": 0,
    "CMD_BUFF_FULL": 0,
    "CMD_STATE": 0,
    "JSON_PARSER": 1,
    "MOT_POWER": 3
}
```

En el mensaje de estado se muestra:

- LS: motor 1 ha llegado a su rango máximo (0b00000010)*
- FSR_TOP: en la parte superior del objeto sujetado se ejercen 4 N*
- FSR_MID: en la parte media del objeto sujetado se ejercen 2 N*
- FSR_BOT: en la parte baja del objeto sujetado no se ejerce fuerza*
- CMD_BUFF_FULL: buffer de gestor de comandos está lleno*
- CMD_STATE: no existe comando en ejecución*
- JSON_PARSER: está ocupado el decodificador JSON*
- MOT_POWER: los motores 0,1,2 están alimentados*

5.3.4.4. GESTOR DE SENSORES

Para las principales funcionalidades de control y de HMI es necesario tener la capacidad de recibir los valores analógicos y digitales de los sensores que estén conectados en el SCARA (LS y FSR). Por esta razón es que este proceso (Figura 5.21) va a estar enfocado en la actualización de los últimos valores que se encuentren en las entradas analógicas o digitales con el uso de las interfaces de los sensores (5.3.3.4) para que se actualicen sus estados.

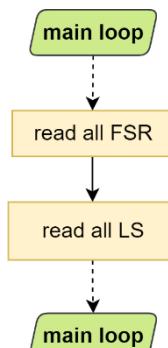


Figura 5.21 Proceso de lectura de sensores en módulo SW de control

5.4. SIMULACIÓN CON ROBOT MÓVIL

Además del robot físico, el módulo GUI también soporta la comunicación con un SW de simulación que va a ser de gran ayuda para probar la cinemática inversa y directa del robot (3.2) gracias al uso del objeto de conexión que ofrece el módulo GUI. De esta forma se pueden hacer pruebas de comandos para control del robot móvil y evaluar el comportamiento en conjunto con el brazo robótico que sirva para futuras implementaciones. Por esta razón se ha optado por el

uso de CoppeliaSim V4.3.0 [60], de Coppelia Robotics¹, debido a que es un SW de libre acceso que permite hacer simulaciones, enfocadas al control de máquinas, bastante avanzada. Cuenta con las siguientes ventajas:

- Interfaz sencilla para uso
- Modelos de robots y manipuladores listos para su uso en simulación
- Herramientas para creación de robots de manera personalizada
- Posibilidad de comunicación vía red IP
- Lenguaje de programación de alto nivel de fácil implementación (LUA [60])
- Configuración de parámetros físicos y estructurales para simulaciones ajustadas a la realidad

Es decir, que es posible incluir diseños personalizados y a escala del robot real que se ha desarrollado con sus características físicas y funcionales. A partir de [61] se ha podido seguir el proceso de construcción de un robot articulado básico que sirve para hacer controles de movimiento.

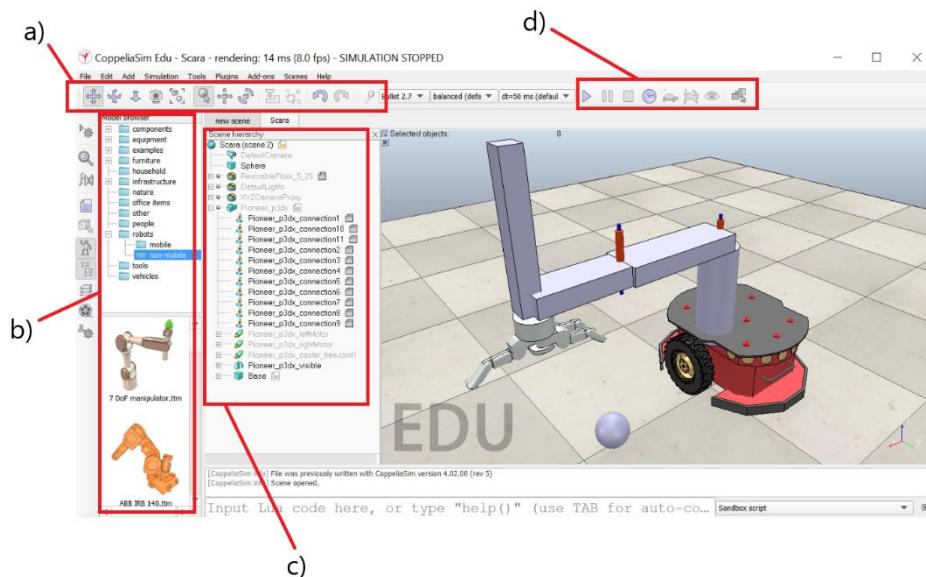


Figura 5.22 Interfaz de CoppeliaSim y modelo de SCARA

- a) *Configuraciones visuales y de modelo 3D*
- b) *Librerías con modelos de robots y grippers*
- c) *Eslabones en diseño*
- d) *Control de simulación*

Posteriormente se incluyó un script encargado de hacer la apertura de un socket local en la máquina donde se esté ejecutando la simulación y configurada en un puerto específico [60]. Esto se logra con un lenguaje de alto nivel llamado LUA que permite hacer estas implementaciones de forma sencilla y también personalizar las funcionalidades de modelos de robots ya creados.

Una vez iniciada una simulación se puede, desde la interfaz gráfica (5.2.2.5), enviar comandos propios de la API de coppeliaSim [60] mediante el uso de la librería específica para

¹ www.coppeliarobotics.com

Python [60]. De esta forma es posible controlar todas las funcionalidades del robot y hacer las pruebas que se requerían de control de movimiento.

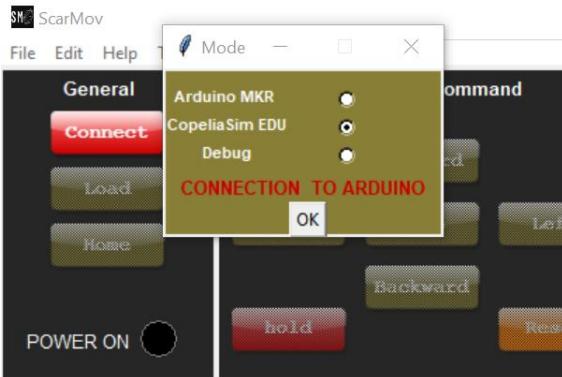


Figura 5.23 Opciones de Simulación en interfaz de aplicación SW

5.5. PROTOCOLO DE COMUNICACIÓN ENTRE MÓDULOS SW

Para que exista comunicación entre los dos módulos de SW es necesario contar con una estructura de mensajes que permita el entendimiento entre el servidor, que es el módulo de control, y el cliente, que sería el módulo GUI. Ya se ha explicado que ambos módulos están comunicados por UDP para cubrir la cuarta capa del modelo OSI (3.7.2). Por ello hace falta cubrir el nivel siguiente para establecer una comunicación que pueda ser entendida por ambas partes.

5.5.1. COMANDOS JSON

Se han creado una serie de comandos que ejecutan ciertas funcionalidades en el modelo de control. Estos comandos van a ser enviados desde la interfaz de usuario hacia el módulo de control, por ende, necesitan una estructura que permita su correspondiente interpretación. Es por lo que se ha elegido una estructura en formato JSON (3.7.5.1) que cuenta con los siguientes atributos:

1. **CMD**: código de comando para identificación de función a ejecutar por gestor de comandos. Hay algunos que requieren de otros atributos para su ejecución. Entre los que se usan están:
 - o RESET_MOTORS: ejecuta la configuración de inicio de la interfaz de motores para establecer valores iniciales del motor a controlar. Requiere solamente de los identificadores (MOT_ID) de motores a reiniciar.
 - o VELOCITY_CMD: lanza el control por velocidad (4.3.2.1) de los motores para alcanzar la posición correspondiente. Requiere del valor de la posición objetivo (POS) y la velocidad lineal (VEL) a la que se debe desplazar el robot.
 - o POSITION_CMD: lanza el control por posición (4.3.2.1) de los motores para alcanzar la posición correspondiente. Requiere del valor de la posición objetivo (POS) y la velocidad lineal (VEL) a la que se debe desplazar el robot.

- CONTROL_ARRAY: ejecuta el control por posiciones de motores a partir de valores que sean recibidos en el comando. Requiere de los identificadores de los motores (MOT_ID) a mover y los valores de posición (POS_ARR) y velocidad (VEL_ARR) que hacen la trayectoria de movimiento.
 - EMER_STOP: efectúa una parada completa del motor que sea indicado. Requiere de los identificadores de motores (MOT_ID) a detener.
 - SET_HOME: efectúa el envío al home del motor que sea indicado (MOT_ID).
 - CONNECTED: este comando debe ser enviado desde el cliente para mantener la conexión UDP activa. Por ende, lanza un evento que refresca el tiempo de vencimiento de cliente activo en el gestor de conexión (Figura 5.18).
 - MOTOR_DISABLE: deshabilita la corriente de alimentación del motor. Requiere del identificador del motor (MOT_ID).
 - MOTOR_ENABLE: deshabilita la corriente de alimentación del motor. Requiere del identificador del motor (MOT_ID).
2. **MOT_ID:** Identificadores de motores sobre los que se efectuará el comando en caso de que este lo necesite. Estos irán en concordancia con la configuración establecida para el control de los motores (5.3.4.1). En caso de hacer control de más de un motor debe ser indicado el valor de este atributo en formato JSON para vector de valores (3.7.5).
 3. **POS:** posición final a la que se quiere enviar el robot. Su formato es un vector con los 3 valores $[x, y, z]$ en unidades en décimas de milímetros $\left[\frac{mm}{10}\right]$ ($1 = 0,1\text{mm}$) para aumentar resolución. Aplica para los comandos POSITION_CMD y VELOCITY_CMD.
 4. **VEL:** es la velocidad lineal medida en $\left[\frac{mm}{110s}\right]$ y que va a afectar el control de movimiento de los comandos POSITION_CMD y VELOCITY_CMD.
 5. **POS_ARR:** es un array de valores correspondiente a las posiciones (en centésimas de grados para articulaciones angulares y en décimas de milímetro para articulaciones lineales) que deben ser establecidas en el comando CONTROL_ARRAY y que serán enviadas de manera secuencial según orden en vector hacia el motor que se indique. En caso de tener más de un motor a controlar deben ordenarse en un mismo vector y establecer el mismo orden de posición en el que se indiquen los identificadores de motor (MOT_ID). Un ejemplo de tener 2 motores a controlar con MOT_ID: [0,1] se ve en Ecuación 5.1.

$$[p_0^1, p_0^2, p_1^1, p_1^2, \dots, p_n^1, p_n^2]$$

Ecuación 5.1

6. **VEL_ARR:** array que contiene las velocidades en centésimas de grados por segundo que debe tener cada motor para desplazarse entre las posiciones de POS_ARR. Cuenta con el mismo formato de organización visto en Ecuación 5.1.

Como ya se ha mencionado, lo más ideal para el control de los motores es que los cálculos de trayectoria (5.6.1.1) sean efectuados en la CU ya que es la que cuenta con un acceso directo a los motores y, por otro lado, se evita el uso de enviar numerosos valores de posición y velocidad (Figura 5.24) en el paquete de envío que pueden requerir mayor trabajo en el SW. Así se acelera

el proceso de envío de datos UDP que es en donde se puede presentar una mayor latencia de envío.

```
{
    "CMD": 5002,
    "POS": [6050,5120,2500],
    "VEL": 500
}
{
    "CMD": 5006,
    "MOT_ID": [0,2],
}
```

Figura 5.24 Comando JSON de control

- El comando superior hace referencia al comando POSITION_CMD. Cuenta con la posición a dirigir el robot [60.5,51.2,25.00] cm y la velocidad lineal de 5cm/s. Nótese los pocos bytes que requiere el paquete a diferencia de enviar un array con posiciones y otro con velocidades (CONTROL_ARRAY).

- En el comando inferior hace un Homing de los eslabones (o motores) con identificación 0 y 2

5.6. MOVIMIENTO DEL ROBOT

Al momento de enviar la posición a la cual dirigirse se debe establecer una trayectoria a seguir por el robot. Se pueden hacer optimizaciones de las trayectorias en función del requerimiento que se busque. En este caso de aplicación se busca rapidez, y por ello, la distancia más corta que exista entre la posición actual y la final. Para este propósito se verá la matemática que se ha incluido en el cálculo de esta trayectoria y cómo es que se controla su envío a los motores.

El movimiento del robot se puede efectuar desde cualquiera de los dos módulos del SW. Una vez establecida esta elección se realizarán los siguientes algoritmos correspondientes al control de la trayectoria del robot.

5.6.1. CÁLCULO DE TRAYECTORIA LINEAL

Para el cálculo de la trayectoria a seguir se dividirán dos posibles casos:

- Movimiento de eslabones ESB1 y ESB2 en conjunto para desplazamientos en el plano XY
- Movimiento lineal de ESB3 para desplazamientos lineales en el eje Z

5.6.1.1. TRAYECTORIA EN XY

En cuanto al primer caso se establece que la trayectoria que se va a seguir será una línea recta (Figura 5.25). La longitud de la trayectoria se puede calcular fácilmente con la ecuación de distancia entre dos puntos como se ve en Ecuación 5.2.

$$d = \sqrt{(p1_x - p2_x)^2 + (p1_y - p2_y)^2}$$

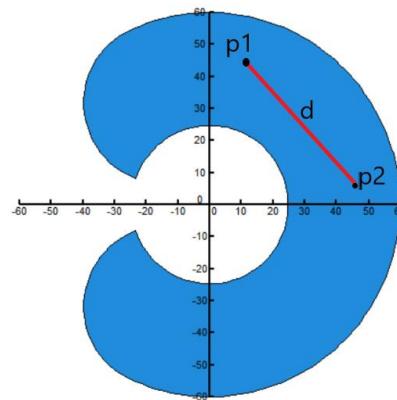


Figura 5.25 Desplazamiento de p1 a p2 en línea recta

Una vez se sabe la distancia a recorrer se divide en fragmentos de puntos de desplazamiento para que el movimiento sea lo más fluido posible. Sin embargo, esto requerirá de un mayor número de comandos enviados a los motores.

Finalmente, se deben utilizar las expresiones Ecuación 3.18 y Ecuación 3.19 en cada uno de los puntos calculados del tramo de movimiento para hacer el cálculo del correspondiente ángulo al que cada motor debe desplazarse siguiendo esta línea recta. Sin embargo, existen casos en los que esta trayectoria debe cambiar para efectuar un movimiento completo:

- La trayectoria entre los puntos de desplazamiento pasa a través de la circunferencia límite central (Figura 5.26): en este caso el robot se frena en la circunferencia central y la recorre hasta hacer el movimiento suficiente que permita seguir un movimiento lineal hasta la posición objetivo.
- El punto objetivo se puede llegar con lógica de eslabones invertida: como ya se ha visto existen siempre 2 posibilidades de posicionamiento de los eslabones del primer caso de trayectoria (ver Ecuación 3.19). Por esta razón es posible que exista un movimiento al cual se pueda llegar de forma matemática pero que excede los límites de rango de ESB1. En este caso se opta por posicionar ambos motores en 0° e invertir la lógica de cálculo que se estaba utilizando. Una vez llegado a este punto se sigue la trayectoria recta hasta la posición final deseada y se mantiene la lógica invertida siempre que no se repita esta casuística.

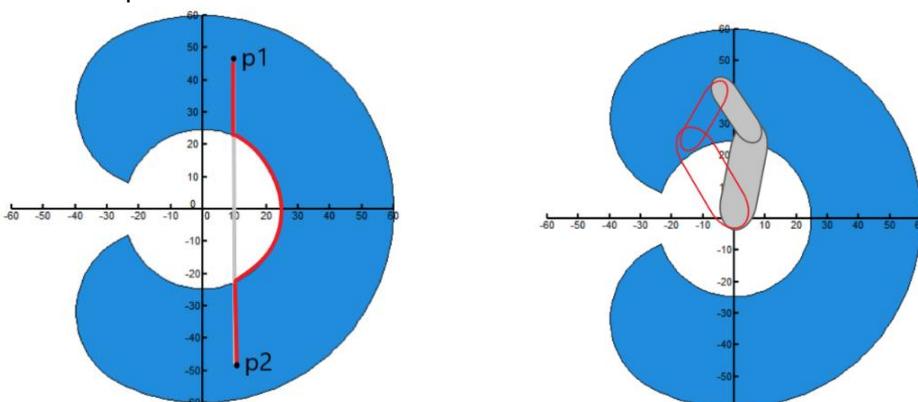


Figura 5.26 Casos especiales de trayectoria lineal

A la izquierda se ve la trayectoria del primer caso especial que sigue la circunferencia límite interna. A la derecha se ve el rojo la posición que calcula la cinemática inversa si no se realiza el cambio lógico y con la cual se pasaría el rango máximo de ESB1

5.6.2. CONTROLADOR DE POSICIÓN

Conociendo las posiciones a enviar para seguir una trayectoria y habiendo estudiado el modelo de control que debe utilizarse en un motor (Figura 3.12) se puede hacer una implementación SW que realice, en un dominio discreto, el control de la posición de cada motor que se esté utilizando.

Para el control de los motores se cuenta con:

- Posición del motor: esta es recibida desde el controlador TMC5160 (4.3.2.1)a partir del registro X_ACTUAL (Tabla 4.6)
- Señal de control: esta va a ser la posición a la que queremos llegar y que es enviada al controlador de motores para efectuar el movimiento.
- Error: se muestrea constantemente la posición objetivo y se compara con la actual para obtener el error. Este dato va a servir para realizar las aceleraciones y frenados de los motores y contar con un movimiento más fluido.
- Controlador tipo proporcional derivativo (PD): la acción de control se hace en función de la derivada del error¹ presentado. Este efecto de control actuará en 2 umbrales:
 - o Aceleración: la acción de control actúa hasta llegar a la velocidad objetivo
 - o Frenado: luego de que el error pase un valor umbral configurado y tenga una pendiente negativa significa que estamos llegando al SP y, por ende, se efectúa el control de frenado

Cabe resaltar que la acción de control es simplificada en gran medida con el uso de los TMC5160. Esto se debe a que con este módulo podemos desplazar el motor a cualquier posición con resoluciones de hasta 0,007° si utilizamos la máxima capacidad de *microStepping*. Entonces este controlador SW de posición va enfocado a asegurar que el movimiento completo del motor sea fluido y no se presenten problemas de oscilaciones debido a la inercia que pueda tener los eslabones.

¹ Conocida también como velocidad de error

Capítulo 6. RESULTADOS DE HW Y SW DE SCARA

En este capítulo se verán los resultados obtenidos a partir de los desarrollos vistos en capítulos anteriores en cuanto a la fabricación y montaje del HW del SCARA (Capítulo 4) y el funcionamiento de la aplicación SW con sus dos módulos integrados (Capítulo 5). Además, se evalúa el funcionamiento de los componentes electrónicos por separados para comprobar que todo funcione correctamente antes de hacer cualquier verificación de movimientos del robot completo. Finalmente, se evaluará el control del SCARA mediante pruebas de velocidad y posicionamiento enviados desde la interfaz gráfica y, por otro lado, la actuación del SG mediante pruebas de sujeción con distintos objetos para comprobar sus capacidades y limitaciones.

6.1. HW FABRICADO Y MONTAJE

Se verán las piezas impresas y montadas por separado para al final formar el cuerpo completo del SCARA y así poder hacer las posteriores pruebas de funcionamiento. Cada una de las piezas que serán mostradas se encuentran, como ya se ha explicado, en formato STL y DFX según la tecnología de fabricación (4.2). Dichos ficheros (Tabla 6.1) cuentan con el mismo nombre identificativo de pieza que se ha utilizado en el desarrollo de este documento (4.1).

Tabla 6.1 Ficheros para fabricación e información adicional

Módulo	Referencia	Extensión de fichero	Tecnología de fabricación	Duración (h)	Material	Imagen
SG	SG_BS	STL	FDM	4	PLA	Figura 6.16
	SG_BS_ACT	STL	FDM	2	PLA	Figura 6.16
	SG_FLX	STL	FDM	4.20	TPU	Figura 6.15
BS	BS_TOP_MOT2	STL	FDM	10.3	PLA	Figura 6.4
	BS_TOP_COV	STL	FDM	17.3	PLA	Figura 6.5
	BS_TOP_TENS/ BS_BOT_TENS	STL	FDM	0.2	PLA	-
	BS_BOT_COV	STL	FDM	15.25	PLA	Figura 6.5
	BS_BOT_MOT1	STL	FDM	8.11	PLA	Figura 6.1
	BS_PULL_P62	STL	FDM	8	PLA	Figura 6.5
	BS_PULL_P20	STL	FDM	4.2	PLA	Figura 6.5
ESB1	BS_PULL_P27	STL	FDM	3.5	PLA	Figura 6.2
	ESB1_SJBS_TOP	STL	FDM	4.3	PLA	Figura 6.6
	ESB1_SJBS_PUL62	STL	FDM	7.5	PLA	Figura 6.6
	ESB1_PRC_TOP	DXF	LASER	0.5	PMMA	Figura 6.6
	ESB1_PRC_BOT	DXF	LASER	0.75	PMMA	Figura 6.6
	ESB1_SJ2_TOP	STL	FDM	4.2	PLA	Figura 6.6
ESB2	ESB1_SJ2_BOT	STL	FDM	4	PLA	Figura 6.6
ESB2	ESB2_SJ1_PUL21	STL	FDM	4.3	PLA	Figura 6.8

	ESB2_SJ1_TOP	STL	FDM	11.3	PLA	Figura 6.7
	ESB2_SJ1_BOT	STL	FDM	16.5	PLA	Figura 6.9
	ESB2_SJ3	STL	FDM	15.3	PLA	Figura 6.9
ESB3	ESB3_MOT3	STL	FDM	2.5	PLA	Figura 6.10
	ESB3_MGRP	STL	FDM	8.5	PLA	Figura 6.11
PCBOX	PCBOX_TOP	STL	FDM	4.2	PLA	Figura 6.14
	PCBOX_BOT	STL	FDM	2.1	PLA	Figura 6.13

6.1.1. MÓDULO BS

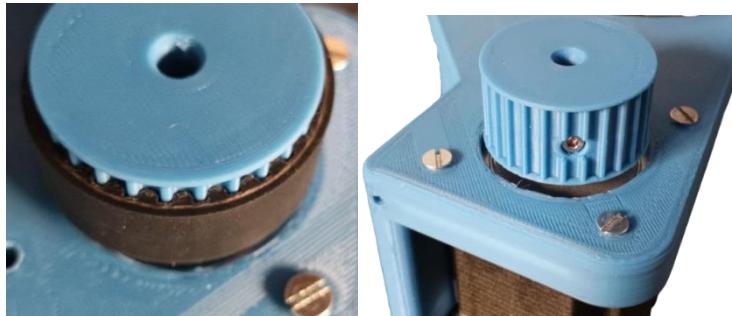
A partir del desarrollo de este módulo (4.1.4.1) se sabe que para su montaje es necesario:

- 8 tornillos M3 de al menos 15 mm de largo con sus tuercas para sujeción de motores
- 6 tornillos M3 de 12 mm de largo con sus tuercas para sujeción de BS_PULL_P27 con los motores
- 6 tornillos M3 de 20 mm de largo con sus tuercas para unión de coberturas de la base a los soportes de motores
- Un sensor LS con 2 tornillos M3 para unión con pieza
- 6 tornillos M3 de 20 mm de largo para unión de parte superior con inferior
- 2 correas T5 de 15 mm de ancho y 500 mm de largo
- Un rodamiento 6805-2RS para pieza BS_TOP_MOT2
- 2 rodamientos 608-2RS para incluir en las piezas de cobertura y que servirán para sujetar la barra de aluminio
- Barra de aluminio sólida de 125 mm de largo y 8 mm de diámetro que irá por el interior de las dos piezas de poleas que conforman el sistema de transmisión para ESB2
- 2 motores NEMA 23



**Figura 6.1 Pieza impresa
BS_BOT_MOT1**

Se ve en las 2 imágenes la pieza que se encarga de sujetar el motor 1 con la configuración de cables pasando a través del canal específico para ello. También se puede ver la sujeción con tornillos M4 del motor y las entradas de las tuercas internas para el sistema de sujeción.



**Figura 6.2 Pieza impresas
BS_PULL_P27**

Se puede ver la pieza unida al eje del motor 2 con el uso del sistema de tuercas internas. En la imagen izquierda se verifica que el ajuste para correas T5 sea el correcto.



**Figura 6.4 Pieza impresa
BS_TOP_MOT2**

Se puede ver la pieza encargada de la sujeción del motor 2 que va en la base del sistema. También, se incluye el motor utilizado para este propósito sujetado con los tornillos M4 y las entradas correspondientes a la sujeción de las dos partes del módulo base. Además, se ve el rodamiento de soporte de BS_PULL



**Figura 6.5 Piezas impresa de
cobertura de BS**

A la izquierda se puede ver la cobertura inferior del módulo base BS_BOT_COV. A la derecha se tiene la cobertura superior BS_TOP_COV del mismo módulo. En ambos casos se puede ver la entrada en los laterales para tornillos M3 y, además, los correspondientes rodamientos para sujeción



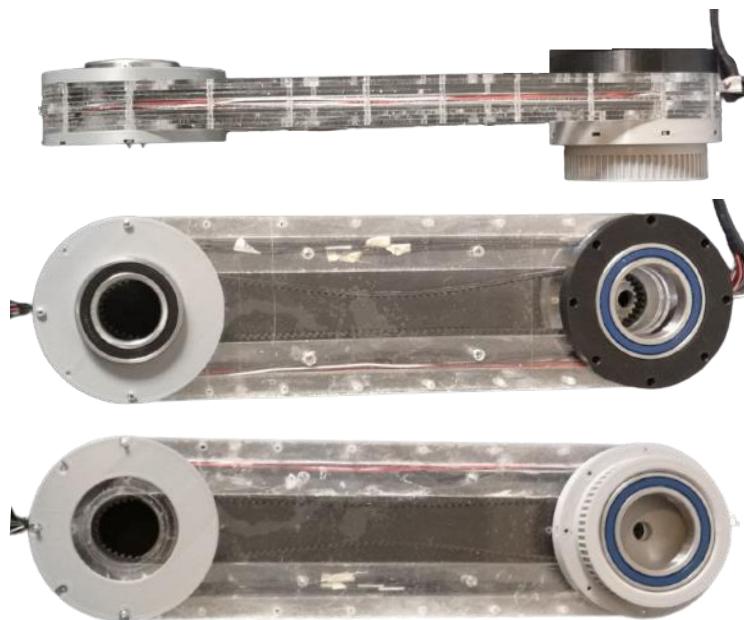
**Figura 6.3 Sistema de poleas
impreso de BS**

Se muestran piezas que conforman el sistema de poleas que va a transmitir la fuerza de tensión provista por el motor 2. Se encuentran en la imagen izquierda de forma separada. Y en la derecha se muestran unidas y con la barra de aluminio pasando por su interior para aportar rigidez y estabilidad a la articulación de la base.

6.1.2. MÓDULO ESB1

Con base en el desarrollo de diseño de este módulo (4.1.4.1) se necesita para su ensamblaje, luego de haberse pegado con acetona las planchas correspondientes, los siguientes componentes:

- 6 tornillos M3 de 45 mm de largo con sus correspondientes tuercas para la unión de todas las piezas del módulo
- 4 tornillos M3 de 30 mm de largo para asegurar el cableado interno de la pieza
- Una correa de T5 de 800 mm de largo
- 2 rodamientos 6010-2RS para las piezas de sujeción de ESB1 con BS
- 2 rodamientos 6008-2RS para piezas de sujeción con ESB2



**Figura 6.6 Módulo construido
ESB1**

En la imagen superior se aprecia una vista lateral del módulo ESB1 donde se ven todas las partes que lo conforman. En la imagen del medio se tiene una vista superior del módulo y se aprecia el sistema de sujeción en sus extremos. Finalmente, en la imagen inferior se puede ver la versión homóloga pero inferior del sistema de sujeción en los extremos del módulo.

6.1.3. MÓDULO ESB2

En función de lo anteriormente mencionado en la fase de desarrollo (4.1.4.3) se sabe que para tener el montaje de este módulo hacen falta los siguientes componentes mecánicos:

- 3 tornillos M3 de 20 mm de largo con sus tuercas para unión de pieza superior con pieza ESB2_SJ1_BOT
- 2 tornillos de 40 mm de largo M3 con sus tuercas para unión de ESB2_SJ1_BOT con ESB2_SJ3
- 4 guías lineales IGUS para incluir en ESB2_SJ3
- 1 tuerca para tornillo de avance de 4 hilos y de paso 8 mm
- 2 sensor LS para detección de final de carrera de ESB2 y ESB3
- 2 tornillos M3 de 10 mm con sus tuercas para sujeción de LS de ESB2
- 2 tornillos M3 de 5 mm para sujeción de LS de ESB3 (se enrosca con el plástico)



Figura 6.7 Pieza impresa ESB2_SJ1_TOP

Se puede ver en la imagen la pieza inferior se sujeción con el primer eslabón y cuenta con la entrada hexagonal para ESB2_SJ1_PUL21. Se ven, también, los agujeros de los tornillos M3 que sirven para hacer el ensamblaje con Bs2_SJ1_TOP



Figura 6.8 Pieza impresa ESB2_SJ1_PUL21

Se puede ver la polea de 21 dientes que se encuentra en la articulación 2 y se encarga de recibir la tensión de motor 2. Se aprecia que en su eje central cuenta con un agujero que puede ser usado para incluir una barra de 8 mm que le aporte una mayor rigidez.



Figura 6.9 Piezas impresa ESB2_SJ1_BOT y ESB2_SJ3

Se pueden ver las que construyen el sistema de desplazamiento del eslabón 3. Por ello, en la imagen superior, se encuentra el LS de ESB3 y los ejes de entrada para las barras lineales del ESB3. En la imagen inferior se ve con mayor claridad la tuerca para el tornillo de avance y, además, el encaje de la polea de 21 dientes que provoca el movimiento de este módulo ESB2 y su LS.

6.1.4. MÓDULO ESB3

Según el desarrollo del diseño del módulo ESB3 (4.1.4.4) para poder hacer su construcción hace falta:

- 4 tornillos M3 de 10 mm de longitud con sus tuercas para hacer la sujeción de las barras huecas de aluminio que guían el movimiento de este módulo
- 8 tornillos M2.5 para poder sujetar los motores NEMA 11 en ambos extremos del módulo
- Tornillo de avance de 300 mm de largo, de 4 hilos y con un avance de 8 mm
- Acoplador elástico de aluminio para unir el tornillo de avance con el motor de desplazamiento lineal
- Dos barras huecas de aluminio de 8 mm y 350 mm de largo



Figura 6.10 Módulo montado ESB3

Se puede ver en la imagen izquierda la parte frontal del módulo ESB3. En esta imagen se aprecian las sujeciones superiores e inferiores de las barras de 8 mm. En la imagen izquierda se puede ver la entrada del motor para el gripper y el agujero pasante para que el tornillo de avance pueda girar libremente, pero de forma controlada. En ambas imágenes se pueden ver el tornillo pasante de 300 mm, el acoplador elástico y el motor NEMA 11 superior que mueve dicho tornillo.



Figura 6.11 Pieza impresa ESB3_MGRP

En esta imagen se aprecia de forma más detallada la entrada del motor NEMA 11 no cautivo que se encargará de hacer el movimiento del gripper. También se ven las conexiones para cada uno de los FSR del SG.

6.1.5. MÓDULO PCBOX

Este módulo (Figura 6.12) es el que cuenta con el montaje más sencillo del robot ya que solo se compone de 2 piezas (4.1.4.5) que requieren únicamente de:

- 4 tornillos M3 de al menos 10 mm de largo y la PCB
- 4 tornillos M2 para sujeción de pantalla OLED



Figura 6.12 Pieza impresa PCBOX_TOP

Se puede ver la sujeción con la pantalla OLED con tornillos M2. También se ven las entradas para tornillos M3 y el puerto para USB Micro para alimentación, programación y comunicación serial con Arduino MKR 1010.



Figura 6.13 Pieza impresa PCBOX_BOT

En esta imagen se muestra la pieza base del módulo PCBOX según sus requerimientos estructurales. Se ven las entradas para los tornillos M3.



Figura 6.14 Montaje módulo PCBOX

En esta imagen se muestra el módulo montado junto con la PCB. Se verificó que el encaje fuera el correcto y los tornillos estuvieran alineados.

6.1.6. MÓDULO DE SG

Este módulo se ha creado para poder hacer tareas de sujeción en base al funcionamiento de los SG basado en el efecto FinRay. Sin embargo, hay que recordar que la pieza ESB3_MGRP cuenta con un sistema sencillo de ensamblaje de grippers con el objetivo de que se puedan diseñar otros sujetadores que cuenten con este encaje y se puedan instalar fácilmente para tener otras funcionalidades con el SCARA.

Una vez aclarado este punto se puede, a partir del desarrollo del SG (4.1.2), estudiar los requerimientos que tiene este módulo para su montaje. Se encuentran:

- 3 tornillos M4 con sus tuercas para sujeción con ESB3
- 3 sensores FSR para incluir en SG_FLX modificados con cables hembra que faciliten su instalación
- Tornillo de avance con 4 hilos específico del NEMA 11 no cautivo [57]
- 3 tornillos M3 para ensamblar SG_BS_ACT con el tornillo de avance

Figura 6.15 Pieza impresa SG_FLX



Se ve la pieza flexible que ha demostrado contar con los mejores resultados de sujeción. En ella se muestra el sensor instalado en el nivel bajo de la zona de contacto. Dicho sensor FSR fue modificado para contar con cables que facilitaran la posterior instalación con el cuerpo

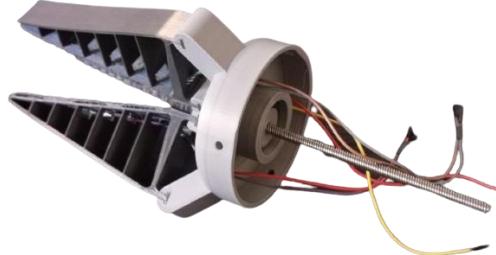


Figura 6.16 Módulo montado SG

En estas imágenes se muestran ensambladas todas las piezas con las que cuenta el módulo. En la imagen izquierda se puede ver la salida del cableado de todos los FSR (uno por SG_FLX) y también el tornillo de avance ensamblado con SG_BS_ACT y que será sujetado por el motor del gripper. A la derecha se tiene una imagen inferior del módulo donde se pueden ver que en total son 3 SG_FLX que tiene el SG para poder hacer la sujeción.

6.1.7. MONTAJE COMPLETO DE SCARA

Para poder hacer el montaje completo es necesario tener una base que pueda soportar el peso del robot. Por ello se construyó una base formada por perfiles de aluminio que requiere:

- 3 perfiles de aluminio de 1 m con referencia XCBM 1X44
- 4 escuadras de sujeción para perfiles
- 10 tuercas para sujeción de escuadras y base de SCARA a perfil

Además, se diseñó una pieza de escuadra que fue impresa en PLA para dar resistencia a el perfil principal de sujeción de la base (Figura 6.17). Y, también, unos extremos de perfiles (Figura 6.19) para asegurar un correcto contacto con el suelo.

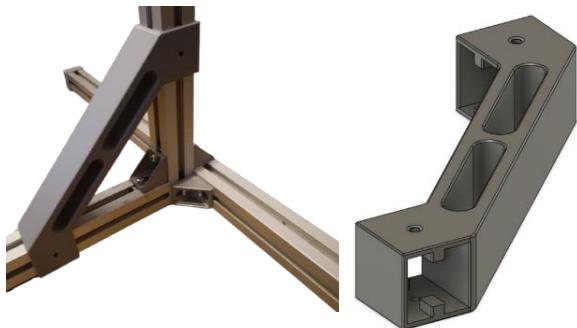


Figura 6.17 Pieza de soporte de perfiles ESC_SUP

En las imágenes se ve la pieza que hace de refuerzo para el perfil principal que soporta el robot. Cuenta con 2 entradas para perfiles en sus extremos y entradas para tornillos M6 para sujetar con tuercas de perfil. A la izquierda se muestra el resultado del montaje de esta pieza para reforzar la estructura.



Figura 6.19 Pieza de cobertura de extremos de perfil COVER_PR

En la imagen derecha se muestra la pieza que cuenta con las dimensiones justas para cubrir el extremo de los perfiles que soportan la base del SCARA. Se requieren de 3 de estas piezas en el montaje total. A la izquierda se muestra el resultado de colocar estas piezas

Con estos detalles aclarados se puede hacer el montaje completo de la base (Figura 6.18) en la que se instalará el cuerpo del robot.



Figura 6.18 Base se soporte de SCARA

En la imagen se puede ver la base montaje con los:

- 3 perfiles de 1 metro
- 4 escuadras de unión de perfiles unidas con sus correspondientes tuercas
- 3 piezas COVER_PR en los extremos de apoyo de la estructura
- Una pieza de refuerzo ESC_SUP.



Figura 6.20 Montaje de los módulos del robot SCARA

Se muestran los módulos ESB1, ESB2, ESB3 y BS de forma en conjunta



Figura 6.21 Estructura completa de robot SCARA

Se muestra el resultado del ensamblaje (imagen izquierda) que se hace para unir la base de perfiles con el cuerpo del SCARA

Posteriormente, se puede hacer el ensamblaje del cuerpo del SCARA con la base para que se tenga una estructura completa (Figura 6.21) con la que, posteriormente, sea posible hacer las pruebas de movilidad y, por otro lado, ensamblar el SG (Figura 6.22) para realizar pruebas de sujeción.



Figura 6.22 Resultado de ensamblaje de SG

En la imagen izquierda se muestra el resultado de ensamblaje de SG con ESB3_MGRP con la ayuda de 3 tornillos M4. Luego, en la imagen derecha, se puede observar la conexión de los FSR con el cableado general del robot

Finalmente, se puede hacer las conexiones de los periféricos a la PCB (4.3.3) y contar con la instalación completa del robot SCARA.

6.2. FUNCIONAMIENTO DE ELECTRÓNICA Y PERIFÉRICOS

Antes de hacer pruebas con la interfaz gráfica se verificó que todos los sensores y periféricos funcionaban correctamente gracias al comando de estado (5.3.4.3) que se puede ver en el terminal de Arduino IDE si es configurado el código con un debug de nivel 2 (5.3.2). Se verificó:

- Salida de información de pantalla OLED (Figura 6.23)
- Pruebas de lecturas de presión con los FSR (Tabla 6.2) y SW implementado (5.3.3.4)
- Prueba de funcionamiento de LS



Figura 6.23 Salida OLED

Se puede ver la pantalla oled indicando la información de inicio de la aplicación de control. Hubo error en la comunicación serial ya que no se abrió el puerto serial del Arduino. Luego se muestra la información para establecer comunicación red e información de interés general que va alternando.

Tabla 6.2 Resultado de medición de fuerza con FSR

Objeto	Peso con balanza (g)	Peso con FSR (g)	Error (%)
Fresa	22	25	13,64
Nuez	11	-	-
Mandarina	85	80	-5,88
Plátano	159	155	-2,52
Cubo	186	189	1,61
Naranja	169	162	-4,14
Motor NEMA 17	373	371	-0,54

En los resultados de medición con el FSR se notó una reducción del error a medida que aumenta el peso. Esto es debido a que la fuerza aplicada es más estable y, además, el sensor entra en un área de funcionamiento más lineal. Por otro lado, se ve que no logra registrar el peso de la nuez ya que no supera el umbral mínimo de fuerza aplicada para este modelo de sensor (4.3.2.2). Después de estas pruebas se comprobó que los periféricos básicos funcionaban correctamente y podían ser integrados en el robot cumpliendo sus funciones correspondientes.

6.3. FUNCIONAMIENTO DE HMI

En este apartado se explican los resultados obtenidos de la GUI conectada con todos los periféricos existentes en el robot y con la simulación de CoppeliaSim (Figura 5.1). Por ello se dividirá esta evaluación en dos puntos que corresponden a estos tipos de funcionamiento.

6.3.1. PRECONDICIONES

Antes de cualquier prueba se comprobó el funcionamiento de:

- Establecimiento de la conexión en función de la simulación establecida (Figura 6.27)
- Configuración de parámetros de conexión (Figura 6.26)
- Configuración del tipo de simulación que se está realizando (Figura 6.24)
- Configuración de los parámetros de control del robot y configuración (Figura 6.25)
- Comprobación de estado de conexión (Figura 6.27)
- Estados de aplicación en GUI (Figura 5.11)
- Comprobación de guardado de información de atributos al cerrar aplicación



Figura 6.24 Configuración de conexión

Desde la interfaz se accede a Edit -> Connection

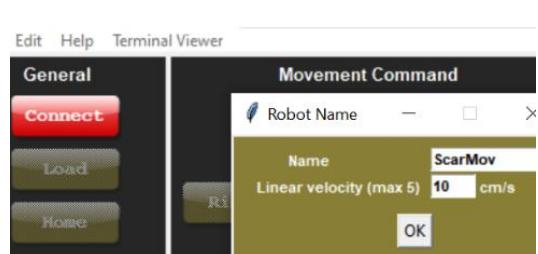


Figura 6.25 Parámetros de configuración de robot

Desde la interfaz se accede a Edit -> Robot

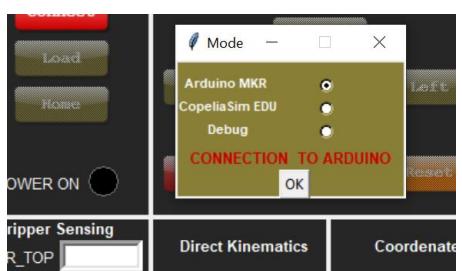


Figura 6.26 Comprobación de configuración de tipo de aplicación de control

Desde la interfaz Edit->Simulation.

Se muestran las 3 opciones disponibles de control y la que actualmente está escogida.

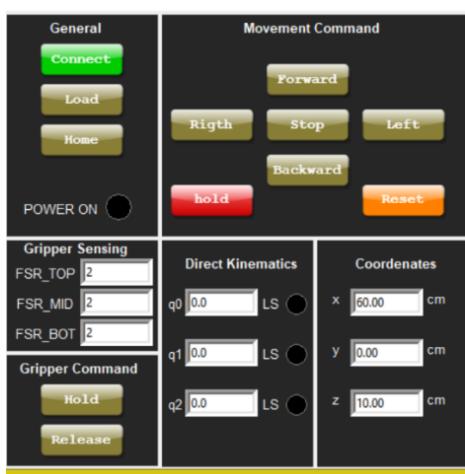


Figura 6.27 Prueba de botón de conexión de GUI

Se muestra una conexión establecida con la placa Arduino y que ha empezado a recibir información de estado del robot. Se puede ver que los botones son activados una vez se ha establecido la conexión para su uso. Los estados de todos los sensores son cero (LS negro) y el robot no se encuentra alimentado (POWER ON en negro).

6.3.2. RESULTADO DE GUI CON ROBOT

Para validar este modo de funcionamiento se comprobaron los siguientes puntos:

- Estado de los LS (Figura 6.28)
- Lectura de cada FSR (Figura 6.28)
- Posición de los motores al moverse (Figura 6.28)
- Estado de alimentación de motores (Figura 6.28)
- Construcción de plano con rango de control (Figura 6.28)
- Envío de comandos de posicionamiento y de sujeción de SG

El objetivo de esta prueba es verificar que el protocolo de comunicación por red (5.5) esté funcionando y todos los paquetes son codificados y codificados correctamente por ambos módulos del código. Por ello solo se evaluaron las señales que aparecían en la interfaz gráfica (Figura 6.28) y los comandos que recibía en módulo de control con la ayuda del terminal serial y el modo debug configurado para imprimir la recepción de comandos y estado de la aplicación.

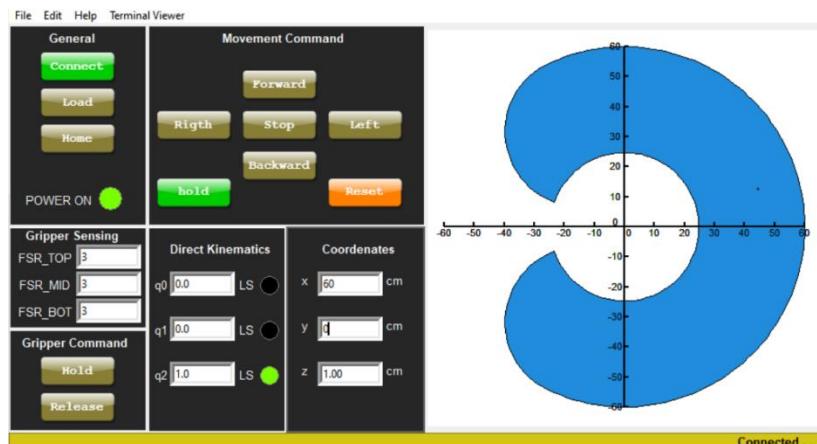


Figura 6.28 Monitorización de señales con GUI

Se puede ver que los motores al alimentarse se ven reflejado POWER ON en verde. Además, se aprecia que el eslabón 3 se ha desplazado 1 cm y tiene LS activo. Por otro lado, se incluye la representación gráfica que corresponde al robot que ha sido conectado con la GUI y en la cual es posible indicar la posición XY para desplazar el robot.

6.3.3. RESULTADOS CON SG

Además de hacer control de posicionamiento se debe evaluar que es posible enviar los comandos correspondientes al SG para hacer pruebas de sujeción de forma controlada (6.5). Por ello debe existir:

- Posibilidad de configuración de radio de sujeción
- Lectura de los FSR constantemente durante la sujeción y mostrados en pantalla
- Posibilidad de configuración de corriente usada para el motor del SG



Figura 6.29 Resultados de control de SG

Una vez conectado al Arduino se puede verificar que se activan los botones correspondientes a el control de sujeción del gripper. Esta acción depende de la configuración que tenga el gripper (Edit -> Gripper). Finalmente, se puede ver que el FSR superior detecta la presión en Newtons.

6.4. FUNCIONAMIENTO DEL CUERPO DEL ROBOT

En este apartado evaluaremos el funcionamiento del robot con base en el control de cada motor por separado y, posteriormente, con el envío del comando de movimiento para que realice la cinemática inversa correspondiente al desplazamiento. Por esta razón se dividirá el apartado en 2 puntos:

- Evaluación unitaria de comandos hacia el robot para pruebas de funcionamiento individual y control de periféricos
- Evaluación de funcionamiento en conjunto con la GUI

En ambos casos se monitoriza el estado de conexión del módulo de control con la ayuda del terminal serial que incluye Arduino IDE (5.1.1) y, a su vez, lo que se envía desde la GUI o el script de pruebas con ayuda del terminal de VSC (5.1.2).

6.4.1. RESULTADOS PRUEBAS UNITARIAS

Para evaluar el funcionamiento del robot se realizaron pruebas unitarias de posicionamiento. De esta forma se puede evaluar el correcto movimiento angular o lineal de cada eslabón para luego hacer pruebas de la cinemática completa que deben realizar los eslabones en conjunto. Por esta razón se creó un script de pruebas en Python que contara con la capacidad de:

- Conexión IP/UDP configurable de forma manual
- Envío de comandos de movimiento angular o lineal en función del motor usado
- Envío de comando home para prueba de funcionamiento de algoritmo y LS
- Entrada por la línea de comandos para indicar qué motor se quiere probar e ingresar la posición de destino
- Recepción de estado para comprobación de sensores LS y FSR

Por esta razón cuenta con el protocolo desarrollado en JSON (5.5) para que el robot pueda entender los comandos enviados y poder realizar las pruebas que se verán en esta parte.

6.4.1.1. PRUEBA DE HOME

Para realizar las pruebas se utilizaron los siguientes comandos:

- Para motores 0 y 1 se les envía el comando home con una velocidad de 45°/s y un retorno de 10°
- Para el motor 2 (ESB3) se le envía el comando home con una velocidad de 5 cm/s y un retorno de 1 cm
- Comando de home para realizar la posición de inicio de los 3 motores a la vez

Se comprueba la funcionalidad de el comando de inicio de posición de cada motor y también el funcionamiento en paralelo. Además, se verifica que al cambiar cualquier tipo de parámetro de inicio el motor responde correctamente.

Una observación que es importante resaltar en esta prueba es el hecho de que la velocidad de home no debe exceder de las utilizadas ya que pueden provocar problemas al chocar con los LS y hacer que haya rebotes en la estructura. Principalmente en el ESB1 que tiene la mayor inercia durante el movimiento. Además, se hizo evidente que sin el control activado para el arranque era complicado moverse correctamente al ESB1 ya que buscaba conseguir esta velocidad con la aceleración máxima configurada y por ende con una perdida notable de torque necesaria para el arranque de este eslabón.

6.4.1.2. PRUEBAS DE DESPLAZAMIENTO

En estas pruebas se utilizaron los siguientes comandos:

- Para motores 0 y 1 se envían comandos de posicionamiento angular con velocidades de V1 = 15°/s, V2 = 30°/s y V3 = 45°/s
- Para motor 2 se envían posiciones a alcanzar con velocidades de V1= 5 cm/s , V2 = 7cm/s y V3 = 10 cm/s

Estas pruebas fueron de gran ayuda para detectar fallos de funcionamiento. Entre los más recurrentes se encontraron:

- Error en conexión de los motores
- Problemas de aislamientos en la sujeción de las piezas BS_PULL27 que se conectaban a motores
- Corriente insuficiente en los motores para funcionamiento correcto

Una vez se solventaron las fallas se probó con éxito la movilidad de cada eslabón según su tipo de articulación y se verificó que los desplazamientos angulares y lineales eran correctos según el comando enviado y la velocidad requerida. Para ello se evaluaron los errores de posicionamiento en función de la velocidad aplicada y con el control SW activado (Tabla 6.3).

Tabla 6.3 Resultados de errores medios en desplazamiento de eslabones

Eslabón	Error V1	Error V2	Error V3
ESB1 (°)	0.5	1	2.5

ESB2 (°)	0	0.8	1.5
ESB3(m)	0	0.5	0.9

Es evidente que, a medida que se aumenta la velocidad, existe un mayor error debido a la propia inercia del mecanismo. Esto se debe a que estos comandos no generan ninguna acción de control y en el momento de arranque y fin se provoca un cambio brusco de corriente que puede provocar salto de pasos. Además, se puede observar, como era de esperarse, que el ESB1 cuenta con una mayor inercia y por ende un mayor error a la hora de hacer desplazamientos.

6.4.2. PRUEBAS CON CINEMÁTICA INVERSA

Para estas pruebas se cuenta con la GUI ya verificada y que puede enviar los comandos correspondientes a la posición a la que se desea desplazar el robot. En este sentido se pueden hacer pruebas de funcionamiento según los algoritmos de trayectorias existentes (5.6.1.1) y así comprobar los errores de movimiento (Tabla 6.4). Para ello se realizaron las pruebas en los tramos:

- Tramo en línea recta: desde $p_1 = [40,40]cm$ hacia $p_2 = [40, -40]cm$
- Tramo con intersección con circunferencia central límite: para ello se envía desde $p_1 = [20,40]cm$ hacia $p_2 = [20, -40]cm$
- Tramo que provoque cambio de lógica de posicionamiento en los eslabones ESB1 y ESB2: se comienza con posicionamiento de inicio en posición $p_1 = [40,40]cm$ y se solicita desplazamiento hacia $p_2 = [10,35]cm$

Para cada prueba se realizaron 3 versiones para las cuales se utilizaron las velocidades:
 $V1 = 5 \text{ cm/s}$, $V2 = 7\text{cm/s}$, $V3 = 10 \text{ cm/s}$.

Tabla 6.4 Errores medios de movimientos XY

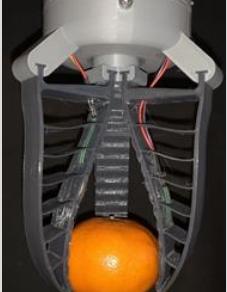
Tramo	Error V1(mm)		Error V2(mm)		Error V3(mm)	
	x(mm)	y(mm)	x(mm)	y(mm)	x(mm)	y(mm)
1	2	3	4	4	4	5
2	1	3	3	3	4	2
3	2	2	4	4	5	6

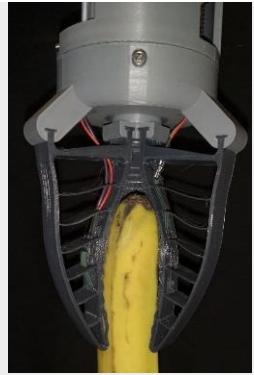
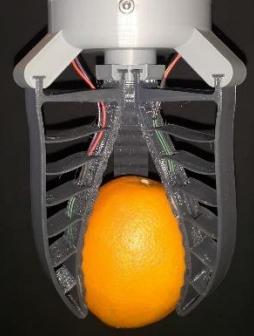
Las pruebas correspondientes al ESB3 ya se efectuaron en las pruebas unitarias (Tabla 6.3) ya que, haciendo el envío del comando de movimiento para Z desde la GUI, se obtendrían los mismos resultados.

6.5. RESULTADOS DE SUJECIÓN CON SG

Una vez se cuenta con una interfaz gráfica que permite la entrada de comandos de control del SG de forma configurable (6.3.2) se pueden realizar distintas pruebas enfocadas en evaluar los resultados de sujeción que van en función del radio del objeto a sujetar (Tabla 6.5) y, en caso de ser necesario, cambiar la corriente de sujeción del motor del SG. En este caso se han hecho pruebas con una corriente al 50 % de funcionamiento máximo del motor NEMA 11, es decir, a unos 0.3 A de corriente durante su desplazamiento.

Tabla 6.5 Objetos de prueba de sujeción

Objeto	Diámetro		Imagen
	Peso(g)	característico (cm)	
Fresa	22	3	
Nuez	11	3.5	
Mandarina	85	4	

Plátano	159	3.5	
Cubo	186	5.3	
Naranja	169	6.7	
Motor NEMA 17	373	4.3	

Los mejores resultados, como era de esperar, se han conseguido con los objetos que cuentan con forma elipsoidal. Sin embargo, es posible hacer sujeción con objetos con formas distintas. Tal es el caso de la sujeción del motor, el cubo y el plátano.

6.6. RESUMEN DE CARACTERÍSTICAS DE ROBOT SCARA

Luego de las pruebas realizadas se llegó a un resumen de caracterización del funcionamiento del robot desarrollado (Tabla 6.6).

Tabla 6.6 Características de SCARA

Parámetros	Valor
Voltaje de alimentación	12 V
Corriente Nominal	2.5 A
Comunicación	WiFi
Número de sensores LS	3
Carga Máxima (Kg)	0.4
Carga Nominal	0.3kg
Alcance (entre eje 1 y 3)	600mm
Grados de libertad	3
Peso(Kg)	6
Rango Eje 1	230°
Rango Eje 2	270°
Rango Actuador 3	280 mm

6.7. RESULTADOS DE SIMULACIÓN CON BASE MÓVIL

Además del control del SCARA que se ha desarrollado físicamente, es posible comprobar el funcionamiento de este robot sobre una base móvil de forma simulada. Para ello hay que elegir la opción correspondiente a la configuración de simulación con CoppeliaSim desde la interfaz gráfica desarrollada (Figura 5.23). Una vez hecho esto se comprobó:

- Conexión con CoppeliaSim y mantenimiento de estado (Figura 6.30). En este caso no se manejan sensores ni señales de alimentación
- Envío de comandos de cinemática inversa para el SCARA (Figura 6.31)
- Envíos de comandos de desplazamiento para base móvil (Figura 6.32)

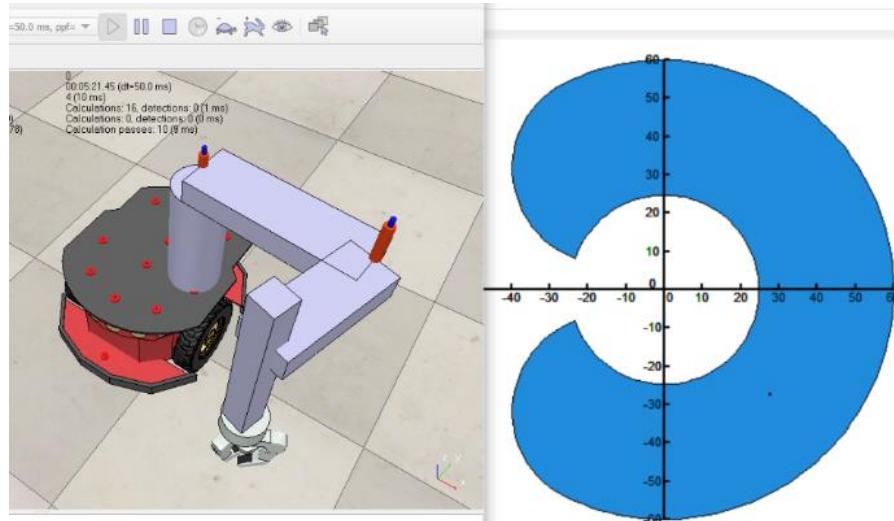


Figura 6.30 Conexión a simulador CoppeliaSim desde GUI

Se ve a la izquierda la conexión al IP local y el puerto establecido desde el simulador. A la derecha se ve la activación de los botones de control de la base móvil con comandos básicos de funcionamiento.

Figura 6.31 Cinemática inversa en CoppeliaSim

Se puede ver a la izquierda el posicionamiento de los eslabones 1 y 2 en la posición que ha sido enviada desde las coordenadas de la interfaz gráfica



En la simulación se incluye una esfera de 7 cm de diámetro que estará colocada a una distancia mayor del rango básico del SCARA (Figura 6.32). Por ende, es necesario desplazar la base móvil hasta una posición en la que sea posible la sujeción de la esfera.

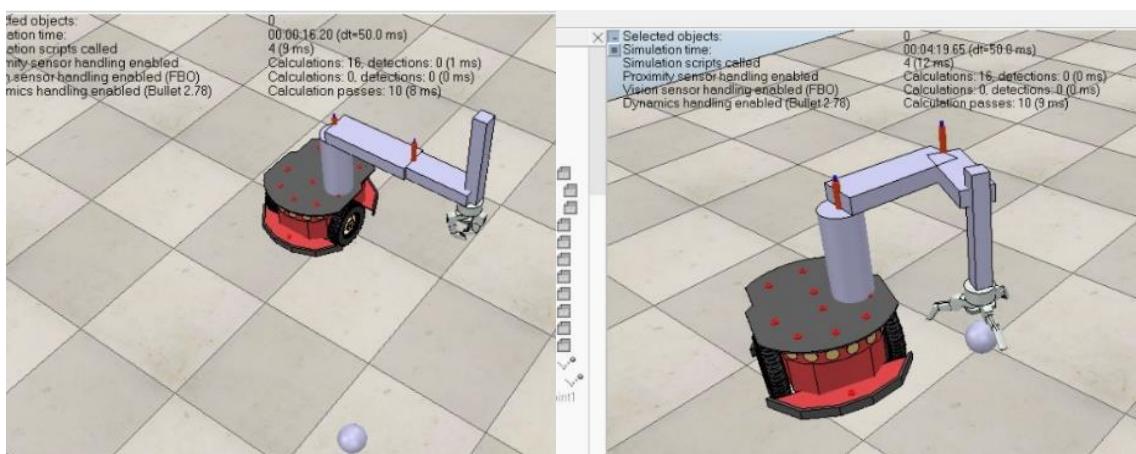


Figura 6.32 Simulación con base móvil en CoppeliaSim

En las imágenes se muestra una simulación iniciada para recibir comandos desde la GUI a través de un socket local del ordenador. A la izquierda se puede ver la esfera que se quiere sujetar y que se encuentra a una distancia mayor que el rango de alcance del SCARA. Posteriormente se envían los comandos de movimiento de la base móvil y la cinemática inversa que lograron, tal como se ve en la imagen derecha, que el robot se posicionara sobre la esfera y pudiera sujetarla.

6.8. DISCUSIÓN DE RESULTADOS

A partir de las pruebas realizadas se puede decir que, en general, el funcionamiento del robot cumple con los requerimientos y especificaciones indicadas como objetivos del proyecto.

En primer lugar, se comprobó que las piezas fabricadas con tecnología FDM y corte láser cumplieron sus funciones estructurales y de resistencia. Además, se evidenció que los diseños cuentan con una geometría optimizada y con ajustes que permiten un montaje relativamente sencillo. Esto facilita el trabajo de ensamblaje y permite que cada módulo del cuerpo del SCARA pueda ser instalado y desinstalado rápidamente en caso de que surja cualquier problema o se quieran realizar pruebas con un eslabón en específico. También, se comprobó que los eslabones cuentan con mecanismos de desplazamiento angular y lineal que evitaban los rozamientos. Esto fue de gran importancia a la hora de hacer la comprobación de la cinemática inversa sobre el robot.

En segundo lugar, se pudo hacer la integración de los dos módulos SW desarrollados con un protocolo de comunicación personalizado para esta aplicación. En este sentido se utilizó la interfaz gráfica como principal HMI que sirviera de conexión con el robot para comprobar que todo el HW funcionaba correctamente. Así se verificó a la vez el correcto funcionamiento de la misma interfaz gráfica y la aplicación de control con base en los requerimientos especificados de monitorización y manejo de periféricos. Para ello se realizaron pruebas unitarias que arrojaron resultados bastante buenos de posicionamiento ya que los errores se encuentran en el orden de 3 mm en cada coordenada y dependen, principalmente, de la velocidad que se le haya solicitado al motor. Esto también hay que unirlo con el hecho de que no se hace uso de algún sensor de posición angular que cierre el bucle de forma real, por ende, estos errores pueden deberse a la pérdida de pasos en los motores que no es comprobada desde los registros de posicionamiento de los controladores de cada motor. Sin embargo, se encuentra en el rango que se esperaba de funcionamiento contando con una velocidad máxima de desplazamiento de 10 cm/s.

También se verificó, con ayuda de la interfaz gráfica, el uso del SG para distintos tipos de objetos. En este caso se pudo comprobar que este mecanismo de sujeción permite aprovechar en gran medida la potencia aportada por motor ya que se pudo incluso superar las expectativas de 300 g como máxima carga que el robot podía sujetar.

Finalmente, se comprobó que los comandos que se enviaban a la aplicación de simulación CoppeliaSim funcionaran correctamente. Esto fue de gran ayuda durante el desarrollo de todo el algoritmo de control y, además, sirve para comprobar que la aplicación puede ser integrada en un futuro junto con una base móvil que permita que el robot cuente con un mayor rango de desplazamiento.

Capítulo 7. CONCLUSIONES Y FUTUROS TRABAJOS

En este último capítulo se presentan las conclusiones obtenidas después realizar las pruebas en el robot SACARA en cuanto a su desplazamiento, el funcionamiento de SG y la integración de los módulos SW. También se hace un resumen del resultado de la fabricación y el montaje de todos los componentes, y se describen ideas de posibles mejoras que han ido surgiendo durante el desarrollo del robot que se pueden implementar en futuros proyectos.

7.1. CONCLUSIONES

Después de hacer la fabricación y ensamblaje de todas las piezas del SCARA se ha demostrado que el cuerpo presenta una estructura compacta y sólida. Es evidente que el sistema de tornillos internos (Figura 4.15) fue de gran ayuda a la hora de aportar solidez en los ensamblajes sin dañar la estética del robot. Por esta razón y por la facilidad de montaje conseguida se ha comprobado que la tecnología de impresión 3D es bastante adecuada para este tipo de desarrollos ya que el resultado muestra unas uniones firmes y no se aprecian deformaciones a pesar del peso del robot y las grandes tensiones cortantes que pueden llegar a sufrir sus eslabones.

Otro punto para destacar de la impresión 3D, es la gran versatilidad que ofrece durante el proceso de fabricación de piezas. Es evidente que, en comparación con otras tecnologías usadas como corte láser, ofrece innumerables ventajas de diseño que se traducen en un mayor aporte de funcionalidades a cualquier modelo que se quiera crear y que requiera de geometrías complicadas. Esto fue de gran ayuda a la hora de diseñar, desde cero, piezas personalizadas y que requieren de distintas propiedades físicas y mecánicas, como fueron:

- Sistema de poleas T5 que no se encuentran en el mercado
- Caja de cobertura de la PCB que cumple con múltiples finalidades estéticas
- Diseño de SG de fácil instalación y cambio
- Estructuras internas complejas
- Posibilidad de optimización de material de fabricación
- Impresión de material flexibles con prestaciones técnicas como es el TPU que demostró ser el más adecuado para la fabricación de SG_FLX

A pesar de las numerosas ventajas que se hicieron evidentes a la hora de utilizar tecnología 3D, hay que mencionar que para poder alcanzar el objetivo de este proyecto se ha dedicado una gran cantidad de tiempo en perfeccionar el diseño. Esto se debe a que era de vital importancia el hecho de contar con un sistema de poleas de transmisión y un ajuste para rodamientos que aseguraran una correcta movilización de cada eslabón y que evitaran los rozamientos y dobleces. Esto llevó a la ejecución varias interacciones de diseño y prototipado con impresión 3D que fue de gran ayuda a la hora de descartar modelos que no cumplían con los requerimientos que se habían especificado en un principio.

Por otro lado, el HW electrónico y eléctrico que se ha integrado es el necesario para el funcionamiento básico del robot y cuenta con múltiples funcionalidades para percepción de información del ambiente de funcionamiento que pueden llegar a ser potenciadas en proyectos

futuros con inteligencia artificial o algoritmos avanzados de control. Tal es el caso del uso de los sensores de presión FSR que, por su posicionamiento, pueden ser integrados en algoritmos especializados en detección de formas o para regulación automática de la fuerza de sujeción del SG. Además, los motores utilizados cuentan con una potencia aceptable para la aplicación que se busca con el SCARA y, por sus características, su control fue bastante simplificado con el uso de los controladores TMC5160 ya que estos módulos aportaron múltiples funciones y posibilidades de manejo que fueron utilizadas a nivel de SW para desarrollar el algoritmo de control de posición.

También, cabe destacar que el desarrollo SW fue un pilar fundamental a la hora de establecer la arquitectura de aplicación que se quería alcanzar. Por ello se crearon 2 módulos de aplicación SW que cuentan con funcionalidades específicas e importantes para controlar el robot. Por un lado, se incluyó una interfaz de usuario que aporta un gran valor agregado al sistema debido a que ofrece una herramienta HMI desde la que se pueden enviar comandos de forma sencilla para controlar el robot. Es decir, que cuenta con el objetivo de ser intuitiva y, de esta forma, de poder ser utilizada por cualquier persona que no sea especialista en el funcionamiento interno de toda la aplicación. Por otro lado, se desarrolló una aplicación enfocada en el control directo de los motores y que podría ser ejecutada desde el propio CU del robot físico o desde una interfaz de simulación que sirviera de banco de pruebas para la verificación de los comandos enviados al SCARA.

Además, tanto el módulo GUI como el módulo de control fueron diseñados con el objetivo de ser fácilmente integrados y de contar con una alta versatilidad. Solamente faltaba establecer el tipo de comunicación más adecuado para su conexión. Para ello se demostró que la mejor opción se conseguía con el uso del protocolo IP/UDP ya que este cuenta con ciertas propiedades que lo hacían el más adecuado para envío de paquetes con baja latencia vía WiFi. Además, se creó un protocolo de empaquetamiento de datos en formato JSON que asegurara la comunicación por red entre ambos módulos y que admitiera futuras ampliaciones en sus comandos. Así, se ha logrado desacoplar ambos niveles SW y, de esta manera, se asegura que estos sean escalables y puedan ser integrados con otras aplicaciones, ya que utilizan un protocolo estándar y altamente empleado en los sistemas de comunicación.

Luego de la finalización del proyecto se realizaron las pruebas de mayor interés sobre cada parte HW y SW. De esta forma se hizo la comparación entre los objetivos planteados y los que se han conseguido. A partir de estos resultados se ha evidenciado que:

- El procedimiento de producción HW y SW es el adecuado para llegar a los objetivos planteados
- El desplazamiento del SCARA cumple con los objetivos de velocidad y repetibilidad de posicionamiento contando con un pequeño error
- La sujeción y desempeño del SG con el uso del efecto FinRay cumplió con los objetivos a la perfección
- La GUI aporta todas las funcionalidades necesarias para el control de robot, simulación con base móvil y monitorización de señales
- La aplicación de control logra cubrir todas las funcionalidades en cuanto al control de periféricos instalados en el robot

- La simulación en CoppeliaSim responde correctamente ante los comandos enviados desde la GUI

Por último, cabe destacar que en este proyecto ha sido posible utilizar numerosos campos de la ingeniería como pueden ser:

- Robótica
- Electrónica
- Aplicación de programación orientada a objetos
- Implementación de lenguaje de bajo nivel para control de MCU
- Montaje de sistema de comunicación red con protocolo personalizado
- Teoría de control
- Diseño 3D
- Métodos de fabricación con tecnologías avanzadas como impresión 3D y corte láser
- Teoría de resistencia de materiales

Gracias a esta integración se ha conseguido desarrollar un prototipo de robot SCARA de tres grados de libertad con capacidad de soporte para distintos mecanismos de sujeción. Para las pruebas se incluyó un SG que permite sujetar objetos elipsoidales (y, en ciertos casos, no elipsoidales) a medida que registra la presión ejercida sobre estos. Además, toda la información y comandos de control están integrados en una GUI de fácil uso que puede conectarse vía WiFi con el robot.

7.2. FUTUROS TRABAJOS Y MEJORAS

Lo más importante de este trabajo es que se establece un FWK capaz de controlar de manera sencilla el robot SCARA. Por ello es posible integrar esta funcionalidad a trabajos futuros de mayor envergadura que aprovechen esta base de desarrollo:

- **Inclusión de encoders:** en el algoritmo de control implementado (5.6.2) se utiliza como señal de realimentación el registro de posición de los controladores (4.3.3.1) usados para mover los motores. Esto puede acarrear posibles errores en pérdidas de paso que pueden acumularse a medida que la aplicación esté funcionando. Por esta razón lo más recomendable sería incluir un encoder en cada eje de motor que asegure el posicionamiento y exista una comparación entre la posición real y la registrada por el TMC5160. De esta forma podemos realizar controles más avanzados de movimiento.
- **Comunicación por Bluetooth:** se ha visto que el módulo NINA-W102 [50] soporta comunicaciones Bluetooth. Por ello, una posible mejora, que aseguraría la transmisión de información de datos entre la GUI y el robot, sería incluir una capa de código en el gestor de comunicaciones del módulo de control que soporte BLE. Lógicamente deben hacerse los cambios correspondientes en el objeto de conexión del módulo GUI. De esta forma evitamos el uso de una red WiFi con posibles vulnerabilidades a la hora de enviar el paquete de información de control.
- **Sensores FSR:** los modelos FSR07 utilizados funcionaron perfectamente para medir presión, pero cuentan con el inconveniente de que su área de acción es bastante reducida y concentrada (Figura 4.54). Es por ello que se plantea el uso de sensores de presión con un área rectangular que pueda abarcar toda la zona de contacto del SG_FLX.

Por otro lado, con la posibilidad de contar con 3 puntos de presión es posible agregar un algoritmo que interprete la forma del objeto sujetado a partir de cuanta presión sea ejercida en cada punto. Lógicamente, mientras más puntos de lectura existan, mayor va a ser la precisión en la obtención de la forma del objeto.

- **Montaje de SCARA en base móvil:** como ya se cuenta con la interfaz y protocolo de envío de comandos probada para enviar mensajes tanto al robot SCARA con una base móvil, es posible hacer una integración del brazo robótico sobre una base móvil que pueda sostenerlo.

Para este propósito, según las observaciones realizadas del actuador, es probable que sea necesario alguna mejora del cuerpo del robot para que sea más liviano y se eviten problemas de inercia durante el movimiento en conjunto con el robot móvil.

- **Aplicaciones IoT:** gracias a que la comunicación entre el módulo GUI y el módulo de control se realizan con IP/UDP con un formato JSON, es bastante sencillo hacer que la aplicación de control sea integrada para recibir comandos desde un servidor web. En este servidor se alojaría todo el funcionamiento planteado para el módulo GUI e incluso algunas mayores prestaciones y ventajas que ofrece el uso de un navegador:

- Conexión con base de datos
- Herramientas visuales avanzadas
- Control desde redes externas

Todos estos cambios no afectarían al módulo de control siempre que se mantenga el mismo formato y comandos de control con JSON (5.5.1).

- **Aplicaciones de segunda generación de robots:** con la base de funcionamiento del robot se pueden incluir sistemas de visión artificial para reconocimiento de entorno. De esta manera se podrían distinguir objetos que tengan una determinada forma o color para saber cuál es la posición a la cual el robot debe dirigirse para sujetarlo.

Con esta ampliación de las propiedades de percepción se pueden implementar funciones específicas para tareas de pick and place que requieran de la toma de objetos con ciertas características y, además, para aportarle al robot la posibilidad de tomar decisiones a la hora de efectuar su trabajo.

- **Montaje de actuadores con múltiples funciones:** es posible desarrollar un actuador que tenga otras funcionalidades y que pueda ser fácilmente incluido en el SCARA. Además, al contar con un sistema de fácil instalación solamente haría falta cambiar el mecanismo de acción del gripper para que se pueda trabajar con él. Incluso podría desarrollarse un gripper que se base en un extrusor de filamentos y, así, poder desempeñar trabajos de impresión 3D (Existen proyectos parecidos desarrollados para corte y grabado láser que hacen uso del FW de impresión 3D Marling¹).

¹ www.marlinfw.org

BIBLIOGRAFÍA

- [1] Stäubli, "FAST picker TP80 robot", 2018.
- [2] FANUC, "Robot SCARA SR-6iA ", 2019.
- [3] How To Mechatronics, "SCARA ROBOT | HOW TO BUILD YOUR OWN ARDUINO BASED ROBOT", 2019.
- [4] J. Shintake *et al*, "Soft Robotic Grippers", *Adv Mater*, vol. 30, pp. 1707035, 2018. . DOI: 10.1002/adma.201707035.
- [5] Black Ram Industries, "Fin Gripper (Robotic/Prosthetic Hybrid) - Mark VI", Marzo, 2020.
- [6] E. Roels *et al*, "Additive Manufacturing for Self-Healing Soft Robots", *Soft Robotics*, vol. 7, (6), 2020.
- [7] B. Berman, "3-D printing: The new industrial revolution", *Bus. Horiz.*, vol. 55, (2), pp. 155-162, 2012. Available: <https://www.sciencedirect.com/science/article/pii/S0007681311001790>. DOI: <https://doi.org/10.1016/j.bushor.2011.11.003>.
- [8] N. Shahrubudin, T. C. Lee and R. Ramlan, "An Overview on 3D Printing Technology: Technological, Materials, and Applications", *Procedia Manufacturing; the 2nd International Conference on Sustainable Materials Processing and Manufacturing, SMPM 2019, 8-10 March 2019, Sun City, South Africa*, vol. 35, pp. 1286-1296, 2019. . DOI: <https://doi.org/10.1016/j.promfg.2019.06.089>.
- [9] Y. Yang *et al*, "A 3D-Printed Fin Ray Effect Inspired Soft Robotic Gripper with Force Feedback", *Micromachines (Basel)*, vol. 12, (10), pp. 1141, 2021. . DOI: 10.3390/mi12101141.
- [10] Z. Bobovský *et al*, "Structural Optimization Method of a FinRay Finger for the Best Wrapping of Object", *Applied Sciences*, vol. 11, (9), pp. 3858, 2021. . DOI: 10.3390/app11093858.
- [11] J. Pi *et al*, "An Octopus-Inspired Bionic Flexible Gripper for Apple Grasping", *Agriculture*, vol. 11, (10), 2021. . DOI: 10.3390/agriculture11101014.
- [12] J. Suder *et al*, "Structural Optimization Method of a FinRay Finger for the Best Wrapping of Object", *Applied Sciences*, vol. 11, (9), 2021. . DOI: 10.3390/app11093858.
- [13] P. Jha, "A Neural Network Approach for Inverse Kinematic of a SCARA Manipulator", *International Journal of Robotics and Automation*, vol. 3, pp. 31-40, 2014. . DOI: 10.11591/ijra.v3i1.3201.
- [14] A. Barrientos, L. F. Peñín and C. Balaguer, "*Fundamentos De Robótica*" (2a. Ed.). Madrid: McGraw-Hill España, 2007.
- [15] W. Khalil and E. Dombre, "*Modeling, Identification and Control of Robots*", Jordan Hill: Elsevier Science & Technology, 2004.
- [16] H. Liu *et al*, "A single-actuator gripper with a working mode switching mechanism for grasping and rolling manipulation", in Jul 2018, pp. 359-364.
- [17] R. Mutlu *et al*, "A 3D printed monolithic soft gripper with adjustable stiffness", in - *IECON 2017 - 43rd Annual Conference of the IEEE Industrial Electronics Society*, 2017, . DOI: 10.1109/IECON.2017.8217084.

- [18] J. Shintake *et al*, "Soft Robotic Grippers", *Advanced Materials (Weinheim)*, vol. 30, (29), pp. e1707035-n/a, 2018. Available:
<https://onlinelibrary.wiley.com/doi/abs/10.1002/adma.201707035>. DOI: 10.1002/adma.201707035.
- [19] J. M. Gandarias, J. M. Gómez-de-Gabriel and A. J. García-Cerezo, "Enhancing Perception with Tactile Object Recognition in Adaptive Grippers for Human–Robot Interaction", *Sensors*, vol. 18, (3), 2018.
- [20] S. Stassi *et al*, "Flexible tactile sensing based on piezoresistive composites: a review", *Sensors (Basel, Switzerland)*, vol. 14, (3), pp. 5296-5332, 2014. . DOI: 10.3390/s140305296.
- [21] M. Granda Miguel and E. Mediavilla Bolado, "*Instrumentación Electrónica*", Santander: Editorial de la Universidad de Cantabria, 2015.
- [22] L. Ada and D. Nosonowitz, "Force Sensitive Resistor (FSR)", Dic 15, 2021, [en linea]:<https://www.Adafruit.com> (acceso 2022).
- [23] Ohmite, "Force Sening Resistor Integration Guide", pp. 3-8, Enero, 2018.
- [24] M. A. Pérez García *et al*, "*Instrumentación Electrónica*", (1^a). Madrid: THOMSON, 2004.
- [25] L. Paredes-Madrid *et al*, "Underlying Physics of Conductive Polymer Composites and Force Sensing Resistors (FSRs) under Static Loading Conditions", *Sensors (Basel, Switzerland)*, vol. 17, (9), pp. 2108, 2017. . DOI: 10.3390/s17092108.
- [26] W. Yeadon, "*Handbook of Small Electric Motors*", New York [u.a.]: McGraw-Hill, 2001.
- [27] M. Bendjedia *et al*, "Position Control of a Sensorless Stepper Motor", *IEEE Transactions on Power Electronics*, vol. 27, (2), pp. 578-587, 2012. . DOI: 10.1109/TPEL.2011.2161774.
- [28] B. Liptak, "*Process Control: Instrument Engineers' Handbook*", Butterworth-Heinemann, 2013.
- [29] F. J. Meza Weber and P. D. Ramos Morales, "Modelo matemático motor DC conexión independiente", Apr 1, 2015.
- [30] K. Ogata, "*Ingeniería De Control Moderna*", (5a. Ed.). Naucalpan de Juárez: Pearson Educación, 2010.
- [31] F. Monasterio-Huelin and A. Gutiérrez, "Modelado de un motor DC", 29 enero, 2021.
- [32] F. Leens, "An introduction to I²C and SPI protocols", *IEEE Instrumentation & Measurement Magazine*, vol. 12, (1), pp. 8-13, 2009. . DOI: 10.1109/MIM.2009.4762946.
- [33] S. Xie, "Manipulating MCU SPI Interface to Access a Nonstandard SPI ADC", 4 Dic, 2019.
- [34] P. Polisan, V. Priyanka B. and Y. Padma Sai, "Design & Verification of Serial Peripheral Interface (SPI) Protocol", *International Journal of Recent Technology and Engineering*, vol. 8, (6), pp. 793-796, 2020. . DOI: 10.35940/ijrte.F7356.038620.
- [35] Cisco Press, "*Network Basics Companion Guide*", 2014.
- [36] ISO/IEC, "7498-1:1995, ISO/IEC 7498-1:1994: Information technology. Open systems interconnection. Basic reference model. The basic model", 1995.
- [37] A. Elnaggar, "*TCP Vs. UDP*". 2015.

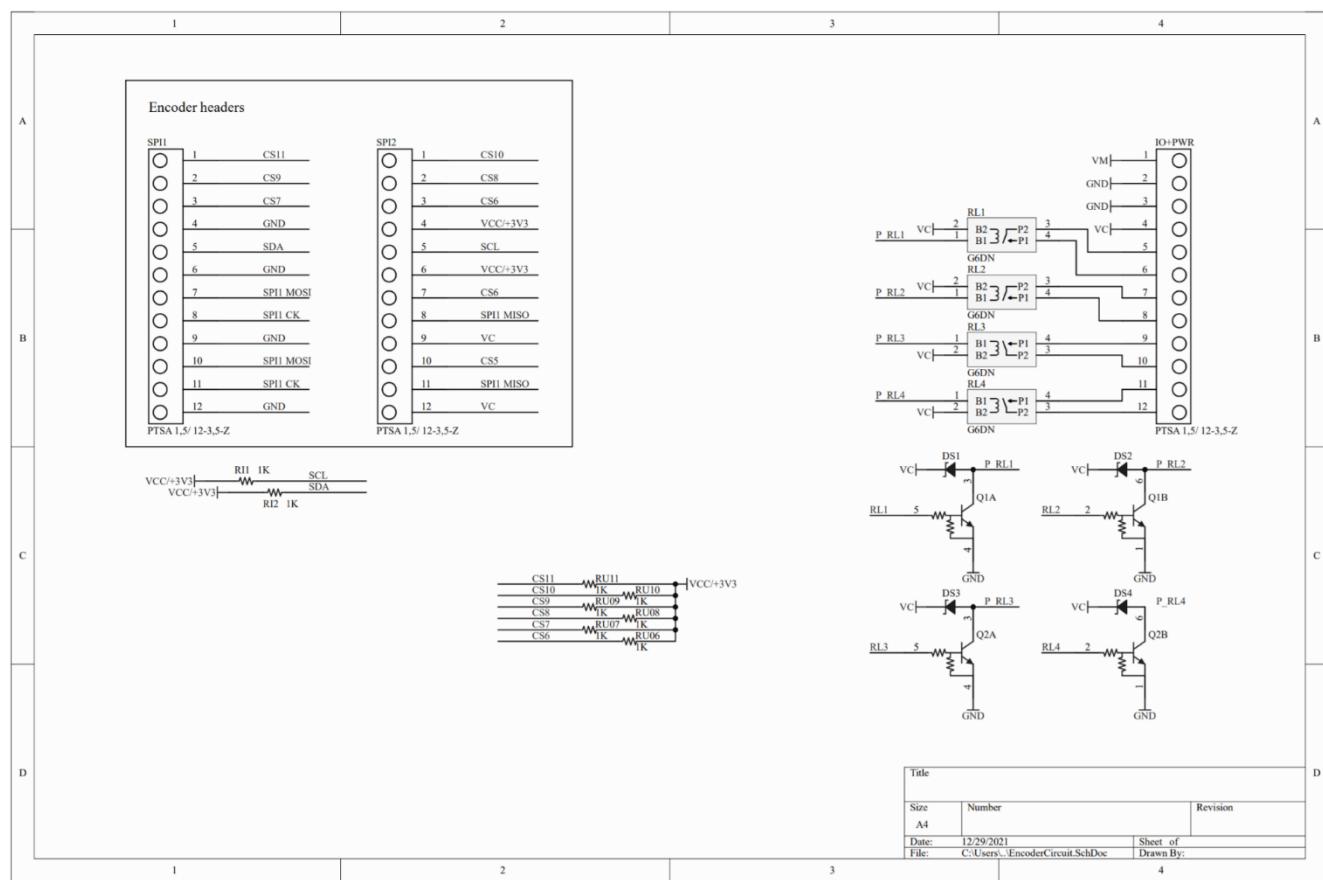
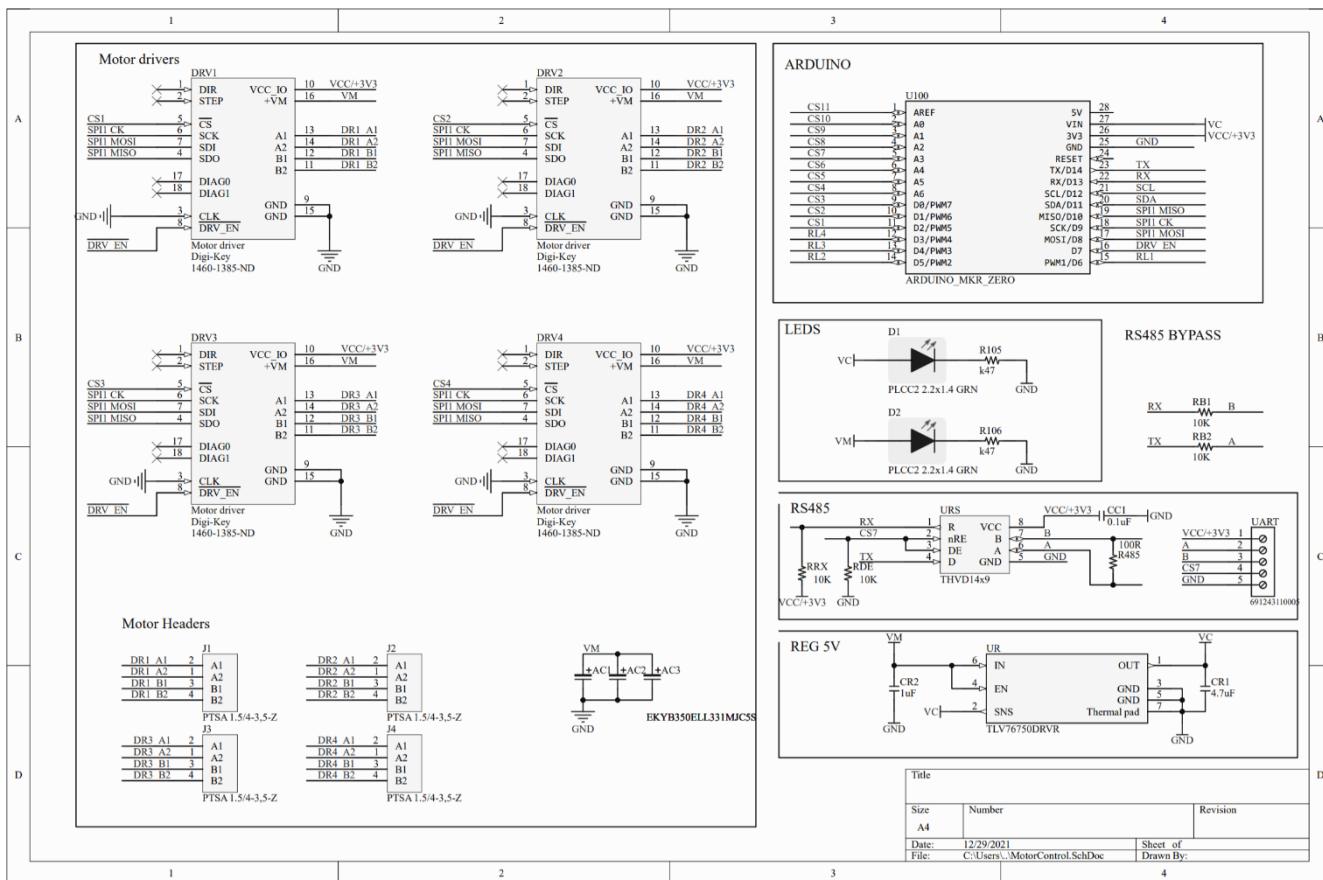
- [38] T. Lv, P. Yan and W. He, "Survey on JSON Data Modelling", *Journal of Physics. Conference Series*, vol. 1069, (1), pp. 12101, 2018. . DOI: 10.1088/1742-6596/1069/1/012101.
- [39] T. Lv *et al*, "On Approximate Querying Large-Scale JSON Data", *Journal of Physics. Conference Series*, vol. 1575, (1), pp. 12066, 2020. . DOI: 10.1088/1742-6596/1575/1/012066.
- [40] M. Munro, "Using JSON", in Anonymous 2021, pp. 319-330.
- [41] S. Terrile, M. Argüelles and A. Barrientos, "Comparison of Different Technologies for Soft Robotics Grippers", *Sensors*, vol. 21, (9), 2021.
- [42] T. A. Harris and M. N. Kotzalas, "*Essential Concepts of Bearing Technology*", Baton Rouge: CRC Press, 2007.
- [43] RS, "Datasheet: Sealed Deep Groove (Popular Metric) Single Row Bearings".
- [44] Optibelt, "Technical manual: Rubber Timing Belt Drives".
- [45] FlexLink, "Structural system XC", 2019.
- [46] M. M. Hanon, R. Marczis and L. Zsidai, "Influence of the 3D Printing Process Settings on Tensile Strength of PLA and HT-PLA", *Period Polytech Mech Eng*, vol. 65, (1), pp. 38-46, 2021.. DOI: 10.3311/PPme.13683.
- [47] Ultimaker, "Important 3D printing software features", 2020.
- [48] 3DFILAMENTS S. L., "Technical Datasheet: eFil TPU 85A", 2021.
- [49] MicroChip, "Data Sheet: SAM D21/DA1 Family", 2020.
- [50] u-box, "Data Sheet: NINA-W10 series", 2021.
- [51] TRINAMIC Motion Control GmbH & Co. KG, "TMC5160 / TMC5160A DATASHEET", 2021.
- [52] TRINAMIC Motion Control GmbH & Co. KG, "TMC5160 SilentStepStick", 2020.
- [53] Ohmite, "FSR Series", 2020.
- [54] Solomon Systech, "SSD1306 data sheet", pp. 18-19, 2008.
- [55] National Electrical Manufacturers Association, "NEMA ICS 16", 2001.
- [56] STEPPERONLINE, "STEPPER MOTOR 11HS18-0674S", 2020.
- [57] STEPPERONLINE, "STEPPER MOTOR 11N13S0754QE-150RS", 2021.
- [58] WANTAI MOTOR, "57BYGH420-2", 2015.
- [59] P. Burgess, "Adafruit GFX Graphics Library", 2021.
- [60] Coppelia Robotics, "CoppeliaSim User Manual", 2022.
- [online]:<https://www.coppeliarobotics.com/helpFiles/index.html> (acceso 2021).
- [61] S. Chakraborty and S. Aithal, "A Custom Robotic ARM in CoppeliaSim", *International Journal of Applied Engineering and Management Letters*, pp. 38-50, 2021. DOI: 10.47992/IJAEML.2581.7000.0091.

Anexos 1. TABLA DE COSTES

	Nombre	Precio €) sin IVA	Precio €) con IVA	Unts.	Total (€) sin IVA	Total con IVA(€)	Observaciones
Mecánica	Rodamiento de bolas (40mm,68mm)	8,65	10,47	2	17,30	20,93	es.rs-online.com/web/p/rodamientos-de-bola/1883287
	Rodamiento de bolas (8mm)	1,98	2,40	2	3,96	4,79	es.rs-online.com/web/p/rodamientos-de-bola/6189957
	Rodamiento de bolas (50mm,80mm)	13,92	16,84	2	27,84	33,69	es.rs-online.com/web/p/rodamientos-de-bola/6190604
	Rodamiento de bolas (25mm,37mm)	5,33	6,45	1	5,33	6,45	https://es.rs-online.com/web/p/rodamientos-de-bola/6190452
	Varilla Aluminio 8 mm ,1 m	2,85	3,45	1	2,85	3,45	www.bricodepot.es/varilla-redonda-108922
	Tubo de aluminio Anodizado 8 mm ,1m	2,59	3,13	1	2,59	3,13	www.bricomart.es/tubo-redondo-aluminio-anodizado-8-x-1-mm-2-m-10256722.html
	Correa síncrona 800 5M 15	11,97	14,48	1	11,97	14,48	https://es.rs-online.com/web/p/correas-de-distribucion/2042959
	Correa T5 de 500mm	8,31	10,06	2	16,62	20,11	https://es.rs-online.com/web/p/correas-de-distribucion/1755218
	Tornillos y tuercas	12,40	15	1	12,40	15,00	Coste aproximado
	Sistema de avance tornillo + tuerca	13,88	16,79	1	13,88	16,79	www.amazon.es/Accesorios-impresoras-trapezoidal-inoxidable-impresora/dp/B07WV44W9K?th=1
	Plancha PMMA 500mm x 500mm	10,74	13,00	3	32,23	39,00	https://www.amazon.es/dp/B07PJZSSR3
	PLA Sunlu	19	22,99	2	38,00	45,98	https://es.rs-online.com/web/p/materiales-para-impresion-3d/2035640
	TPU EFIL 85A	13,18	15,95	1	13,18	15,95	www.3dfils.com/es/efil-tpu-85a/filamento-3d-flexible-tpu-85A-antracita.html
	Acoplador flexible 5x6	2,93	3,55	1	2,93	3,55	www.impresoras3d.com/producto/acople-flexible-acoplamiento-flexible-5x8-mm
	Guías lineales	1,10	1,33	4	4,40	5,32	www.amazon.es/08-Rodamiento-deslizamiento-Rodamientos-A8-A6-Makerbot-MK8-creality/dp/B079P717LJ
	Perfil 44x44 mm x 1m XCBM	26,7	32,31	3	80,10	96,92	https://es.rs-online.com/web/p/tubos-y-perfiles/2489742
	Tuercas para perfil de ranura 11m y tornillo 6M	18,21	22,03	1	18,21	22,03	https://es.rs-online.com/web/p/componentes-de-conexion/2489843
	Escuadra de conexión de perfil 44 mm y 6M	6,25	7,56	4	25,00	30,25	https://es.rs-online.com/web/p/componentes-de-conexion/2490433
Electrónica	Sensor de Fuerza Resistivo	7,14	8,64	3	21,42	25,92	https://es.rs-online.com/web/p/sensores-de-fuerza/2122477
	Final de Carrera mecánico vertical	1,32	1,60	3	3,96	4,79	https://es.shop.ledger.com/products/mechanical-endstop-vertical
	cableado	15,73	19,03	1	15,73	19,03	https://es.rs-online.com/web/p/motores-dc/8293522
	Arduino MKR 1010	27,9	33,76	1	27,90	33,76	https://store.arduino.cc/products/arduino-mkr-wifi-1010?selectedStore=eu
	TMC5160 SilentStepStick	13,22	16,00	4	52,88	63,98	https://www.digikey.es/es/products/detail/trinamic-motion-control-gmbh/TMC5160SILENTSTEPSTICK/9990281
	PCB + conectores	8,26	10	1	8,26	10,00	Coste aproximado
Eléctrica	Motor DC Nema 23	39,6	47,95	2	79,26	95,90	https://es.rs-online.com/web/p/motores-paso-a-paso/5350439
	Motor DC Nema 11	13,9	16,84	1	13,92	16,84	https://www.omc-stepperonline.com/nema-11-bipolar-1-8deg-9-5ncm-13-5oz-in-0-67a-4-6v-28x28x45mm-4-wires-en-gb.html
	Motor DC Nema 11 non-captive + tornillo de avance de 15 cm	33,59	40,64	1	33,59	40,64	https://www.omc-stepperonline.com/nema-11-non-captive-34mm-stack-0-75a-lead-4-877mm-0-192-length-150mm.html
TOTAL(€) (SCARA + Base)					538,78	651,92	
TOTAL(€) (SCARA sin base)					415,47	502,72	

Hay que tomar en cuenta que muchos de los elementos comprados no fueron usados por completo y por ello el coste se reduciría en cierta cantidad.

Anexos 2. ESQUEMÁTICOS PCB



Anexos 3. IMÁGENES ADICIONALES

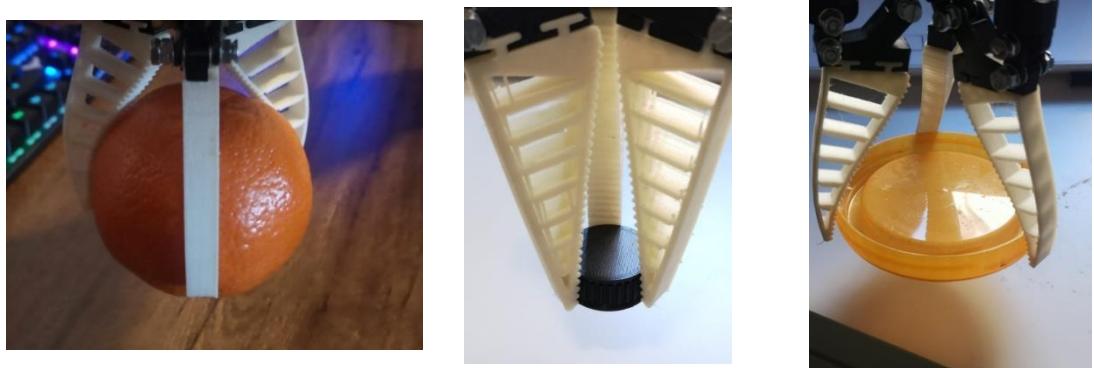


Figura A3. 1 Pruebas de sujeción con prototipo



Figura A3. 2 Pruebas SG primera versión

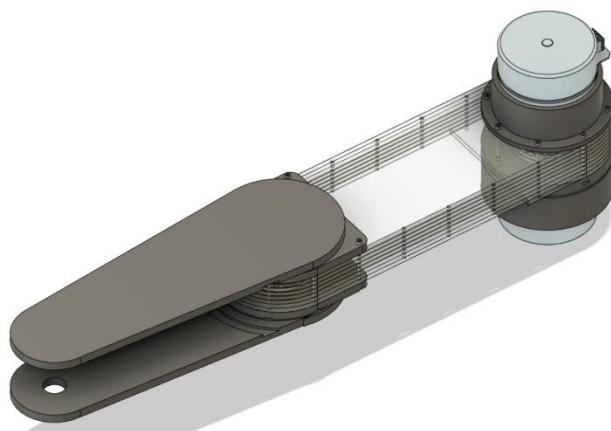


Figura A3. 3 Diseño descartado SCARA

Este diseño cuenta con motores EC90fl.

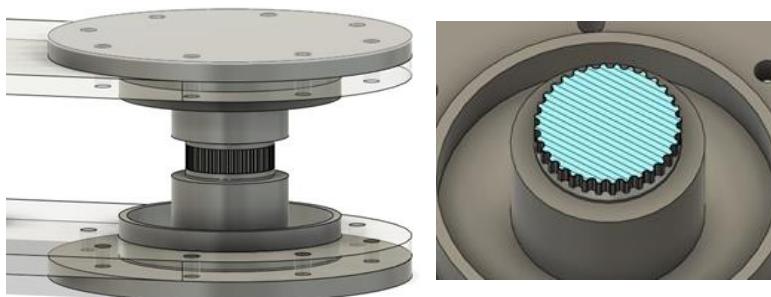


Figura A3. 4 Poleas V2 cuerpo SCARA

Se representa el diseño de polea GT2 para transmisión de potencia a ESB2. A la derecha se ve un corte medio de la pieza. A la izquierda una imagen en perspectiva.



Figura A3. 5 Versión de ESB3 con perfil de aluminio

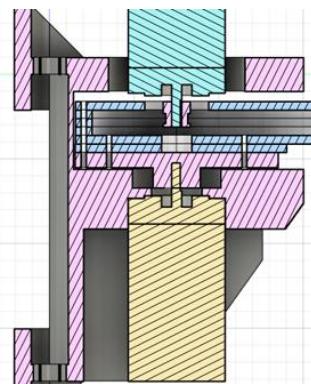


Figura A3. 6 Base de versión 4 SCARA

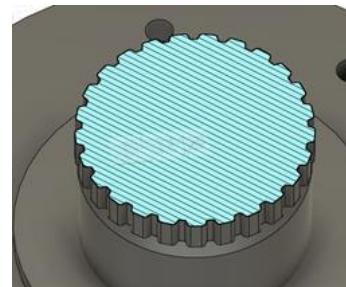


Figura A3. 7 Primera Versión de polea GT5

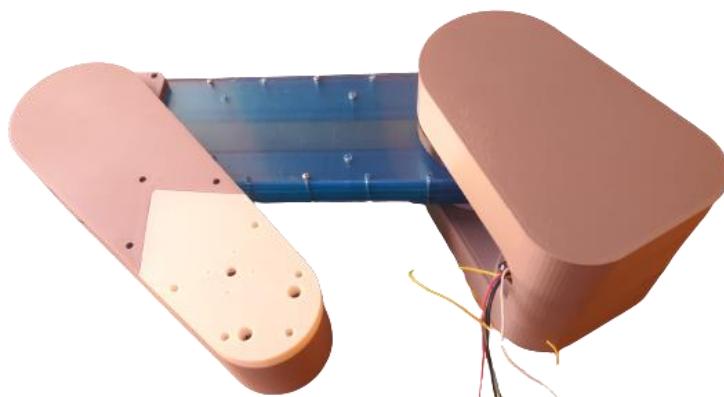


Figura A3. 8 Primer prototipo fabricado de SCARA

En ambas imágenes se representa el diseño de Figura 4.18



Figura A3. 9 Piezas V5 completas
A la izquierda se ve BS y a la derecha BS_BOT



Figura A3. 10 Piezas cortadas con láser para ESB1_PRC antes de tratamiento con acetona



Figura A3. 11 Problema de bajo flujo en impresión

Se puede ver que la pared de la pieza está separada del cuerpo por problemas de poco flujo. Esto provoca numerosos inconvenientes de funcionalidad y estructura.



Figura A3. 12 Modificación Ender 3D Pro y problema de Filamento flexible

Se observa una pieza impresa gris que corresponde con la modificación necesaria para control de tensión en el extrusor. Se ve también el problema que esta tensión puede causar en caso de ser muy elevada, haciendo que no haya más salida de filamento y provocando una pérdida total de la impresión.

Anexos 4. DATA SHEET: SCARMOV

SCARMOV

Fast Picker – Robot V1.0



1. FIABILIDAD-FLEXIBILIDAD-DESEMPEÑO

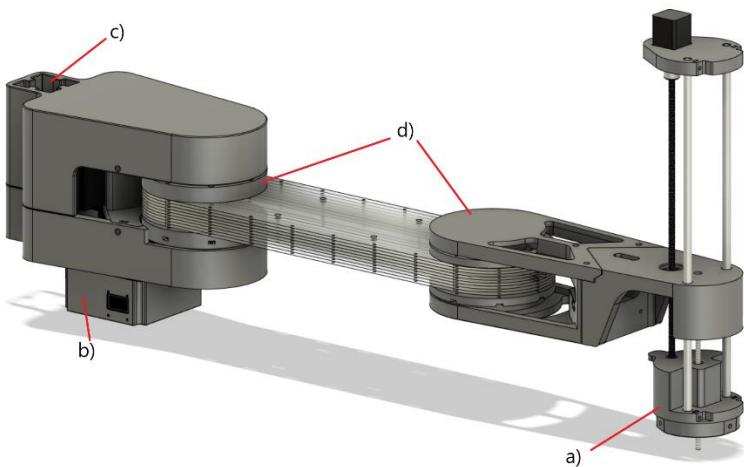


Figura A3. 13 Cuerpo de SCARA

- a) Adaptación de gripper genérico
- b) Caja de control con pantalla OLED para HMI
- c) Sujeción a perfil de aluminio
- d) Articulaciones robustas

Las principales ventajas que ofrece este robot son:

- Diseño pensado para tener un amplio rango de movimiento (Figura A3. 14)
- El actuador tiene un acoplamiento que permite la adaptación de numerosos tipos de gripper (Figura A3. 15)
- Capacidad de agarre adaptado y velocidad variable que lo hacen la opción ideal para numerosas aplicaciones industriales
- Velocidad de movimiento de hasta 15 cm/s con buena precisión (Figura A3. 13)
- Conforma una solución flexible y de fácil uso para trabajar junto con las personas
- Sujeción con perfil XCBM 1X44 lo cual le permite adaptarse en cualquier estructura fija o móvil que cuente con este perfil (Figura A3. 12)

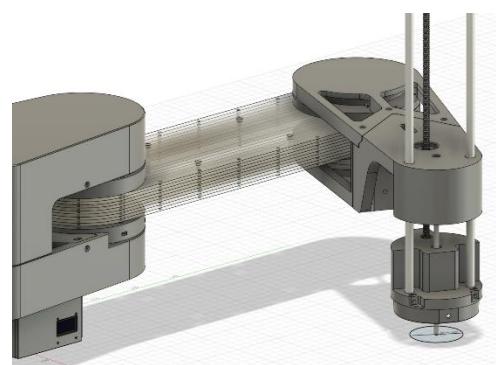
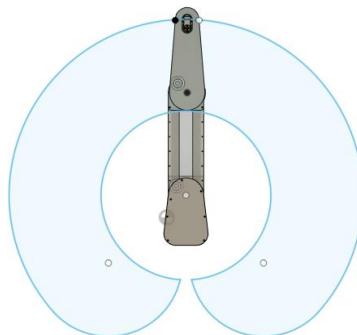
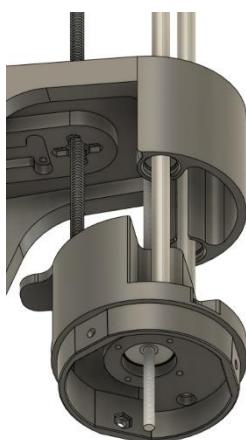
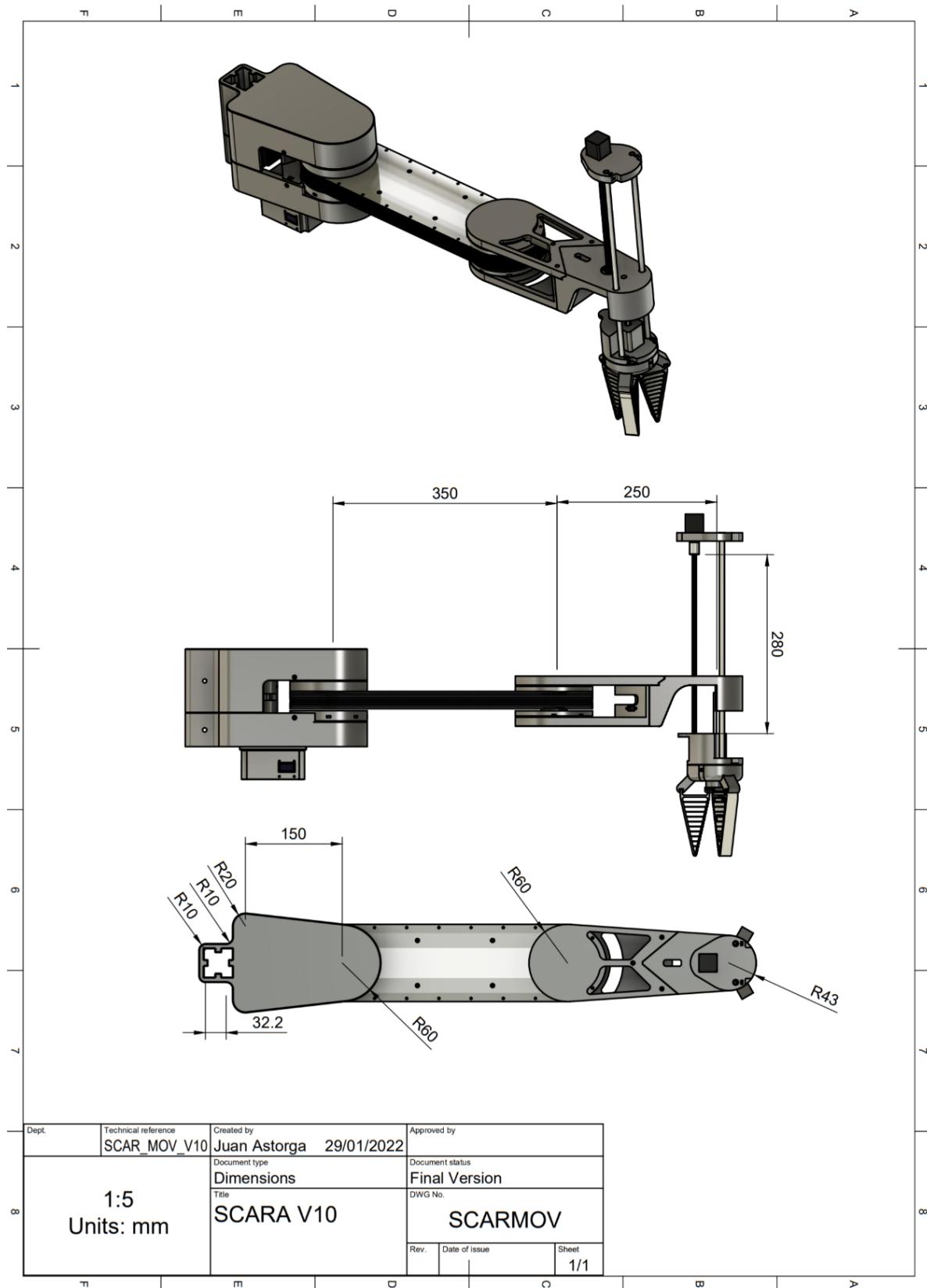


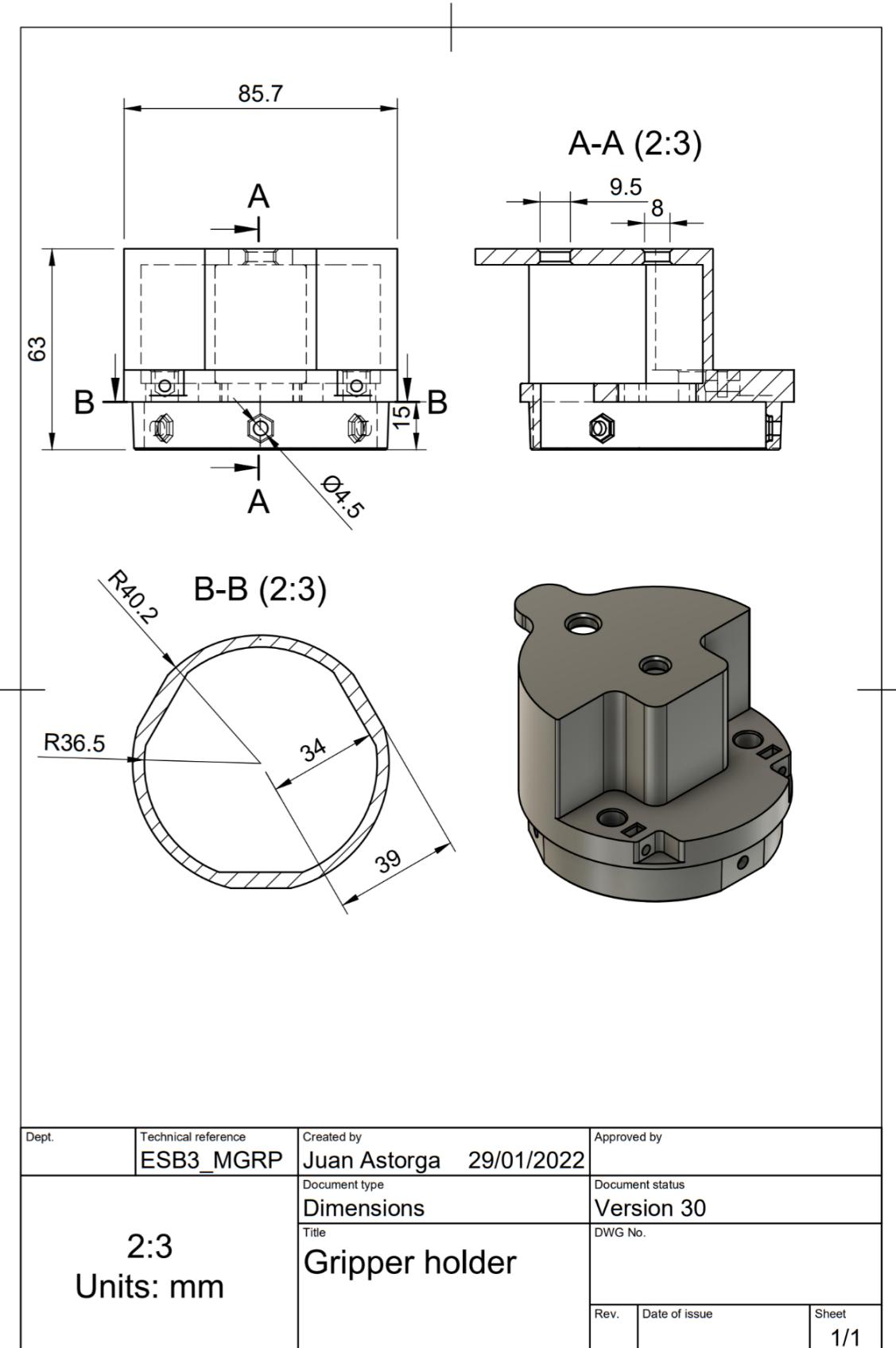
Figura A3. 16 Adaptador de Gripper Figura A3. 15 Amplio rango de movimiento

Figura A3. 14 Rápida acción con precisión

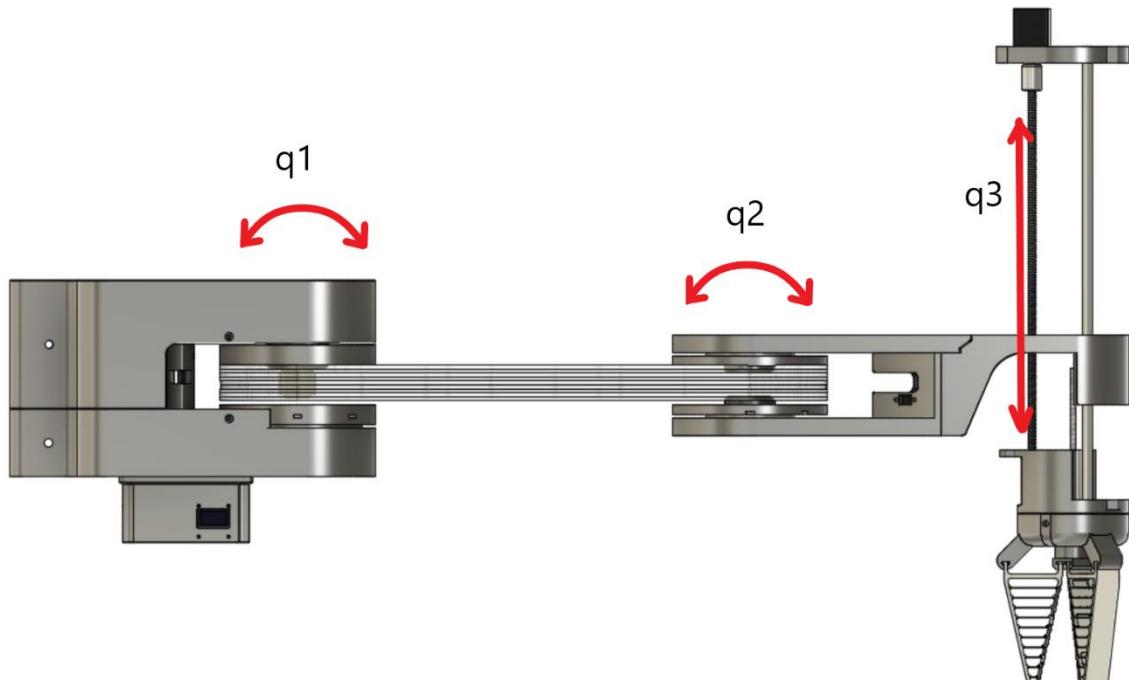
2. DIMENSIONES CUERPO DEL ROBOT



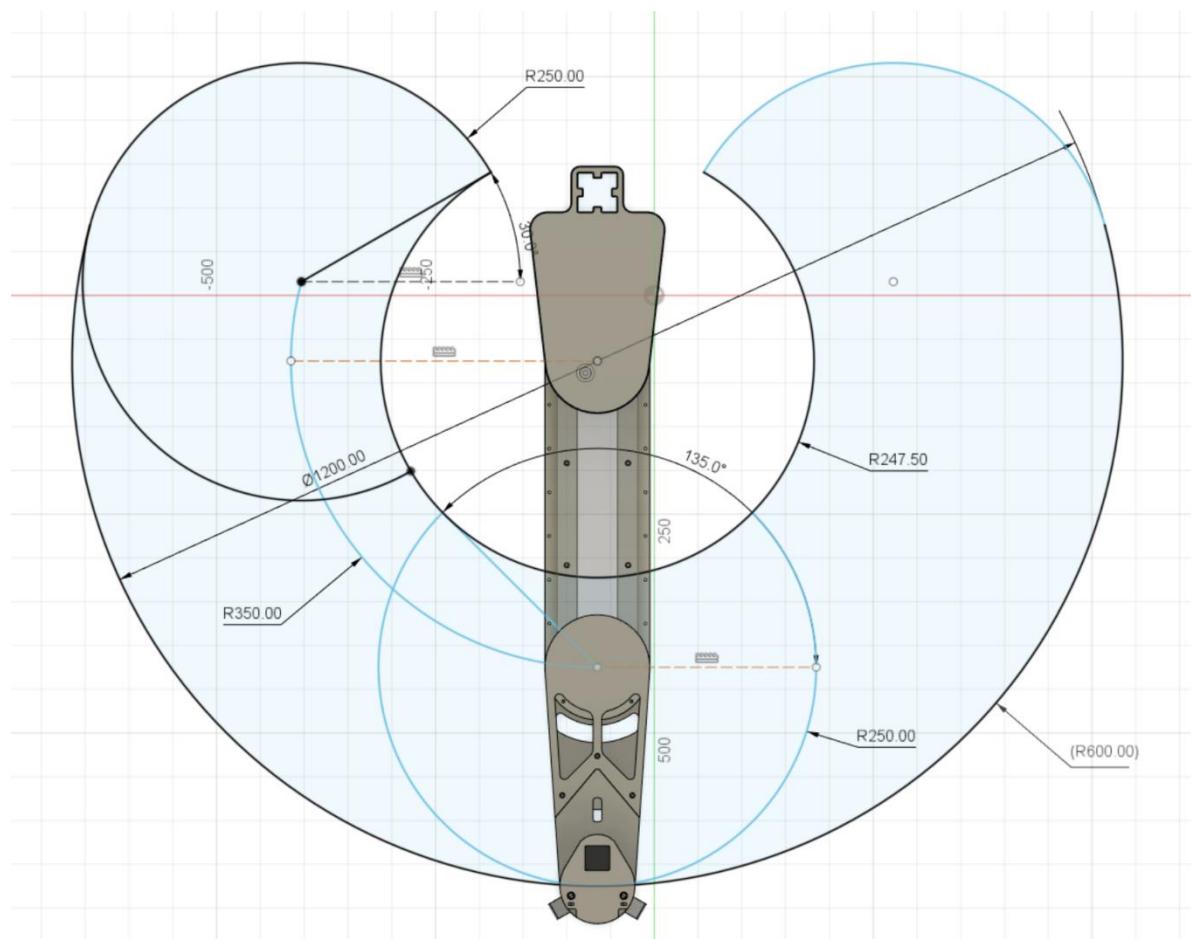
3. DIMENSIONES ACOPLADOR GRIPPER



4. EJES DE MOVIMIENTO



5. RANGO DE TRABAJO



6. ESPECIFICACIONES TÉCNICAS

Voltaje de alimentación	12 V
Corriente Nominal	2.5 A
Comunicación	WiFi
Número de sensores LS	3
Carga Máxima (Kg)	0.4
Carga Nominal	0.3kg
Alcance (entre eje 1 y 3)	600mm
Grados de libertad	3
Peso(Kg)	6
Rango Eje 1	230°
Rango Eje 2	270°
Rango Actuador 3	280 mm