Especificación de Requisitos
para el
Trabajo de Fin de Grado sobre el manipulador pArm

Javier Alonso Silva
Mihai Octavian Stanescu
José Alejandro Moya Blanco

Universidad Politécnica de Madrid
Ingeniería de Computadores
Trabajo de Fin de Grado
Tutor: Norberto Cañas de Paz

Madrid, 27 de enero 2020

Índice general

Historial de versiones

Revisión	**Fecha**	**Autor(es)**	**Descripción**
1.0	27.01.2020	J. Alonso, M. Stanescu, A. Moya	Primera especificación de los requisitos del proyecto.
1.1	04.02.2020	J. Alonso, M. Stanescu, A. Moya	Nuevos requisitos, incluídos casos de uso – eliminado de la especificación el cómo se realizan ciertos procedimientos.

Capítulo 1

Introducción

El μArm es un brazo robótico creado por la compañía UFACTORY[1] el cual se ha diseñado con propósito didáctico y, a su vez, creacional.

En la actualidad, se puede obtener uno a través de su página web o de proveedores externos, pero no está previsto fabricar más, por lo que en un tiempo estará fuera de existencias.

Debido a su propósito didáctico, todos los recursos sobre el manipulador son de código libre, por lo que resultan accesibles a cualquiera que los necesite. Entre otros, se encuentran[2]:

- *Firmware* del μArm Swift Pro.
- *Software Development Kit* (SDK) de Python para el μArm Swift Pro.
- *Firmware* que maneja el controlador del brazo.
- *Robot Operating System* (ROS) para el μArm Swift Pro.
- Distintos ejemplos para toda la gama de brazos robóticos.
- *μArm Creator Studio*.
- Visión esquemática de las conexiones de la placa Arduino.
- Modelos 3D del brazo robótico.
- Guías de usuario, desarrollador y especificaciones técnicas.

Aprovechando dichos recursos, se pretende desarrollar un brazo robótico basado en el μArm que esté impreso en 3D y sea controlado por un microcontrolador en conjunción con un ordenador cualquiera. Aprovechando los recursos provistos por UFACTORY, se busca que el brazo desarrollado sea más barato de construir (frente a los casi 800€ que cuesta el original) y que pueda ser desarrollado por cualquiera con acceso a Internet y a los recursos necesarios, a saber, una impresora en 3D y un *software* (SW) de impresión en 3D.

[1] `https://www.ufactory.cc/#/en/uarmswift`

[2] todos los elementos descritos se encuentran disponibles tanto en GitHub como en la web de UFACTORY

1.1. Propósito

El propósito de este documento es establecer un punto de partida claro y conciso que permita empezar el desarrollo del brazo robótico sabiendo los puntos primordiales del mismo. A su vez, también pretende establecer ciertos puntos que se consideran importantes e incluso necesarios para poder continuar el desarrollo del sistema en un futuro, implementando nuevas funciones o arreglando errores que pudieran existir.

Este documento está dirigido a ingenieros que quieran llevar a cabo su propia implementación del brazo robótico o que quieran conocer la estructura en la que se basa el proyecto, así como las necesidades del mismo y las adiciones extraordinarias que se han incluido. A su vez, se pretende que sea accesible a cualquiera que pretenda iniciarse en el mundo de la robótica y que busque estudiar y aprender sobre el brazo robótico.

1.2. Alcance

El objetivo principal de este proyecto fin de grado es construir una brazo robótico similar al manipulador μArm, al cual se le ha asignado el nombre *PLA – Arm* (pArm).

Este brazo robótico debe ser capaz de moverse libremente dentro de su campo de movimiento, el cual está limitado por su estructura física. Además, el *p*Arm debe ser capaz de coger, transportar y depositar objetos de poco peso y, en consecuencia, debe ser capaz de describir trayectorias previamente planificadas o calculadas en el momento.

Es importante destacar que, dado que el brazo robótico *p*Arm no está sensorizado, este no será capaz de moverse de forma completamente autónoma ni de imitar movimientos realizados por el usuario.

Cabe destacar que el brazo robótico está controlado mediante un microcontrolador. Sin embargo, las instrucciones de movimiento y trayectorias no se computan, en principio, en el mismo sino en un ordenador auxiliar.

Debido a la estructura física, tamaño y materiales de fabricación, el *p*Arm no es un brazo robótico pensado para la realización de tareas industriales ni para el transporte de cargas pesadas.

En relación a lo anteriormente mencionado, la aplicación principal del *p*Arm es didáctica, dado que se busca construir un brazo robótico económico y sencillo que facilite la introducción de los usuarios a este tipo de tecnologías.

1.3. Definiciones, siglas, y abreviaturas

SDK *Software Development Kit*

ROS *Robot Operating System*

SW *software*

HW *hardware*

pArm *PLA – Arm*

USB *Universal Serial Bus*

ODS Objetivos de Desarrollo Sostenible

OS *Open–Source*

OH *Open–Hardware*

S1 Sistema 1 – ordenador

S2 Sistema 2 – *p*Arm

GUI *Graphical User Interface*

GTK *GIMP Toolkit*

SoC *System On Chip*

PWM *Pulse–Width Modulation*

GPIO *General Purpose Input/Output*

UART *Universal Asynchronous Receiver–Transmitter*

RAM *Random Access Memory*

ADC *Analog–Digital Conversor*

- SDK – colección de herramientas SW disponibles para instalar en un único paquete.
- ROS – conjunto de librerías SW que ayudan a construir aplicaciones para robots.
- *Firmware* – SW programado que especifica el orden de ejecución del sistema.
- *Graphical User Interface* (GUI) – siglas que significan "Interfaz Gráfica de Usuario" (en castellano).
- *GIMP Toolkit* (GTK) – biblioteca de componentes gráficos multiplataforma para desarrollar interfaces gráficas de usuario.
- *System On Chip* (SoC) – tecnología de fabricación que integra todos o gran parte de los módulos en un circuito integrado. Un ejemplo muy común es la placa base de un teléfono móvil, la cual es un SoC que integra todos los componentes (antenas, conversores, sensores, etc.).
- *Pulse–Width Modulation* (PWM) – señal en la cual se modifica el ciclo de trabajo de una señal periódica, para transmitir información por un canal de comunicaciones o para controlar la cantidad de energía que se envía a una carga.

- *General Purpose Input/Output* (GPIO) – pin genérico cuyo comportamiento puede ser controlado en tiempo de ejecución.
- *Universal Asynchronous Receiver–Transmitter* (UART) – estándar de comunicación dúplex simultáneos.
- Dúplex – término que define a un sistema que es capaz de mantener una comunicación bidireccional, enviando y recibiendo mensajes de forma simultánea.
- Widget – la parte de una GUI (interfaz gráfica de usuario) que permite al usuario interconectar con la aplicación y el sistema operativo.
- *Random Access Memory* (RAM) – memoria principal del ordenador, donde se guardan programas y datos, sobre la que se pueden efectuar operaciones de lectura y escritura.
- *Deep–Sleep* – estado de un microcontrolador por el cual consume muy poca cantidad de energía.
- *bit* – unidad mínima de información de una computadora.

1.4. Visión global

En las siguientes páginas se pasa a explicar los distintos detalles del sistema que debe construirse, estructurado:

- Perspectiva del producto.
- Funciones del producto.
- Características del producto.
- Restricciones.
- Supuestos y dependencias.
- Requisitos propuestos.

En el punto “Requisitos específicos” (3) se detallan los requisitos específicos del sistema.

Capítulo 2

Descripción general

2.1. Perspectiva del producto

El *p*Arm se basa en el trabajo inicial del μArm, no utilizando directamente lo desarrollado por la empresa UFACTORY sino aprovechando el trabajo ya realizado y los recursos disponibles para estudiarlos. Si bien es cierto que el μArm ya es un sistema avanzado y capaz, como se explicó en la Introducción (1), se pretende estudiar y desarrollar un sistema propio el cual pueda servir para ayudar y facilitar la entrada a este tipo de tecnologías a otras personas, haciéndolo comprensible y, aprovechando la tecnología de la impresión en 3D, fabricable por uno mismo.

Además, en relación a los Objetivos de Desarrollo Sostenible (ODS), con el desarrollo de este sistema se pretende trabajar en:

4 - Educación de Calidad[1].

7 - Energía Asequible y No Contaminante[2].

10 - Reducción de las desigualdades[3].

Para el primero, se tiene en cuenta que el producto se desarrollará siguiendo las iniciativas *Open–Source* (OS) y *Open–Hardware* (OH), las cuales facilitan el acceso a la información a cualquiera que la requiera. Además, se facilitará el desarrollo al completo detallado y explicado, con la resolución de los problemas pertinentes y el porqué de ella.

Para el segundo, el *p*Arm utilizará la electricidad como fuente de energía, evitando así otras más contaminantes como las producidas por combustibles fósiles. En añadido, se trabajará para que el consumo de energía sea el menor posible, permitiendo así un mayor tiempo de uso con la misma fuente de alimentación y no abusando de los recursos de los que se disponen.

[1] https://www.un.org/sustainabledevelopment/es/education/
[2] https://www.un.org/sustainabledevelopment/es/energy/
[3] https://www.un.org/sustainabledevelopment/es/inequality/

Finalmente, se pretende hacer que el *p*Arm tenga un coste bajo, permitiendo así el acceso a los recursos y los procesos de fabricación a todo el mundo que pudiera estar interesado y que disponga de la cantidad mínima necesaria para poder poner en funcionamiento el brazo robótico.

Por otra parte, el *p*Arm es dependiente de otro sistema que lo controle, ya que no se plantea como sistema autónomo. Será necesario disponer de un sistema que sea capaz de comunicarse por puerto serie (*Universal Serial Bus* (USB)) y que permita la ejecución de aplicaciones Python. De ahora en adelante, se denominará "Sistema 1 – ordenador (S1)" al equipo que controla al *p*Arm; y "Sistema 2 – *p*Arm (S2)" al brazo robótico en sí.

Para este proyecto, se ha de desarrollar el SW que se ejecutará en el S1 y tanto el SW como el *hardware* (HW) que irán en el S2, así como la estructura del mismo.

2.1.1. Interfaz del sistema

En un principio, el sistema estará dividido en dos módulos:

S1

El S1 consiste en un equipo el cual controlará el brazo robótico (S2). Para ello, según el planteamiento inicial del proyecto, será necesario que permita la ejecución de aplicaciones Python las cuales tengan GUI.

En lo referente al sistema operativo, al ser una aplicación en Python la cual es multiplataforma, no se define ninguna restricción respecto al mismo.

Finalmente, se plantea la conexión con el S2 utilizando el puerto serie USB, por lo que también será necesario que el equipo anfitrión S1 disponga de una conexión de ese estilo.

En resumen (ver la tabla 2.1):

Componente	**Función**	**Restricciones**
Sistema Operativo	Hospedar y ejecutar la aplicación Python que controlará el brazo robótico.	Debe poder ejecutar aplicaciones Python con GUI, por ejemplo, GTK.
UART (USB)	Permitir la comunicación con el sistema S2 en modo dúplex.	Velocidad adaptable (*baud–rate).*
Python	Control del sistema S2 y monitorización del estado del mismo.	Versión Python $\geqslant$ 3.6.*

Tabla 2.1: Requisitos del sistema S1.

En principio, no será necesaria la conexión a Internet, pero tampoco se descarta el uso de la misma en el proyecto a la hora de poder recibir actualizaciones o en lo referente a futuras mejoras.

En añadido, pese a que no se restringe, se sugiere que el equipo que hospede la aplicación disponga de un procesador multinúcleo (para realizar el cálculo matricial más rápido) así como suficiente memoria RAM para poder manejarlas.

S2

Para el sistema S2 no se ha pensado en ningún microprocesador ni SoC en particular, pero se han contemplado algunos que cumplen con las características requeridas (ver la tabla 2.2).

Será necesario que el circuito escogido disponga de entrada serie, UART, para la comunicación con el S1. Debido a la característica descrita en la tabla 2.1 sobre la conexión USB, no será estrictamente necesario que la velocidad sea adaptable (ya que se asume que se adaptará en el S1); sin embargo, sí será requisito fundamental que la conexión sea dúplex.

Por otra parte, el microcontrolador deberá poder modular señales PWM para controlar los distintos motores de los que dispondrá el brazo. Sin embargo, en caso de que finalmente el *chip* escogido no disponga de dicha modularización, se podrán usar motores los cuales cuenten con un *driver* que permitan controlarlos usando señales digitales y/o analógicas.

Finalmente, a raíz del punto anterior, será imprescindible que el sistema escogido tenga la capacidad de controlar señales digitales y analógicas, permitiendo así mayor versatilidad en el desarrollo del producto final.

Teniendo en cuenta lo anterior, se plantea el uso de los siguientes dispositivos (ver tabla 2.2):

Placa	**Ventajas**	**Desventajas**	**ID *RS–Online* y precio**
ESP8266	SoC bastante barato (5€) con conexión WiFi y modo de bajo consumo	Señal PWM generada por SW; poca cantidad de GPIO (6).	124-5505 – 19,29€
ESP32	SoC con procesador de dos núcleos que permite comunicaciones WiFi y Bluetooth	No cuenta con GPIO pero permite la comunicación mediante el protocolo I^2C.	188-5441 – 25,29€
PIC16F18326-I/P	Microcontrolador de 8 bits de baja potencia de consumo y bajo precio con capacidad de modularizar hasta dos señales PWM y con más memoria RAM que otros componentes de su familia. Finalmente, cuenta con bastantes salidas GPIO, suficientes como para añadir más componentes al sistema.	No está integrada en una placa (SoC) por lo que habría que hacer toda la lógica del diseño HW. No dispone de conexiones de red (aunque no son necesarias) y la capacidad de cómputo, en comparación con las otras propuestas, es menor.	124-1554 – 1,375€

Tabla 2.2: Posibles *chips* que se han planteado para el proyecto.

2.1.2. Interfaz de usuario

El usuario final del producto solamente interactuara de manera directa con el S1. Para que esta interacción sea posible, se desarrollara un panel de control que permita al usuario definir movimientos que el robot deberá realizar. El panel de control se mostrará en una sola pantalla y permitirá al usuario, mediante una interfaz gráfica sencilla, mover de manera independiente cada uno de los motores del robot, o bien mediante el uso del ratón, describir trayectorias que el robot realizara en tiempo real replicando el movimiento del ratón.

2.1.3. Interfaz *hardware*

El S2 deberá de tener una interfaz con el HW tal que se puedan rotar los motores que controlan el movimiento del robot. Por tanto, el microcontrolador que sea elegido para el proyecto final deberá permitir dicho control en base a las características de los motores. Por otro lado, si los motores devuelven su posición, el microcontrolador deberá ser capaz de recibir este dato para poder enviarlo al S1.

2.1.4. Interfaz de comunicaciones

Debido a la naturaleza del proyecto deberá existir una interfaz de comunicaciones que permita el envío y recepción de datos desde el S1 al S2 y viceversa. Según lo indicado en el apartado Descripción General (2), esta comunicación se hará a través del estándar UART mediante un conector USB. La razón de elegir este estándar de comunicación se debe a que permite una conexión bidireccional simultanea a través del mismo puerto y por tanto permite controlar el robot a la vez que se recibe información referente al estado de este.

Además con este estándar la comunicación es serie y cumple con las limitaciones de diseño que se han impuesto en la recepción de datos en el S2.

2.1.5. Memoria

- En cuanto a la memoria de programa se refiere, el microcontrolador del sistema S2 debe tener una memoria suficientemente grande como para poder albergar el SW que recibirá datos del S1 y que actuará sobre los motores en base a estos.
- Por otro lado, la memoria RAM deberá ser tal que permita realizar las operaciones necesarias para ejecutar los movimientos. En caso de que se considere posible la completa implementación de la lógica en el sistema S2, el microcontrolador deberá tener memoria principal suficiente como para conseguir realizar los cálculos matriciales relacionados con los distintos movimientos.

Por otro lado, la memoria principal deberá ser tal que permita las operaciones necesarias para realizar los movimientos en tiempo de ejecución. Cabe destacar que una de las prioridades principales del equipo es ser efectivos en cuanto a la ocupación de memoria del código que desarrolla. Por ello, se buscará reducir el tamaño de este al mínimo manteniendo las funcionalidades necesarias para cumplir los objetivos establecidos.

2.1.6. Operaciones

Los usuarios deberán desempeñar acciones tales que generen los movimientos deseados en el brazo. Estas acciones pueden implicar interactuar con los *widgets* presentes en el panel de control o bien efectuar movimientos con el ratón para que el robot los desempeñe directamente.

2.2. Funciones del producto

Las funcionalidades principales del brazo robótico han sido descritas de forma introductoria en apartados anteriores de este documento.

En general, existen dos funcionalidades principales que caracterizan tanto al *p*Arm como al sistema de control del mismo:

- La funcionalidad principal del brazo robótico S2 es la de realizar movimientos dentro de su campo de movimiento y describir trayectorias previamente planificadas o calculadas en el momento. Mediante este movimiento, se pretende transportar objetos de poco peso.

  Cabe destacar que el brazo robótico S2 recibe ordenes del S1 y procesa las mismas utilizando el microcontrolador que posee, por lo tanto el computo principal se realiza en S1.

- El sistema de control en del brazo robótico S2 ofrece la funcionalidad principal de planificar trayectorias y controlar el movimiento del brazo. Este sistema se muestra al usuario mediante una interfaz gráfica en S1, la cual permite al usuario controlar el movimiento del brazo mediante la modificación de sus parámetros.

2.3. Características del usuario

El sistema de control ejecutado en S1 ofrecerá una interfaz gráfica que permitirá al usuario interactuar con los parámetros del brazo robótico S2 y, por lo tanto, permitirá al mismo controlar el movimiento del robot así como la establecer la descripción de ciertas trayectorias.

Dado que el objetivo el proyecto es ofrecer un sistema didáctico, amigable y fácil de usar, no se imponen requerimientos específicos sobre el usuario en cuanto a conocimientos técnicos sobre programación, HW, electrónica o matemáticos.

El usuario debe estar familiarizado con la interacción y el uso básico de aplicaciones de escritorio para poder interactuar de forma correcta con el sistema de control del brazo.

A pesar de no ser completamente necesario, es recomendable que el usuario esté familiarizado con la estructura física del robot, los movimientos que este puede realizar y los parámetros que se usan para controlar al mismo, ya que de esta forma el control del brazo robótico será más eficaz y seguro.

2.4. Restricciones

Dado que actualmente no se presenta ninguna limitación de presupuesto, se busca en el proyecto intentar reducir los costes todo lo posible, para permitir a cualquiera que pueda acceder a los recursos que se necesitan para desarrollar el proyecto. Por ello, se propone usar si es posible el PIC16F18326-I/P y montar la placa con los componentes necesarios.

En cualquier caso, como se ha mencionado anteriormente, es necesario que:

- Se provea de una interfaz UART para comunicación por puerto serie USB para comunicarse con S1.

- El sistema ha de consumir la menor energía posible, entrando en el modo de *deep–sleep* cuando fuera posible.

- La estructura de S2 ha de ser imprimible en 3D, permitiendo así replicarlo.
- El sistema S1 ha de poder ejecutar aplicaciones Python, siendo la versión de este superior a la 3.6.
- Además, el sistema S1 ha de tener capacidad de cómputo suficiente para realizar cálculos matriciales de forma efectiva. Para ello, se sugiere que el equipo disponga al menos de un procesador con dos núcleos y 512 MB de memoria RAM.
- Todo lo realizado en el proyecto ha de ser OS y OH, permitiendo así que cualquiera pueda acceder y estudiar el proyecto.

2.5. Supuestos y dependencias

Indiferentemente de la placa que finalmente se use, el sistema ha de tener tres motores: uno para la base, otro para el primer segmento del brazo robótico y el último para el segundo segmento. Además, para controlar el *end–effector* hará falta una conexión con el extremo del brazo que permita, por ejemplo, añadir un pequeño motor que permita la rotación del mismo (ver el manual de desarrollador de UFACTORY para más información).

Para ello, en la tabla 2.3 se muestran distintas propuestas de motores que podrían ser viables para el proyecto. Intentando cubrir las necesidades, se tienen en cuenta para este proyecto:

- Motor paso a paso: dispositivo electromecánico que convierte una serie de pulsos eléctricos en desplazamientos angulares. Esto permite realizar movimientos muy precisos, los cuales pueden variar de 1,8° hasta 90°. Además, presentan la ventaja de poder quedarse en una posición de forma estática.
- Servomotor: dispositivos de accionamiento para el control de la velocidad, par motor y posición. En su interior suelen tener un decodificador el cual convierte el giro mecánico en pulsos digitales. Además, suelen disponer de un *driver* el cual permite comandar los distintos controles mencionados al principio.

Nombre	**Tipo**	**Características**	**Código RS y precio**
Servomotor Parallax Inc.	Servomotor	▪ Voltaje entrada: $4\ V$ a $6\ V$. ▪ Conector de tres contactos. ▪ PWM a $50\ Hz$.	781-3058 – 16,01 €

Servomotor Faulhaber 9 W	Servomotor	▪ Par máximo: 9,5 mNm. ▪ Voltaje entrada: 6 V. ▪ Potencia nominal: 9 W. ▪ Conector MOLEX Microfit 3.0.	184-6932 – 186,73 €
Motor paso a paso bobinado unipolar	Motor paso a paso	▪ Precisión de 1,8°. ▪ Par de sujección: 70 mNm. ▪ Voltaje entrada: 6 V. ▪ Conexión de 6 cables.	440-420 – 30,29 €
Motor paso a paso híbrido	Motor paso a paso	▪ Precisión de 1,8°. ▪ Par de sujección: 1,26 Nm. ▪ Voltaje entrada: 2,5 V. ▪ Conexión de 4 cables.	535-0439 – 108,69 €

Motor paso a paso híbrido	Motor paso a paso	▪ Precisión de 0,9°. ▪ Par de sujección: 0,44 Nm. ▪ Voltaje entrada: 2,8 V. ▪ Conexión de 4 cables.	535-0401 – 66,72 €

Tabla 2.3: Lista de motores propuestos para el sistema S2.

2.6. Requisitos pospuestos

En esta sección se describen algunos requisitos del sistema que se postergan a futuras implementaciones o versiones del proyecto.

En el comienzo del proyecto se plantearon algunas funcionalidades y requisitos que, finalmente, se han decidido postergar a futuras implementaciones del proyecto, principalmente debido a su complejidad. En la siguiente se lista se presentan las mas relevantes, las cuales representan posibles mejoras futuras del *p*Arm:

- Implementación del sistema de control y planificación de trayectorias en el microcontrolador del *p*Arm, de esta forma se busca centralizar el computo en S2.
- Implementación de un sistema de descripción de trayectorias mediante imitación de movimientos realizados por el usuario, es decir, el usuario podría mover físicamente el *p*Arm y memorizaría dicha trayectoria para posteriormente describirla.
- Construcción e implementación de diversos tipos de *end–effector* para el *p*Arm, los cuales le dotarían de nuevas funcionalidades en cuanto a manejar objetos.
- Implementación de las estructura física del *p*Arm utilizando materiales metálicos para mejorar su resistencia y estabilidad. Junto con esta mejora, se podrían utilizar nuevos rotores para dotar al *p*Arm de una mayor capacidad de carga.

Capítulo 3

Requisitos específicos

3.1. Requisitos de la interfaz externa

3.1.1. Interfaz con el usuario

El S1 dispondrá de una interfaz que permitirá al usuario tener un control total sobre los movimientos del robot. Dicha interfaz dispondrá de 6 *sliders*. Tres de ellos que permitirán mover el *end–effector* en coordenadas cartesianas y los tres restantes variarán las coordenadas angulares de los motores generando cambios en la posición del *end–effector*. De esta manera, se consigue mover el *end–effector* con la máxima precisión que ofrecen los motores. Por otro lado, se permitirá al usuario cambiar del modo de funcionamiento en precisión al modo de funcionamiento en movimiento libre mediante un botón presente en la interfaz. Por último, la interfaz dispondrá de un botón que permita interactuar con el *end–effector*.

3.1.2. Interfaz *hardware*

El S2 esta formado por el brazo robótico *p*Arm y el microcontrolador que computa las instrucciones recibidas desde S1. Mediante dicho microcontrolador, S2 interactúa directamente con el HW. El microcontrolador realiza las labores de comunicación con S1, así como las labores de recepción y procesamiento de las instrucciones que controlan el movimiento del *p*Arm.

Tras la recepción y procesamiento de las diferentes secuencias de bits, las cuales son instrucciones, el microcontrolador genera señales de salida mediante sus pines, las cuales controlan el movimiento de cada uno de los motores, así como del *end–effector*. Cabe destacar que, en el caso de utilizar motores que proporcionen información sobre su posición angular actual, el microcontrolador debe recibir dicha señal y procesarla, enviando dicha información a S1.

Dependiendo del tipo de motores que se utilicen finalmente, el microcontrolador debe ser capaz de generar señales analógicas PWM, así como señales digitales de control.

3.1.3. Interfaz de comunicaciones

Las comunicaciones que se realicen entre el S1 y S2 están planteadas para utilizar UART como método de comunicación. Además, se mencionó como futura implementación poder hacer las comunicaciones entre ambos sistemas utilizando protocolos de red inalámbricos.

No se restringe la velocidad de transmisión (*baud–rate*), ya que se asume que S1 tendrá la posibilidad de adaptar su velocidad. Se escoge el USB como método para intercambiar la información debido a:

- Universalidad: los dispositivos cuentan con al menos una conexión USB.
- Energía: el USB provee 5 V al circuito que se conecta en el otro extremo. Además, la versión 2.0 del estándar, que es lo generalizado en microcontroladores, puede proveer hasta 500 mA al componente conectado.
- Simplicidad: no es necesario entender cómo se conectan los cables sino directamente conectar los extremos.

Para un correcto funcionamiento, la comunicación ha de ser bidireccional, en particular *full duplex*. De esta forma, se podrán recibir y enviar datos simultáneamente, pudiendo así conocer el estado del brazo robótico y actuar en consecuencia en caso de que se encuentre algún tipo de error o problema. Al utilizar el USB como método de comunicación este problema está subsanado, ya que va implícito en la definición del estándar.

3.2. Casos de uso

`0001`	Encender brazo robótico (S2)	
Descripción	El usuario deberá se capaz de encender el sistema del brazo robótico de manera independiente de la aplicación de control.	
Secuencia Normal	**Paso**	**Acción**
	1	El usuario interactúa con el sistema para encenderlo
	2	El sistema comprueba que los componentes están correctamente conectados.
	3	Si las comprobaciones son satisfactorias, el sistema continúa con su normal ejecución.
Excepciónes	**Paso**	**Acción**
	3	Si las comprobaciones no son satisfactorias el sistema activará un indicador luminoso.
Importancia	1	
Comentarios	Sin comentarios	

3.3. Requisitos funcionales

3.3.1. Requisitos *software*

S1

Mostrar pantalla de control

- Descripción: se muestra una pantalla donde serán situados los *sliders*.
- Entradas: ninguna.
- Salidas: ninguna.
- Errores: no se espera ningún error.

Mostrar pantalla informativa

- Descripción: se mostrarán distintos datos informativos sobre el estado del sistema S2, si se detecta.
- Entradas: los datos recibidos por el sistema S2, si estuviera conectado.
- Salidas: los datos recibidos debidamente interpretados por el sistema.
- Errores: se mostrará un aviso en caso de que el sistema S2 no se detecte o presente algún problema.

Mostrar botones de edición de pantalla

- Descripción: se muestran *widgets* de tipo botones clicables de minimizar, maximizar y cerrar pantalla.
- Entradas: ninguna.
- Salidas: ninguna.
- Errores: en caso de que algún proceso pudiera quedar bloqueado, los *widgets* permitirían el cierre inmediato de la aplicación.

Interactuar con botones de edición de pantalla

- Descripción: se permite interactuar con los botones que aparecen en la GUI para controlar la pantalla S2.
- Entradas: interacción del usuario con los botones de edición de pantalla.

- Salidas: cambios lógicos tales que se realicen los cometidos de cada botón.
- Errores: no se espera ningún error.

Mostrar *sliders*

- Descripción: se muestra por pantalla una serie de *widgets* de tipo *sliders*.
- Entradas: coordenadas angulares $(\theta_1, \theta_2, \theta_3)$ y coordenadas cartesianas (X, Y, Z).
- Salidas: mostrar por pantalla de manera gráfica el valor de las variables.
- Errores: no se espera ningún error.

Editar la posición de los *sliders*

- Descripción: se permite al usuario interactuar de manera directa e independiente con cada uno de los motores del brazo robótico, así como con las componentes de la posición cartesiana del *end–effector*. Este requisito permite al usuario tener una mayor precisión en cuanto a la posición que desea obtener para el brazo robótico.
- Entradas: variación por parte del usuario de la posición de los *sliders*.
- Salidas: modificar el valor numérico de las variables.
- Errores: no se espera ningún error.

Mostrar botón cambio de modo de funcionamiento del S1

- Descripción: se muestra por pantalla un *widget* de tipo botón clicable que permite cambiar el modo de funcionamiento.
- Entradas: al hacer clic se permite el cambio de modo.
- Salidas: mostrar el nuevo modo de funcionamiento.
- Errores: no se espera ningún error.

Interactuar con botón de cambio de modo de funcionamiento del S1

- Descripción: se permite al usuario interactuar con el botón para cambiar el modo de funcionamiento.
- Entradas: interacción del usuario con el botón.
- Salidas: modificación del modo de funcionamiento.
- Errores: no se espera ningún error.

Mostrar variables

- Descripción: se muestran por pantalla las variables que definen las posiciones cartesianas del *end–effector* y las angulares de los motores.
- Entradas: valores numéricos de las variables.
- Salidas: mostrar por pantalla dichos valores.
- Errores: no se espera ningún error.

Mostrar botón de control del *end–effector*

- Descripción: se muestra por pantalla un *widget* de tipo botón clicable que permite cambiar el estado del *end–effector*.
- Entradas: estado del *end–effector*
- Salidas: mostrar el estado de *end–effector*.
- Errores: no se espera ningún error.

Interactuar con botón de control del *end–effector*

- Descripción: se permite al usuario interactuar con el el botón para cambiar el estado de la pinza
- Entradas: interacción con el botón por parte del usuario.
- Salidas: modificar el estado del *end–effector*.
- Errores: no se espera ningún error.

Editar variables

- Descripción: se permite al usuario editar las variables numéricas de manera directa interactuando con el campo y escribiendo los valores numéricos deseados.
- Entradas: cambio por parte del usuario del valor numérico mostrado.
- Salidas: modificar el valor numérico de la variable.
- Errores: no se espera ningún error.

Comprobar variables

- Descripción: El sistema, al detectar cambios en alguna de las coordenadas, ya sean cartesianas o angulas se encarga de verificar que dichas coordenadas están en el rango de trabajo del robot. De no ser así se impide el movimiento del brazo para prevenir daños en su estructura o en los motores.
- Entradas: valor de las coordenadas angulares y cartesianas deseadas.
- Salidas: validación de dichas coordenadas.
- Errores: si alguno de los valores introducidos no es válido, se notificará al usuario de dicho error y se evitará que el S2 realice dichos movimientos.

Comunicación con el sistema S2

- Descripción: utilizando un lenguaje binario, se comunicarán las secuencias de órdenes desde el sistema S1 al sistema S2.
- Entradas: secuencia de movimientos representada como movimientos en puntos cartesianos o como rotaciones de las articulaciones.
- Salidas: secuencia binaria que especifica, en el sistema S2, los movimientos que se han de realizar.
- Errores: como se han comprobado los elementos con anterioridad, no se esperan errores.

Cálculo de coordenadas articulares

- Descripción: dadas unas coordenadas en forma cartesiana, el sistema S1 debe poder calcular las coordenadas articulares de cada una de las articulaciones del robot.
- Entradas: conjunto de tres puntos cartesianos (X, Y, Z).
- Salidas: conjunto de tres puntos articulares $(\theta_1, \theta_2, \theta_3)$.
- Errores: dada la configuración geométrica del robot, no se esperan errores.

Cálculo de coordenadas cartesianas

- Descripción: dadas unas coordenadas articulares, el sistema S1 debe poder obtener las coordenadas cartesianas en las que se encuentra en *end–effector*.
- Entradas: conjunto de tres puntos articulares $(\theta_1, \theta_2, \theta_3)$.
- Salidas: conjunto de tres puntos cartesianos (X, Y, Z).
- Errores: no se esperan errores.

Interpretación de los datos

- Descripción: el sistema S1 debe de poder entender e interpretar los datos recibidos por S2.
- Entradas: cadena binaria con información provista por S2.
- Salidas: mostrar, utilizando la interfaz de usuario, la información pertinente.
- Errores: no se esperan errores.

Protocolo de intercambio de información

- Descripción: Debe existir un protocolo de intercambio de información que defina la longitud, significado y estructura del las instrucciones o secuencias de bits que se transmiten mediante *UART*.
- Entradas: Ninguna.
- Salidas: Ninguna.

Encendido del sistema

- Descripción: el S1, al encenderse, debe comprobar si está conectado el S2 e inicializar aquellos recursos que serán necesarios.
- Entradas: ninguna.
- Salidas: ninguna.
- Errores: se esperan errores si hubiera algún tipo de corrupción de datos en los ficheros del programa o en los contenedores de datos.

Apagado del sistema

- Descripción: cuando el usuario cierra la interfaz, el sistema S1 debe desconectarse del todo y cesar cualquier comunicación que pudiera existir con S2. Además, deberá eliminar cualquier tipo de dato residual resultante.
- Entradas: el usuario cierra la aplicación.
- Salidas: ninguna.
- Errores: se esperan errores si no fuese posible cesar la comunicación debido a alguna política del sistema operativo. Se notificará al usuario al respecto.

S2

Encendido del sistema

- Descripción: cuando se inicie el sistema HW, debe iniciarse también el SW.
- Entradas: encendido del sistema HW.
- Salidas: activación del sistema SW.
- Errores: no se esperan errores.

Apagado del sistema

- Descripción: cuando se reciba la orden de apagado desde el S1, el sistema debe cortar toda comunicación con el mismo y apagarse lo antes posible.
- Entradas: orden de apagado desde S1.
- Salidas: ninguna.
- Errores: no se espera ningún error.

Interpretación de los valores binarios

- Descripción: tras recibir el HW una cantidad de bits que represente el tamaño designado para un determinado comando, los bits se interpretarán y se definirá de que comando se trata.
- Entradas: bits de control
- Salidas: comando para el sistema físico
- Errores: no se espera ningún error.

Comprobación de los dispositivos

- Descripción: el sistema deberá comprobar que detecta adecuadamente los dispositivos que están conectados al mismo.
- Entradas: conexiones con cada uno de los dispositivos.
- Salidas: ninguna.
- Errores: si no se detecta algún dispositivo se notificará al sistema S1 sobre dicha falta.

Comunicación con S1

- Descripción: utilizando los protocolos de comunicación, el sistema debe poder comunicarse con S1 correctamente.
- Entradas: valores recibidos por S1.
- Salidas: valores enviados hacia S1.
- Errores: no se esperan errores en la comunicación. En caso de existir, se reenviarían las tramas hasta que se recibieran por el S1.

Comprobación de la conexión

- Descripción: como el S2 es dependiente del S1, este necesitará comprobar que se encuentra activado para empezar a funcionar.
- Entradas: valor acordado por el sistema S1.
- Salidas: valor acordado con el sistema S2.
- Errores: en caso de no encontrar al sistema S1, el microcontrolador emitiría algún tipo de señal visual o acústica.

Recepción y envío de secuencias de bits entre S1 y S2

- Descripción: debe existir un medio de comunicación basado en UART entre S1 y S2 que permita el intercambio de secuencias de bits.
- Entradas: secuencia de bits o instrucción a enviar.
- Salidas: recepción correcta por parte del destinatario.
- Errores: no se espera ningún error.

Generación de señales PWM

- Descripción: el microcontrolador situado en S2 debe ser capaz de generar señales eléctricas analógicas PWM, las cuales serán usadas para controlar los motores.
- Entradas: instrucciones del sistema de control en S1
- Salidas: señal de control PWM que se corresponde con la respuesta a dicha instrucción.
- Errores: no se espera ningún error.

Generación de señales digitales

- Descripción: el microcontrolador situado en S2 debe ser capaz de generar señales digitales.
- Entradas: instrucciones del sistema de control en S1.
- Salidas: señal de control digital.
- Errores: no se espera ningún error.

Recepción y procesamiento de señales analógicas

- Descripción: El microcontrolador debe ser capaz de recibir mediante sus pines y procesar las señales analógicas provenientes de los motores, en caso de que estos informen sobre su posición angular. Estas señales deben ser recibidas y procesadas mediante el *Analog–Digital Conversor* (ADC) para poder ser tratadas a nivel de software.
- Entradas: señal analógica
- Salidas: señal procesada y convertida a datos tratables por el SW.
- Errores: no se espera ningún error.

Recepción y procesamiento de señales digitales

- Descripción: El microcontrolador debe ser capaz de recibir mediante sus pines y procesar las señales digitales. Estas señales deben ser recibidas y tratadas nivel de *software*.
- Entradas: señal digital
- Salidas: ninguna.
- Errores: no se espera ningún error.

Modo *Deep–Sleep*

- Descripción:el microcontrolador debe ser capaz de entrar en modo *Deep–Sleep*
- Entradas: ninguna.
- Salidas: ninguna.
- Errores: no se espera ningún error.

Encendido

- Descripción: se requiere un periodo de inicialización cuando se produce el encendido del sistema. Durante este periodo se realiza la inicialización de la señal de reloj, así como de los periféricos del microcontrolador.
- Entradas: ninguna.
- Salidas: ninguna.
- Errores: no se espera ningún error.

Apagado

- Descripción: S1 puede enviar la señal de apagado del sistema a S2. Cuando esto suceda, el microcontrolador debe de apagarse por completo y debe interrumpir la recepción de instrucciones, así como la generación de señales de control hacia los motores.
- Entradas: instrucción de apagado.
- Salidas: apagado del sistema.
- Errores: no se espera ningún error.

3.3.2. Requisitos *hardware*

Recepción y envío de secuencias de bits entre S1 y S2

- Descripción: debe existir un medio de comunicación basado en UART entre S1 y S2 que permita el intercambio de secuencias de bits.
- Entradas: secuencia de bits o instrucción a enviar.
- Salidas: recepción correcta por parte del destinatario.
- Errores: no se espera ningún error.

Generación de señales PWM

- Descripción: el microcontrolador situado en S2 debe ser capaz de generar señales eléctricas analógicas PWM, las cuales serán usadas para controlar los motores.
- Entradas: instrucciones del sistema de control en S1
- Salidas: señal de control PWM que se corresponde con la respuesta a dicha instrucción.
- Errores: no se espera ningún error.

Generación de señales digitales

- Descripción: el microcontrolador situado en S2 debe ser capaz de generar señales digitales.
- Entradas: instrucciones del sistema de control en S1.
- Salidas: señal de control digital.
- Errores: no se espera ningún error.

Recepción y procesamiento de señales analógicas

- Descripción: El microcontrolador debe ser capaz de recibir mediante sus pines y procesar las señales analógicas provenientes de los motores, en caso de que estos informen sobre su posición angular. Estas señales deben ser recibidas y procesadas mediante el ADC para poder ser tratadas a nivel de software.
- Entradas: señal analógica
- Salidas: señal procesada y convertida a datos tratables por el SW.
- Errores: no se espera ningún error.

Recepción y procesamiento de señales digitales

- Descripción: El microcontrolador debe ser capaz de recibir mediante sus pines y procesar las señales digitales. Estas señales deben ser recibidas y tratadas nivel de *software*.
- Entradas: señal digital
- Salidas: ninguna.
- Errores: no se espera ningún error.

Modo *Deep–Sleep*

- Descripción:el microcontrolador debe ser capaz de entrar en modo *Deep–Sleep*
- Entradas: ninguna.
- Salidas: ninguna.
- Errores: no se espera ningún error.

Encendido

- Descripción: se requiere un periodo de inicialización cuando se produce el encendido del sistema. Durante este periodo se realiza la inicialización de la señal de reloj, así como de los periféricos del microcontrolador.
- Entradas: ninguna.
- Salidas: ninguna.
- Errores: no se espera ningún error.

Apagado

- Descripción: S1 puede enviar la señal de apagado del sistema a S2. Cuando esto suceda, el microcontrolador debe de apagarse por completo y debe interrumpir la recepción de instrucciones, así como la generación de señales de control hacia los motores.
- Entradas: instrucción de apagado.
- Salidas: apagado del sistema.
- Errores: no se espera ningún error.

3.3.3. Requisitos de rendimiento

Pese a que no se ha restringido en particular, interesa que el sistema propuesto tanto en S1 como en S2 utilice los menos recursos posibles. Por una parte, en S1 la cantidad mínima de RAM que se recomienda es 512 MB, junto con un procesador que permita la ejecución de aplicaciones de forma concurrente.

Además, los cálculos matemáticos, que en principio se harán sobre ese sistema, han de poder ejecutarse, a ser posible, de forma asíncrona y estar optimizados para permitir un cómputo mínimo de un millón de operaciones cada segundo.

S2, por su parte, presenta más limitaciones en lo que a memoria y capacidad de cómputo se refiere. En particular para este proyecto, interesa que S2 bloquee el menor tiempo posible a S1, por lo que se intentará optimizar en tiempo de ejecución intentando además usar la menor cantidad de memoria posible. De esta forma:

1. El uso de la memoria RAM se buscará que sea el menor posible, sin sacrificar en rendimiento.
2. El uso de la memoria *flash* no se buscará reducirlo necesariamente, ya que eso puede afectar directamente al rendimiento.
3. Se trabajará en que el tiempo que el microcontrolador esté haciendo ejecuciones sea el menor posible, permitiendo así un ahorro de energía junto con estar menos tiempo bloqueando al S1.

3.3.4. Restricciones del diseño

En esta sección se describen algunas limitaciones existentes debido a distintos motivos, principalmente al HW y estructura física del *p*Arm.

En primer lugar, existe una limitación en cuanto a los materiales de fabricación de la estructura física del brazo, ya que se quiere construir íntegramente mediante un material plástico denominado *PLA*. Este material se utiliza para impresión en 3D y, dado que el *p*Arm se quiere imprimir por piezas utilizando una impresora de este tipo, el *PLA* es un material adecuado.

Por otro lado, para simplificar los cálculos en el modelo dinámico, se ha optado por usar un manipulador robótico pantográfico. Este tipo de manipuladores tienen una estructura similar a un flexo y la principal ventaja es que los motores se encuentran muy cercanos a la base. De esta forma, el peso de los mismos no debe ser desplazado al realizar movimientos en las articulaciones del brazo. En el caso de este tipo de brazos, los motores no se encuentran incluidos en los ejes de giro, si no que estos se encuentran en la base del mismo y transmiten su movimiento gracias a la estructura pantográfica formada por una serie de juntas.

3.3.5. Atributos del sistema *software* y *hardware*

Tanto para el SW como para el HW, se busca que ambos cumplan las siguientes premisas:

1. El sistema al completo ha de ser fiable. Esto es, no se permitirá al S2 realizar movimientos que puedan perjudicar la estructura del mismo de forma irremediable. A su vez, el sistema S2 deberá tener en cuenta posibles fallos en las órdenes de S1 y comprobar así que la secuencia de órdenes es segura.

2. Relacionado con el punto anterior, también se busca que el sistema sea seguro. En particular, se coordina junto con la fiabilidad para evitar que se puedan hacer movimientos que dañen al robot y, además, se harán diversas comprobaciones relativas al S1, en las cuales se habrá de verificar que está conectado, que es el sistema que dice ser y que se reciben instrucciones coherentes con la programación del sistema.

3. Teniendo en cuenta lo desarrollado en el punto de "Descripción general" (2) y lo mencionado en la "Introducción" (1), es importante que el sistema sea mantenible. Esto se traduce en que, por una parte, se pueda actualizar para corregir problemas que se han encontrado una vez se ha desplegado el sistema; y que la sustitución de piezas o elementos del mismo resulte accesible y barato.

4. Finalmente, dado que el *p*Arm está impreso en 3D, se busca que sea portable en lo referente a que pueda ser fácilmente transportado de un lugar a otro. Esto se traducirá en un bajo peso y que el área ocupada por el mismo sea también baja.

3.4. Requisitos no funcionales

Por motivos de tiempo, se dejan los requisitos no funcionales para una futura especificación.

Anexo A

Enlaces útiles

- GitHub de UFACTORY: https://github.com/uArm-Developer.
- Web de UFACTORY: https://www.ufactory.cc/#/en/support/download/pro.
- Estudio del manipulador μArm: https://github.com/UPM-Robotics/uarm.
- Funcionamiento del μArm: https://www.youtube.com/watch?v=VeZOi11NQRA.