COMILLAS
UNIVERSIDAD PONTIFICIA
ICAI

GRADO EN INGENIERÍA EN TECNOLOGÍAS INDUSTRIALES

TRABAJO FIN DE GRADO

DISEÑO DE UN MARCO DE REFERENCIA PARA EL CONSUMO DE ENERGÍA DE UN BRAZO ROBÓTICO INDUSTRIAL

Autor: Gonzalo Jiménez Berazaluce

Director: Álvaro Jesús López López

Madrid

Junio de 2020

Declaro, bajo mi responsabilidad, que el Proyecto presentado con el título

Diseño de un marco de referencia para el consumo de energía de un robot industrial en la ETS de Ingeniería - ICAI de la Universidad Pontificia Comillas en el curso académico 2019/20 es de mi autoría, original e inédito y no ha sido presentado con anterioridad a otros efectos.

El Proyecto no es plagio de otro, ni total ni parcialmente y la información que ha sido tomada de otros documentos está debidamente referenciada.

Fdo.: Gonzalo Jiménez Berazaluce Fecha: …29…/ …06…/ …2020…

Autorizada la entrega del proyecto

EL DIRECTOR DEL PROYECTO

Fdo.: Álvaro Jesús López López Fecha: 29 / 06 / 2020

COMILLAS
UNIVERSIDAD PONTIFICIA
ICAI

GRADO EN INGENIERÍA EN TECNOLOGÍAS INDUSTRIALES

TRABAJO FIN DE GRADO

DISEÑO DE UN MARCO DE REFERENCIA PARA EL CONSUMO DE ENERGÍA DE UN BRAZO ROBÓTICO INDUSTRIAL

Autor: Gonzalo Jiménez Berazaluce

Director: Álvaro Jesús López López

Madrid

Junio de 2020

Agradecimientos

Agradecerle a mi familia todo el apoyo mostrado durante estos años y su paciencia infinita para aguantarme en los momentos más complicados. Agradecerles también la presencia a los miembros del CIC LAB, en especial, a Javi, a Marcos y a Luis. Y, sobre todo, agradecerle a Álvaro esta gran oportunidad que me ha brindado para poder formarme y seguir creciendo como persona en un ambiente tan acogedor, le estaré agradecido el resto de mi vida.

DISEÑO DE UN MARCO DE REFERENCIA PARA EL CONSUMO DE ENERGÍA DE UN BRAZO ROBÓTICO INDUSTRIAL

Autor: Jiménez Berazaluce, Gonzalo.
Director: López López, Álvaro Jesús.
Entidad Colaboradora: Instituto de Investigación Tecnológica en colaboración con ICAI-Universidad Pontificia Comillas

RESUMEN DEL PROYECTO

En este proyecto se ha realizado un diseño de un marco de referencia para comprobar la eficacia de las técnicas de aprendizaje por refuerzo en el consumo energético de un robot industrial y se ha implementado la comunicación entre el controlador del robot con un código de aprendizaje por refuerzo, el encargado de optimizar el consumo de energía.

Palabras clave: Aprendizaje por refuerzo, consumo energético, robot industrial, brazo robótico, optimización, RobotStudio

1. Introducción

Desde 1939 que apareció Elektro, el primer robot de la historia, la industria de la robótica no ha parado de crecer, y ahora en la era de la digitalización y de la Industria 4.0, los robots se están volviendo un elemento indispensable en la industria. Hoy en día, la automatización industrial es ya una realidad y es indudable que los robots mejoran la eficiencia en los trabajos, optimizan los recursos y facilitan la vida de las personas. Sin embargo, a pesar de todas las ventajas que trae la automatización industrial y de que parezca impensable que no se utilicen robots para mejorar la productividad, solo un 40% de la industria española emplea los mismos. La tendencia de seguir automatizando procesos va a seguir creciendo en los próximos años a un ritmo anual del 10%. Buena muestra de ello son las grandes inversiones que están realizando empresas del sector logístico que están implementado estos robots en todas sus fases de la cadena logística. [1]

En la era de la digitalización y de la Industria 4.0, el empleo de herramientas y técnicas que utilicen el aprendizaje por refuerzo y la inteligencia artificial como método para resolver problemas es cada vez más frecuente. Año tras año, el número de datos crece de manera exponencial y es indispensable tener los recursos necesarios para extraer los datos más importantes y no quedarse atrás en un mundo que está en continuo cambio.

2. Definición del Proyecto

En este proyecto se busca mejorar la eficiencia de los robots en la industria mediante el ahorro de energía. Para ello, se han desarrollado dos elementos fundamentales. Por un lado, la comunicación necesaria entre la aplicación que maneja el robot industrial, RobotStudio, y un código en Python que es el que se encarga de llevar a cabo el aprendizaje por refuerzo. Y, por otro lado, se ha diseñado un marco de referencia, un elemento básico para comprobar el funcionamiento de la inteligencia del algoritmo, que consiste en una serie de campañas de medidas que tiene como objetivo determinar el consumo energético y el tiempo de operación del brazo robótico y que esté tiempo no se vea comprometido por un ahorro energético.

3. Descripción del modelo

Como principal herramienta de trabajo se ha utilizado un brazo robótico IRB120 de 6 grados de libertad de la empresa ABB situado en el laboratorio de Automatización y Sistemas Digitales de ICAI junto con un programa que lleva incorporado que es RobotStudio. Este programa ofrece una gran variedad de aplicaciones y herramientas que entre otras permiten al robot moverse y obtener los datos de ese movimiento. Este programa ofrece la función de programar en RAPID, donde se va a crear un código que le mande al robot las instrucciones necesarias para moverse y con qué tipo de movimiento (suave, lineal o circular). También se va a crear otro código en RAPID que establezca la comunicación con Pyhton, donde el algoritmo tomará las mejores decisiones para reducir el consumo energético. Una vez creado el código, dentro de RobotStudio se ha utilizado una herramienta llamada *signal analyzer online*, que permite obtener en tiempo real los datos de potencia y energía del brazo robótico que se exportarán a un Excel y se usarán para un posterior análisis y obtención de resultados. (Ver Ilustración 1)

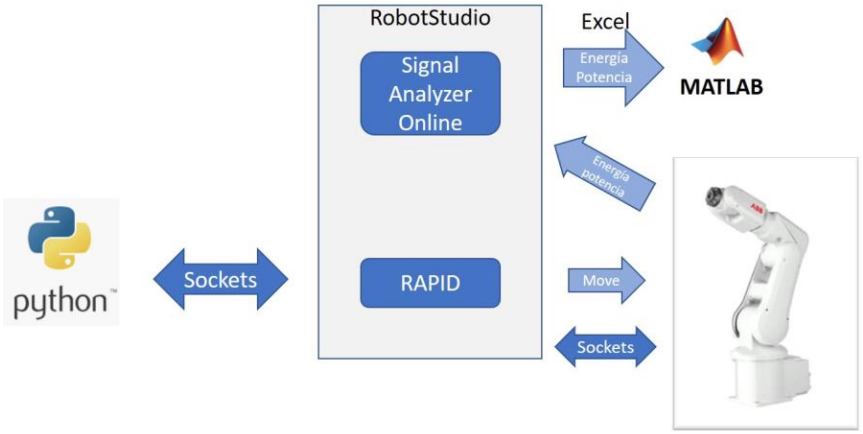


Ilustración 1: Esquema de funcionamiento

Para el análisis y la obtención de los resultados se ha desarrollado un código en MATLAB que se encarga de filtrar todo el conjunto de datos de potencia para obtener el consumo y el tiempo de cada ciclo de operación. A continuación, se muestra cómo es la potencia aplicada en un ciclo completo. (Ver Ilustración 2)

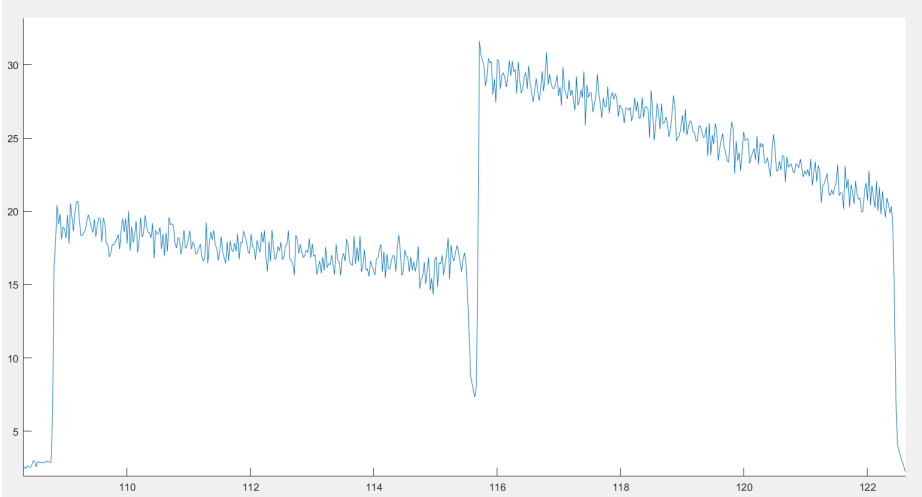


Ilustración 2: Potencia en W aplicada por el brazo robótico en un ciclo de trabajo

Esta figura se prologaría en el tiempo los ciclos que hiciera falta. El primer tramo corresponde al robot bajando y el segundo subiendo ya que va en contra de la gravedad y por eso consume más.

4. Resultados

Del código en MATLAB una vez filtrado se obtiene el consumo energético de cada ciclo de operación y el tiempo del mismo. A continuación, se muestran los resultados. (Ver Ilustración 3)

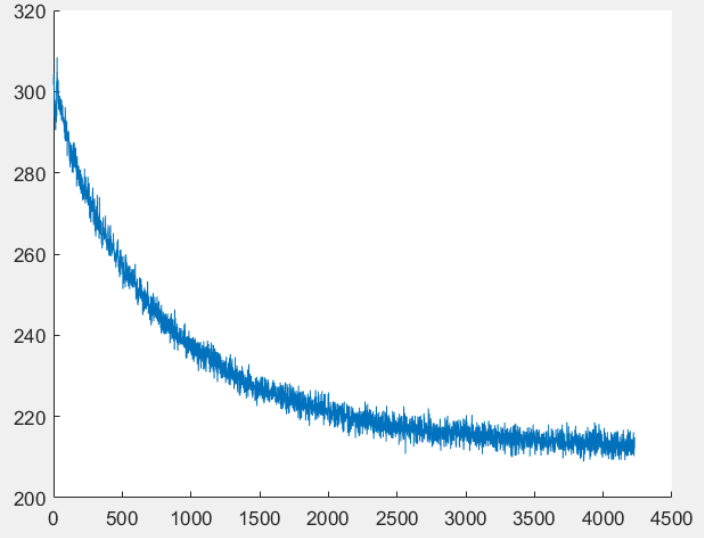


Ilustración 3: 4250 ciclos de energía

Se puede observar cómo sigue una etapa de régimen transitorio hasta alcanzar un régimen permanente. El tiempo no se incluye porque ha sido contante en 1,4 segundos y hasta alcanzar el régimen permanente ha sido 1 hora y media aproximadamente con el brazo robótico en modo automático y a máxima potencia.

Con los últimos datos de la Ilustración 3 ya en régimen permanente se procede a estimar la media quedando de 212,73 J con un intervalo al 95% de confianza de [212,07 - 213,39].

5. Conclusiones

Para comparar los resultados que ofrece el aprendizaje por refuerzo y que el algoritmo llegue a aprender cuáles son las rutas óptimas es necesario trabajar en régimen permanente. El régimen transitorio que aparece se debe a que los motores del brazo robótico tienen que calentarse. Al calentarse el aceite y la grasa que el brazo robótico tiene en sus ejes, éstos reducen su viscosidad. Entonces se reducen las pérdidas por fricción y, por tanto, el consumo total de energía disminuye.

6. Referencias

[1] "Ex Summary World Robotics 2019 Industrial Robots", International Federation of Robotics, septiembre 2019. Último acceso 26/06/2020 https://ifr.org/downloads/press2018/Executive%20Summary%20WR%202019%20Industrial%20Robots.pdf

BENCHMARK DESIGN FOR THE ENERGY CONSUMPTION OF AN INDUSTRIAL ROBOTIC ARM

Author: Jiménez Berazaluce, Gonzalo.
Supervisor: López López, Álvaro Jesús.
Collaborating Entity: Institute for Research in Technology in collaboration with ICAI

ABSTRACT

In this project, a benchmark design has been made to check the efficiency of the Reinforcement Learning techniques in the energy consumption of an industrial robot and the communication has been implemented between the robot's controller and a reinforcement learning code, the one in charge of optimizing the energy consumption.

Keywords: Reinforcement Learning, energy consumption, industrial robot, robotic arm, optimization, RobotStudio

1. Introduction

Since 1939 when Elecktro, the first ever robot, appeared, the robotics industry has not stopped growing, and now in the digitalization era and Industry 4.0, robots are becoming and indispensable element in the industry. Nowadays, industrial automation is already a reality and there is no doubt that robots improve work efficiency, optimize resources and make people's live easier. However, despite all the advantages that industrial automation brings and the fact it seems unthinkable that robots should not be used to improve productivity, only 40% of Spanish industry employs them. The trend of continuing to automate processes will continue to grow in the coming years at an annual rate of 10%. A good example of this are the large investment being made by companies in the logistics sector that are implementing these robots in all phases of the supply chain. [1]

In the digitalization era and Industry 4.0, the use of tools and techniques that use Reinforcement Learning and Artificial Intelligence as a method for solving problems is becoming more and more frequent. Every year, the number of data items is growing exponentially and it is essential to have the necessary resources to extract the most important data and not be left behind in a world that is constantly changing.

2. Project definition

This project seeks to improve the efficiency of robots in industry by saving energy. To this end, two fundamental elements have been developed. On the one hand, the necessary communication between the application that handles the industrial robot, RobotStudio, and a code in Python that is in charge of carrying out the Reinforcement Learning. And, on the other hand, a benchmark has been designed, a basic element to check the functioning of the algorithm's intelligence, which consists of a set of measurement campaigns that aim to determine the energy consumption and operating time of the robotic arm and that the time is not compromised by energy saving.

3. Description of the model

As the main working tool, a robotic arm IRB120 with 6 degrees of freedom form the company ABB located in the Automation and Digital Systems laboratory at ICAI has been used, together with a program that it has incorporated, called RobotStudio. This program offers a wide variety of applications and tools that among others allow the robot to move and obtain the data of that movement. This program offers the function of RAPID programming where a code is created and it sends to the robot the instructions to move and with a type of movement (joint, linear, circular). Another RAPID code will also be created that establishes communication with Python, where the algorithm will make the best decisions to reduce the energy consumption. After the code is created, a tool within RobotStudio, called signal analyzer online, is used to obtain real-time power and energy data from the robotic arm that will be exported to an Excel and used for further analysis and results. (See Figure 1)

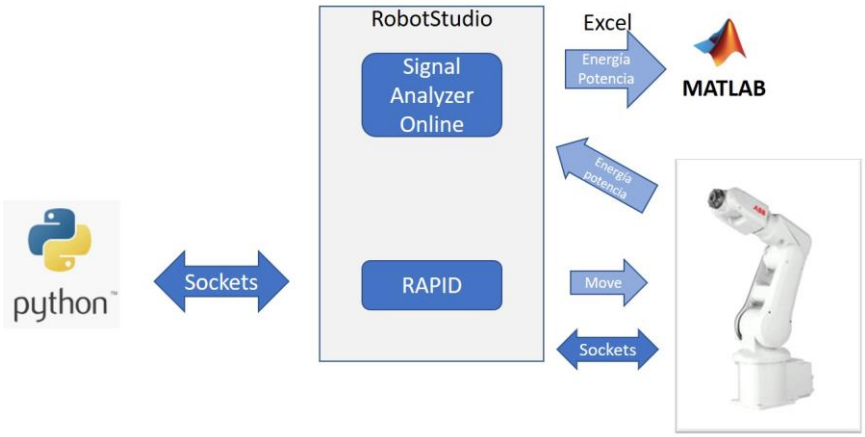


Figure 1:Operating diagram

For the analysis and obtaining the results, a code has been developed in MATLAB that filters the entire power data set to obtain the consumption and time of each operation cycle. The following shows what the power applied in a complete cycle looks like. (See Figure 2)

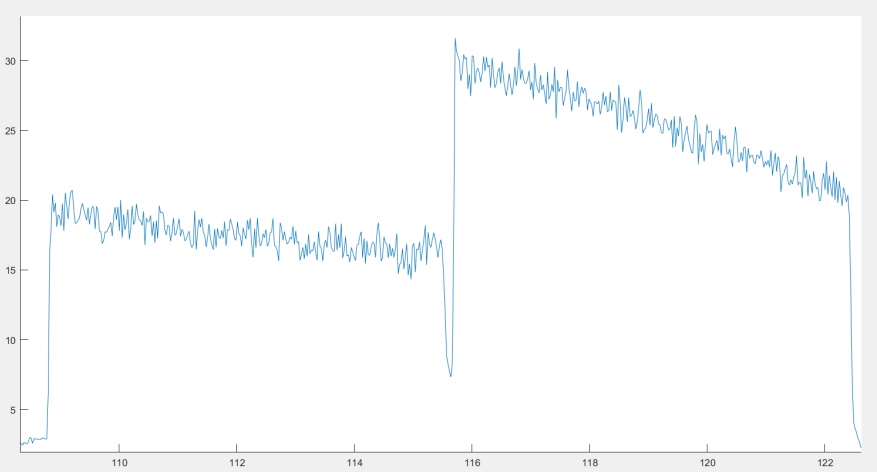


Figure 2: Power in W applied by the robotic arm in one work cycle

This figure would be prolonged in time as many cycles as necessary. The first section corresponds to the robot going down and the second one going up since it goes against gravity and therefore consumes more.

4. Results

From the MATLAB code once filtered, the energy consumption of each operation cycle and the time of it is obtained. (See Figure 3)

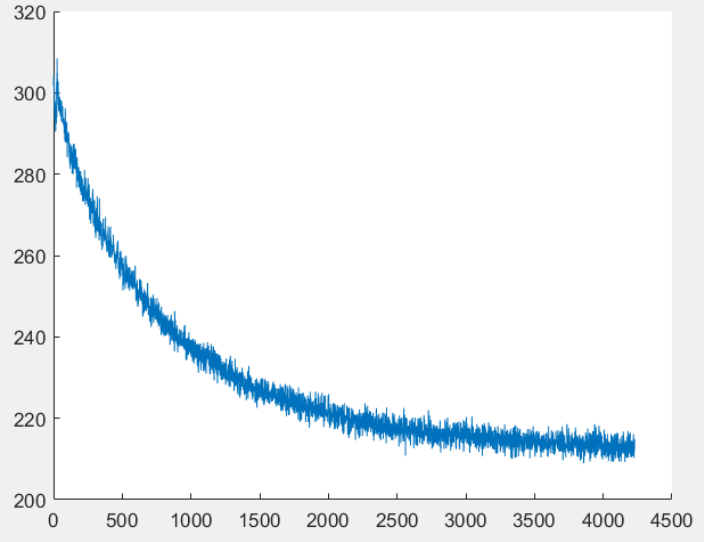


Figure 3: 4250 energy cycles

It can be seen how a stage of transitional regime is followed until a steady state is reached. The time is not included because it has been constant in 1.4 seconds and until reaching the steady state it has been approximately 1 hour and a half with the robotic arm in automatic mode and at maximum power.

With the last data of already in steady state, it is proceeded to estimate the average, remaining 212.73 J with an interval at 95% of confidence level of [212.07 – 213.39].

5. Conclusions

In order to compare the results offered by Reinforcement Learning and for the algorithm to learn which are the optimal routes, it is necessary to work on a permanent basis. The transitional state that appears is due to the fact that the motors of the robotic arm have to heat up. Once the oil and the grease on the robotic arm´s axles heat up, their viscosity is reduced. Friction are then reduced, and the total energy consumption is therefore reduced.

6. References

[1] "Executive Summary World Robotics 2019 Industrial Robots", International Federation of Robotics, septiembre 2019. Último acceso 26/06/2020 https://ifr.org/downloads/press2018/Executive%20Summary%20WR%202019%20Industrial%20Robots.pdf

Índice de la memoria

Índice de figuras

Capítulo 1. INTRODUCCIÓN

Desde 1939 que apareció Elektro, el primer robot de la historia, la industria de la robótica no ha parado de crecer, y ahora en la era de la digitalización y de la Industria 4.0, los robots se están volviendo un elemento indispensable en las industrias. En la actualidad, la mayoría de los robots se encuentran realizando trabajos repetitivos para los humanos o en trabajos peligrosos que puedan poner en riesgo la vida de las personas. Hoy en día, la automatización industrial es ya una realidad y es indudable que los robots mejoran la eficiencia en los trabajos, optimizan los recursos y facilitan la vida de las personas. Parece impensable que siga habiendo una fábrica que no haga uso de robots que automatizan procesos. Sin embargo, solo un 40% de la industria española hace uso de estos en su cadena logística. Al menos hay noticias esperanzadoras en este sentido y es que se espera que la industria de la robótica crezca a un ritmo anual del 10%. Buena muestra de ello son las grandes inversiones que están realizando empresas como Amazon que están implementado estos robots en todas sus fases de la cadena logística. [1]

La industria robótica: un mercado en continuo crecimiento

El mercado de los robots ha estado en constante expansión desde 2012, pero la tendencia de la robotización, que se ha acelerado desde 2017, se ha incrementado aún más en la primera mitad de 2018. Sin embargo, la guerra comercial entre EE.UU. y China ha puesto un poco de freno al desarrollo de esta industria, reduciendo sus inversiones en esta industria. [2]

Aun así, la necesidad de tener robots está aumentando debido a la escasez de mano de obra y al descenso de la población activa, que cada vez demanda más calidad en su trabajo como no tener que realizar tareas repetitivas o trabajos pesados. Además, gracias a la llegada de los robots colaborativos, dejando a un lado a los robots convencionales que requerían de un gran espacio para su instalación por razones de seguridad, los trabajadores pueden trabajar con estos robots codo con codo y sin vallas de seguridad. Así que, las oportunidades para la robotización y la automatización industrial están aumentando cada vez más.

En 2014 se esperaba que la cifra de robots para el 2020 estuviese ya cerca de los 2 millones y medio. Sin embargo, debido al crecimiento en los últimos años, la cifra ha alcanzado un crecimiento récord de unidades vendidas de 422.271 en 2018, lo que supone un total de 2.439.543 unidades en total. A la espera de un mayor crecimiento en 2019 que podría suponer estar cerca de las 450.000 unidades y en 2020 que se espera la cifra supere las 500.000 unidades, la cifra de 2 millones y media se habría sobrepasado con creces y estaría hablando de una cifra cercana a los 3,4 millones de robots en el mundo. Además, se espera que la industria de la robótica a nivel mundial siga creciendo en los próximos años a un ritmo del 12% anual con una mayor acentuación en Asia y Estados Unidos. [1]

La industria de la robótica en España: más prometedora y productiva

Según los últimos informes del SEPI, el porcentaje de empresas que empleaban robot en la industria era del 33,3% en el período entre 2014-16, actualmente la cifra ronda el 40%. Sin embargo, las empresas con robots, a pesar de ser solo un tercio del total, cuadriplican en ventas a las empresas no robotizadas (132.2 millones de euros frente a 30.1 millones de euros), así como en el volumen de activos totales (117.6 frente a 28.7 millones de euros). La mayoría de estos robots (aproximadamente un 70%) se encuentran principalmente en cuatro ramas: automoción, electricidad y electrónica, metalurgia y química.

En términos de productividad, las empresas robotizadas tienen una productividad media de 70.3 miles de euros frente a 54.7 miles de euros. En consecuencia, también son superiores los gastos por trabajador, pero no tanto (39 mil euros frente a 34), y el gasto externo en formación del trabajador son 134.4 euros frente a 70.4. Por último, las empresas con robots tienen de media más empleados (337 frente a 97), luego se podría decir que aproximadamente el 63% de los trabajadores de la industria trabaja en empresas robotizadas y este porcentaje es mayor que el 40% del que se ha hablado inicialmente. Aun así, sigue haciendo falta seguir invirtiendo en temas de automatización industrial ya que, en definitiva, los robots facilitan las tareas en la industria, ofrecen una mayor seguridad y, además, abaratan los costes de producción. [3] [4]

Inteligencia artificial y Machine Learning

En la actualidad un campo que está haciendo crecer la industria de la robótica y que está mejorando su eficacia y productividad es el campo de la inteligencia artificial, especialmente, el del Machine Learning o aprendizaje automático.

Algunos de los logros que la inteligencia artificial ha logrado en la última década ha sido vencer a los campeones de ajedrez, go y starcraft. Juegos que requieren de mucha habilidad y de años de experiencia. Pero la inteligencia artificial también está ayudando en otros campos como la medicina o la farmacia, donde ya ha detectado algún cáncer en mucho menos tiempo que el médico y con una fiabilidad del 99%. Si la inteligencia artificial ya ha sido capaz de abrir estas puertas que parecía impensables que se abriesen, no quiero ni pensar dónde queda el límite de lo que puede llegar a hacer en un futuro, a lo mejor no tan lejano.

Un área del *Machine Learning* es el aprendizaje por refuerzo, del cual se hablará más a fondo en el Capítulo 2. En esta introducción solo se va a mostrar la tendencia que tiene y como cada vez el uso del aprendizaje por refuerzo está más generalizado. A continuación, se adjuntan en las Figura 1 y Figura 2 las tendencias de los términos “Reinforcement Learning” y “Deep Reinforcement Learning” en todo el mundo según Google Trends, una página que recoge las búsqueda y publicaciones de una determinada palabra en Internet.

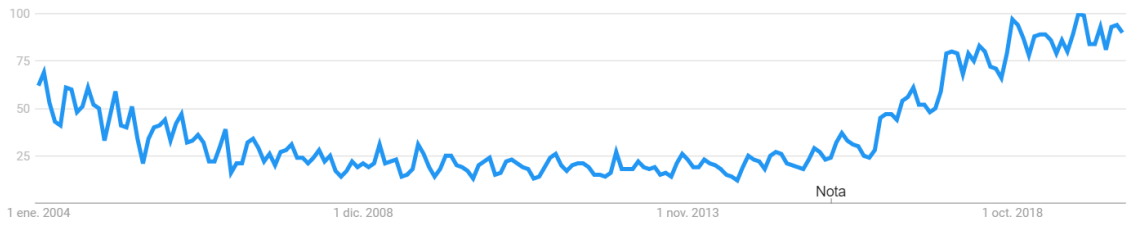


Figura 1: Búsquedas de “Reinforcemente Learning” a lo largo de los años. Datos vía Google Trends

Figura 2: Búsquedas de "Deep Reinforcemente Learning" a lo largo de los años. Datos vía Google Trends

En vista de las crecientes tendencias y a que durante los próximos años lo va a seguir haciendo, este Trabajo de Fin de Grado se ha orientado a este campo del Machine Learning, el aprendizaje por refuerzo, abordando otro campo que está en un continuo auge como es la robotización de la industria.

Capítulo 2. APRENDIZAJE POR REFUERZO

En este capítulo se a hacer una descripción más profunda de lo qué es el aprendizaje por refuerzo. Además, se van a analizar las diferentes partes de un sistema que funciona por aprendizaje por refuerzo.

2.1 APRENDIZAJE POR REFUERZO

El Aprendizaje por Refuerzo es un área del aprendizaje automático y ha sido objeto de estudio desde hace aproximadamente 60 años, pero su forma más actual se debe a la gran influencia de la teoría de decisión de Markov que apareció en 1965 y se estableció en los 90 como un sistema de decisión clave para lograr alcanzar el objetivo deseado. El problema de decisión de Markov se basa en elegir la mejor acción en un marco de estados, acciones y tiempo discreto. Da una estructura matemática para modelar sistemas de decisión en los que algunas acciones se toman de manera aleatoria (exploración) y otras bajo un control de decisión (explotación). [5]

En los últimos 20 años el campo del aprendizaje por refuerzo ha crecido a pasos agigantados. Los nuevos conocimientos de este período reciente han sido una mayor aplicabilidad, la conexión con la inteligencia artificial y las teorías sobre *brain theory*. En 1999 se distinguían 3 áreas de desarrollo del aprendizaje por refuerzo: pasado, presente y futuro.

El pasado del aprendizaje por refuerzo se ha basado en la idea de aprendizaje por prueba y error. En este período se desarrolló la idea de que de que era el agente el explorador activo y el concepto clave de una recompensa escalar para especificar el objetivo del agente. Los métodos empleados por lo general sólo aprendían políticas y no eran capaces de tratar con eficiencia recompensas retrasadas.

El presente del aprendizaje por refuerzo es el período en el que se formalizaron las funciones de valor (*value functions*). Las funciones de valor son el centro de gravedad del aprendizaje

por refuerzo y prácticamente todos los métodos se centran en aproximaciones de las funciones de valor para calcular las políticas óptimas. La hipótesis de las funciones de valor dice que la aproximación de las funciones de valor es el objetivo principal de la inteligencia.

El futuro del aprendizaje por refuerzo dará un paso más grande y se centrará en estructuras que habiliten la aproximación de las funciones de valor. Muchos desarrollos serán sobre las propiedades, capacidades y garantías de la convergencia y el comportamiento de estas nuevas estructuras. Bayesian frameworks, aproximaciones lineares eficientes, relational knowledge representation y descomposición de jerarquías y el uso de multiagentes son algunas de nuevas estructuras que se están empleando y que se emplearán con mayor frecuencia en años futuros. [5]

El Aprendizaje por Refuerzo es un área del Machine Learning que tiene también conexiones con otros campos como la psicología, la investigación operativa y la optimización matemática. El Aprendizaje por Refuerzo se basa en que un agente realiza acciones en función de lo que sucede en un entorno con el objetivo de alcanzar un determinado objetivo. Para ello, maximizará una función recompensa que indica al agente cómo de buena o mala ha sido la acción que ha tomado y cuando está función es acumulativa, es decir, las recompensas de cada acción que toma se almacenan en una variable, ésta simula el objetivo al que se ha de llegar el agente. Mediante repeticiones e iteraciones el agente aprende una política óptima, lo cual da una mayor robustez y flexibilidad al agente, que puede aprender varias políticas óptimas para alcanzar su objetivo y modificar y aprender una nueva en caso de que aparezcan irregularidades en el entorno. En la Figura 3 se puede ver un esquema del aprendizaje por refuerzo. [6]

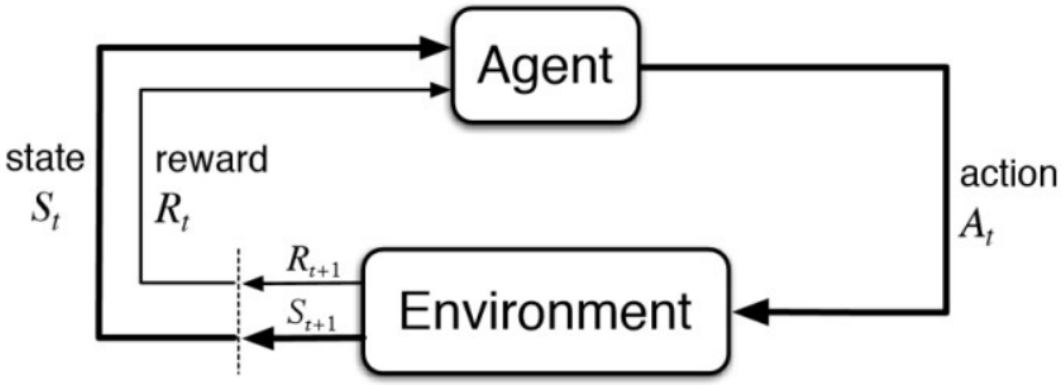


Figura 3: Esquema del Aprendizaje por Refuerzo

Agente, entorno, estado, acción y recompensa se describirán mejor en la sección 2.2 con un epígrafe cada uno y enfocados al problema a resolver. El aprendizaje por refuerzo se basa en el proceso de decisión de Markov y en la ecuación de Bellman que forman parte de la programación dinámica. El agente percibe un conjunto finito de estados en su entorno, S, y puede realizar un conjunto de acciones finitas, A. El tiempo es discreto, y en cada instante de tiempo el agente percibe un estado S_t y toma una acción llegando a un estado S_{t+1}. El entorno devuelve una recompensa R_t que será un valor numérico asignado por nosotros. Tanto el estado al que se ha llegado como la recompensa obtenida no tienen por qué ser conocidos por el agente. Antes de aprender el agente no sabe qué pasará cuando tome una determinada acción estando en un estado concreto. Sin embargo, una vez haya aprendido, sabrá reconocer cuáles son las mejores acciones y la mejor política. Ese es el objetivo del aprendizaje por refuerzo, que se explica con ecuaciones a continuación. [7]

$$V_\pi(s) = E_\pi[R_{t+1} + \gamma V_\pi(S_{t+1})|S_t = s]$$

Ecuación de Bellman para el valor de un estado

Donde γ es factor de descuento.

Esta ecuación nos da el valor de un estado en función de la recompensa que obtendría del estado siguiente y del valor del estado siguiente. γ está entre 0 y 1 y cuanto mayor sea más valor se le da a las recompensas a largo plazo.

Como existen varios valores de la función y varias políticas distintas, cuando se está resolviendo un problema que involucra el proceso de decisión de Markov se está buscando optimizar el valor de la función, es decir, el de la recompensa. La siguiente ecuación es la que emplean la mayoría de los algoritmos de aprendizaje, si no todos.

$$q_*(s,a) = \underbrace{max}_{\pi} [R_{t+1} + \gamma q_\pi(S_{t+1}, A_{t+1}) | S_t = s, A_t = s]$$

Ecuación del valor de la función óptimo de estado-acción

La Ecuación de arriba se trata del máximo valor de la función de todas las políticas, similar a la ecuación de Bellman que solo tiene en cuenta el valor de un estado para una política concreta.

En búsqueda de una mejor optimización, en algunos casos se emplea un factor de aprendizaje para evitar que haya cambios bruscos en las acciones y, por tanto, sirve para controlar el valor de Q. [7]

$$q'(s,a) = (1-\nu)q(S_t, A_t) + \nu[R_{t+1} + \gamma max_{A_{t+1}} q(S_{t+1}, A_{t+1}) | S_t = s, A_t = s]$$

Donde ν es el factor de aprendizaje. Cuanto menor sea este factor, menos valor se le da a la parte de buscar el óptimo del estado-acción, es decir, menos cambios bruscos va a dar la función.

El nuevo valor de Q es una combinación ponderada del antiguo valor de Q y la nueva información que le llega al agente al tomar la siguiente acción.

Un aspecto a tener en cuenta del aprendizaje por refuerzo es que solo se aprende de las acciones que se toman. De los estados por los que no se pasa y las acciones que no se toman no se aprende. Aquí aparecen 2 conceptos: exploración y explotación. Tiene que haber un compromiso entre ambos (Ver Figura 4). Bien es cierto que interesa que se conozcan todos los estados posibles y se tomen todas las acciones posibles, pero esto llevaría a un proceso de aprendizaje lento y un mayor consumo de recursos. Y bien por otro lado, se interesa que se llegue lo más rápido posible a la solución del problema, pero puede ser que esa solución

no sea la óptima ya que no ha explorado otras opciones y el agente o la inteligencia ha considera como óptima una política errónea. En resumen, la explotación selecciona el valor de Q para ese estado y la exploración selecciona una acción al azar. [6]

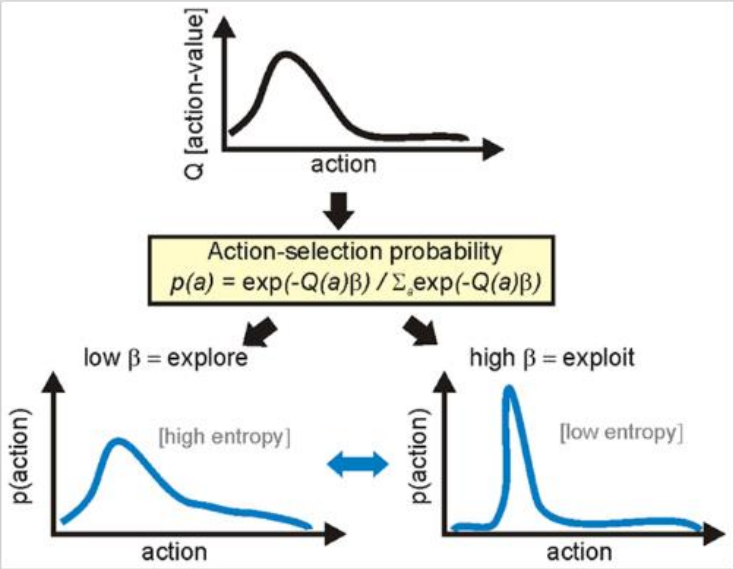


Figura 4: Exploración vs Explotación, probabilidad de tomar una acción

Donde β es la constante de Boltzmann que si es alta indica que se prioriza la explotación y si es baja indica que se está priorizando exploración.

La mayoría de los algoritmos comienzan con una exploración muy alta, para después tener una explotación muy alta. Este método funciona bien cuando la mejor acción está separada de las demás. Sin embargo, cuando los valores de la acción están muy cerca unos de otros tarda en converger.

2.2 APRENDIZAJE POR REFUERZO APLICADO A LA ROBÓTICA

Uno de los campos en los que es más empleado el aprendizaje por refuerzo es en de la robótica. Algunos de los principales retos que tiene el aprendizaje por refuerzo en este campo son: alcanzar (reaching), empujar (pushing), deslizar (sliding) y coger y depositar (pick & place).

Reaching es mover la pinza a una posición de destino. Esta tarea es sencilla de aprender y es una buena manera de comprobar si un algoritmo funciona y hace que el agente llegue a aprender. *Pushing* consiste en poner una caja en el entorno y el objetivo es moverla a otro punto de la superficie en la que esté apoyada. Los dedos del brazo robótico están bloqueados para evitar que el robot agarre la caja. *Sliding* consiste en colocar un disco sobre una superficie sobre la que pueda deslizar de tal forma que el brazo robótico golpee el disco y este llegue a un punto concreto de la superficie. *Pick & place* consiste en que el robot coge una caja y la deja en otro punto del entorno, el cual no tiene por qué estar situado sobre la superficie. [8]

Concretamente en este proyecto se va a trabajar en la optimización del consumo energético en tareas de "*pick and place*". En el resto de los epígrafes que vienen a continuación se van a explicar las diferentes partes que constan el aprendizaje por refuerzo aplicadas a un brazo robótico industrial con 6 grados de libertad. Además, se va a realizar una investigación teórica sobre cuál es el algoritmo de aprendizaje por refuerzo idóneo para lograr alcanzar el objetivo propuesto: minimizar el consumo de energía del robot industrial sin ver comprometido su tiempo de ciclo.

2.2.1 ENTORNO

Existen dos tipos de entorno: el virtual y el real. El entorno virtual es el desarrollado en Python que será lo más parecido posible a los límites de funcionamiento del robot (Ver Figura 7 y Figura 8). El entorno real es el entorno en el que se encuentra el brazo robótico, es decir, los límites de funcionamiento que tiene. Es físico y en este caso se encuentra en el laboratorio de automatización y sistemas digitales de ICAI. El agente se va a entrenar en el entorno virtual y después se va a comprobar su funcionamiento en el entorno real donde con datos reales se va a terminar entrenando.

2.2.2 ESTADO

El estado es una representación de lo que el agente está observando del entorno en un determinado momento concreto del tiempo. En este caso cada estado viene definido por una

posición en coordenadas cartesianas o por la posición de los 6 ángulos de los ejes del brazo robótico.

2.2.3 ACCIÓN

La acción está formada por las distintas posibilidades de movimiento que tiene el brazo robótico. El diseño en Python se realiza como si se pudiera mover en una esfera de radio r, generalmente pequeño, para tener controlado el movimiento. Hay que tener en cuenta que al ser distintos el entorno virtual de Python y el entorno real del brazo robótico, unas acciones le estarán permitidas en el entorno virtual que, sin embargo, en el entorno real harían que el robot se bloquease. Para nuestro robot la acción está formada por la variación en coordenadas cartesianas o por la variación de los grados de cada uno de los ángulos que permiten libertad de movimiento al brazo robótico.

2.2.4 RECOMPENSA

Este apartado es uno de los más complicados de todos ya que como se comentará más adelante no solo vale con disminuir el consumo energético, también hay que tener un compromiso con el tiempo, es decir, este no puede ser mayor que el que tardaría sin el uso del aprendizaje por refuerzo. Por tanto, hay que desarrollar una recompensa que combine el tiempo con el consumo energético. Una mala combinación puede provocar que aprenda a llegar a la posición final en poco tiempo, pero puede hacer que no disminuya el consumo energético. Y viceversa, puede ser que apenas consuma energía, pero que tarde mucho más de lo esperado en llegar al objetivo.

De aquí la necesidad de diseñar un marco de referencia que nos dé el tiempo que tarda en llegar a un objetivo y el consumo energético en ese tiempo. (Ver Capítulo 5.). Entonces teniendo el tiempo medio y el consumo energético de cada secuencia se va a penalizar negativamente aquellos ciclos que superen considerablemente estos parámetros. Y como el objetivo es maximizar la recompensa, el agente va a aprender un poco de ambas (a veces se le penalizará por el tiempo, otras por el consumo energético).

2.2.5 ALGORITMO DE APRENDIZAJE

En este apartado se va a hacer una revisión de los principales algoritmos de aprendizaje que existen en el aprendizaje por refuerzo y que se van a emplear en este proyecto. Cabe destacar que existen varios tipos de algoritmos: on-policy, off-policy y actor critic. El método on-policy tiene una política que va actualizando al terminar cada episodio. El método off-policy tiene una política inicial que la mantiene y actúa sobre ella. Existe otra división que se comenta en el Capítulo 3. que es en función de la continuidad, entre discretos y continuos. El algoritmo que se va a emplear es una combinación de Deep Deterministic Policy Gradient (DDPG) y Hindsight Experience Replay (HER). Se explican más en detalle en la sección 3.2.

Capítulo 3. ESTADO DE LA CUESTIÓN

3.1 TRABAJOS PREVIOS

El desafío de los robots en los sistemas de producción que engloban la Industria 4.0 es ser flexibles y adaptables al entorno a la vez que ser robustos y económicamente eficientes. En este apartado se van a tratar diferentes enfoques que se están aplicando a la industria para lograr alcanzar dichos objetivos [9]. Ya sea hacer al robot seguir una cuerda, que simularía una trayectoria o hacer llegar al robot de un sitio a otro esquivando objetos. En ambos problemas hay un elemento común, el Aprendizaje por Refuerzo.

De los algoritmos de *Reinforcement Learning* (RL) que existen se pueden distinguir dos tipos distintos: *discrete action space* (DAS) y *continuous action space* (CAS). Dentro de los que son continuaos se dividen en estocáticos (SCAS) y determinísticos (DCAS). El algoritmo más empleado en los robots es el *Deterministic Policy Gradient* (DPG), el cual se puede combinar con otros [9]. En esta línea es en la que se ha trabajado en la industria.

Uno de los últimos estudios (T. Johannink et al., mayo 2019) consiste en que un brazo robótico haga una trayectoria esquivando objetos. Este estudio combina las técnicas del RL con los métodos de control tradicionales diseñados a mano (ver Figura 5). La ventaja que tiene este algoritmo mixto es que recoge lo mejor de cada control y aborda los problemas del otro. El problema principal del RL es que requiere aprender mediante iteración, que puede ser inseguro inicialmente, y debido a la complejidad del problema puede consumir mucho tiempo hasta que aprende. Los algoritmos de control por realimentación de estado son caros, ya que un diseño robusto de las ecuaciones puede llegar a ser más caro que el propio hardware del robot debido a los sensores empleados y al tiempo empleado en modelar el sistema, con el RL estos costes se verían reducidos. Gracias a los controles por realimentación se obtiene un ahorro en el tiempo de exploración del comportamiento del RL, y esto permite al RL resolver las incertidumbres del ruido y que los objetos a esquivar se coloquen en distintas posiciones. [10]

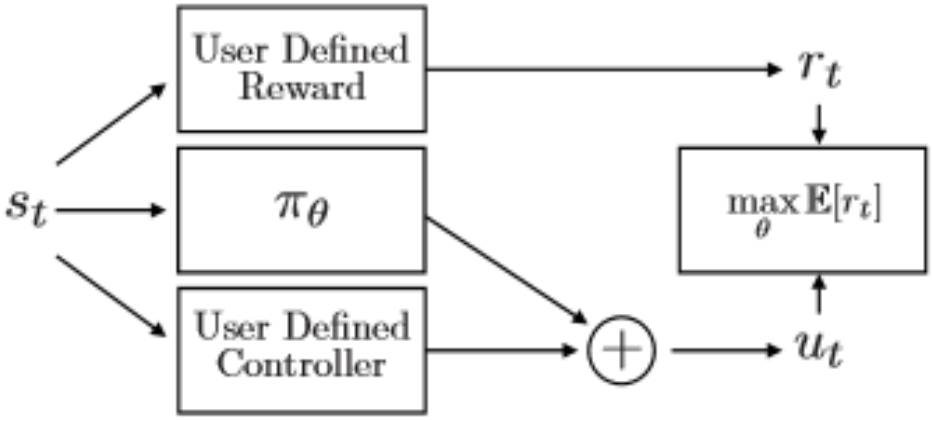


Figura 5: Diagrama de bloques de un algoritmo mixto que combina RL con un sistema de control convencional

El algoritmo que emplean es el *twin delayed deep deterministic policy gradients* (TD3), que es más eficiente y requiere menos tiempo de desarrollo que el *Deep Deterministic Policy Gradient* (DDPG). Los resultados de este estudio muestran como es el uso de un algoritmo mixto es considerablemente mejor que el empleo de ambos por separado.

Otro estudio emplea una cámara en el robot que le da al agente una visión del entorno, en el cual hay una cuerda que el agente tiene que seguir. El problema que tiene el agente en este entorno es que una vez que comete un fallo (toca la cuerda), o una parte de la cuerda se encuentra fuera del rango de alcance del robot, el agente reinicia el movimiento. Para solucionar este problema se emplean dos fases. Una primera parte de entrenamiento en la que el agente explora el entorno. Y una segunda fase que comprueba el entrenamiento de la primera fase, entonces se actualiza el estado inicial el estado inicial como el último estado al que ha llegado. El algoritmo que se emplea es el *e-gready* crea un ciclo iterativo de entrenamiento y comprobación. Este algoritmo está dentro del DAS. [11]

Con respecto al ahorro energético, un robot bípedo redujo su consumo energético un 18% gracias al uso del RL. El algoritmo de aprendizaje se basó en optimizar la variación de la altura del centro de masas. Trabajaron primero en un entorno virtual y luego en un entorno real con un robot COMAN donde obtuvieron resultados extraordinarios. [12]

3.2 ALGORITMOS ÓPTIMOS DE APRENDIZAJE POR REFUERZO

Además, se van a analizar los principales algoritmos de aprendizaje por refuerzo que existen y se va a seleccionar el que mejores resultados de aprendizaje ofrezca. Ya sea por ser el que menos tarde en aprender, o por ser el que más robustez ofrezca a los posibles cambios que pueda haber en el entorno. El objetivo es que el algoritmo termine convergiendo en la solución óptima y no consuma mucho tiempo de operación de la CPU.

A continuación, se explican los dos algoritmos que se van a emplear para desarrollar el código en Python.

- Deep Deterministic Policy Gradient (DDPG)

  El brazo robótico tiene 6 grados de libertad, entonces una acción discretizada puede tener los valores de {-a,0,a}. Lo cual serían en total 36=729 posibilidades por cada acción que se realice que llevaría a una alta carga computacional. Se trata de entorno difícil de explorar y en el que el agente no aprendería. Además, la discretización de los espacios de la acción elimina innecesariamente información sobre la estructura del dominio de acción, que puede ser esencial para resolver muchos problemas. De aquí la necesidad de trabajar en un entorno continuo.

  El algoritmo DDPG se basa en el Q-learning y en la ecuación de Bellman para aprender, pero no es posible únicamente aplicar Q-learning a acciones continuas ya que el aprendizaje sería lento y poco práctico.

$$\begin{aligned}\nabla_{\theta^\mu} J &\approx \mathbb{E}_{s_t\sim\rho^\beta}\left[\nabla_{\theta^\mu} Q(s,a|\theta^Q)|_{s=s_t,a=\mu(s_t|\theta^\mu)}\right]\\ &= \mathbb{E}_{s_t\sim\rho^\beta}\left[\nabla_a Q(s,a|\theta^Q)|_{s=s_t,a=\mu(s_t)}\nabla_{\theta_\mu}\mu(s|\theta^\mu)|_{s=s_t}\right]\end{aligned}$$

  Se basa en el descenso del gradiente y en buscar direcciones que den un mayor valor de la función. Este algoritmo almacena en un buffer de repetición los datos de (st,at,rt,st+1) en el que las muestras más antiguas se descartan. En cada paso el actor crítico o agente se actualiza cogiendo nuevas muestras del buffer. Al ser tiempo y

acción continuas el buffer tiene que ser grande para así aprender y beneficiarse de un conjunto de transiciones incorreladas. El actor crítico coge objetivos sencillos en vez de copiar directamente los pesos de la política. Esto permite que se aumente la estabilidad del aprendizaje.

Este algoritmo resuelve uno de los principales problemas de los aprendizajes en espacios continuos, que es la exploración. Para ello, introduce un parámetro de ruido que lo añade a la política de aprendizaje y que favorece la exploración.

En resumen, el algoritmo DDPG es un algoritmo sencillo que requiere solo un actor crítico y aprende rápidamente con pocos movimientos. Es un algoritmo escalable según (Lillicrap, T. et al) que permite trabajar con problemas sencillos y después escalarlo a problemas más extensos y con mayor componente computacional. Lo cual lo hace idóneo para este proyecto ya que se va a entrenar el agente en un entorno virtual en una primera etapa. [13][14]

- Hindsight Experience Replay (Her)

  En español repetición de la experiencia a posteriori. Este algoritmo trabaja en tiempo discreto y discretiza el entorno. Tiene un estado $S=\{0,1\}^n$ y una acción $A= \{0,1,\ldots,n-1\}$ para un entero n que se ejecutará en bucle. Se va a utilizar este algoritmo en lugar de otros, debido a que no hay que modelar una función compleja de recompensa para que el agente aprenda y a que otros algoritmos no aprenden para muestras superiores a n=40 ya que no alcanzan el objetivo. Para ayudar al algoritmo a que converja en menos interacciones se introduce un valor ε.

  $$r(s, a, g) = -[|g - s_{object}| > \varepsilon\ ]$$

  Donde g es el *goal* (objetivo).

  Donde s_{object} es el punto del entorno donde se encuentra el agente.

Y donde r(s,a,g) es la recompensa obtenida al haber tomado una acción concreta y llegar a un estado en función del objetivo.

Como se trata de una función de recompensa booleana, cuando no se ha llegado al objetivo será TRUE y se obtendrá un 1 que con el menos de delante será negativo y se obtiene una recompensa negativa por cada vez que no ha llegado. De tal forma que el agente entiende que no está haciéndolo bien. En caso contrario, si se ha llegado al objetivo se empiezan a obtener ceros y la recompensa acumulada se mantiene constante.

Lo particular de este algoritmo además de su sencillez es que antes de terminar de ejecutarse, y una vez han terminado todas las acciones, cambia su objetivo inicial por un nuevo objetivo que es la posición a la que ha llegado. De esta manera, el algoritmo siempre le dice al agente que ha llegado a su objetivo para hacerle creer que ha llegado a su objetivo real y así, facilita el aprendizaje y evita en futuras iteraciones exploraciones tan grandes.

Además, este algoritmo no solo aprende con muy pocas recompensas, también funciona mejor con recompensas escasa que con recompensas complejas. Gracias a este algoritmo se puede solucionar en parte el problema de la recompensa compleja que se tiene en este proyecto, la parte de llegar al objetivo la cumpliría. [15][13]

Capítulo 4. COMUNICACIONES DEL ROBOT

En este capítulo se van a tratar dos temas fundamentales. Por un lado, la comunicación del brazo robótico con Python y, por otro lado, los comandos necesarios de la programación RAPID para poner en movimiento el robot.

4.1 COMUNICACIÓN POR SOCKETS

En este proyecto es necesario comunicar el brazo robótico IRB 120 con un código en Python de manera que estos puedan intercambiar información en el momento. La información que se van a transmitir son las posiciones de cada uno de los seis ángulos del brazo robótico, la potencia del robot, la energía acumulada y el tiempo. Este proceso de comunicación se va a realizar cada vez que el código de Python termine una ejecución y decida a qué punto moverse de manera que Python envía la siguiente posición de los ángulos del robot y recibe tanto la energía consumida, como la potencia del robot y el tiempo de ejecución. Es necesario que reciba los datos de energía porque tiene que acumular una función recompensa que también depende del tiempo.

Idealmente la comunicación se haría por sockets, que son unos protocolos que permiten el intercambio de datos a través de un par de direcciones IP local y remota, un protocolo de transporte y un par de números de puerto local y remoto. Los sockets permiten implementar una arquitectura cliente-servidor, la cual la inicia el servidor (RobotStudio) y el cliente (Python) está a la espera de que se inicie esta comunicación. La comunicación se realiza mediante un protocolo TCP/IP (Transmission Control Protocol/ Internet Protocol) y se trata de una comunicación rápida y eficiente. Sin embargo, existe un problema para implementar la comunicación mediante sockets y es que el agente utilizado proviene de una fuente de código abierto (OpenAI) [16] que bloquea la comunicación entre puertos y, por tanto, bloquea la comunicación mediante sockets.

Para solucionar este problema de la comunicación entre el ordenador y el robot físico se implementó la edición de archivos del servidor *File Transfer Protocol* (FTP) del brazo robótico. De esta forma, el ordenador y el robot tienen acceso y pueden manipular un archivo .txt del servidor FTP creado por el controlador del brazo robótico. Al tener acceso ambos al archivo .txt pueden manipular su contenido de tal manera que en algunas ocasiones es Python el receptor de la información y en otras el emisor. Ocurro de igual manera con el controlador del brazo robótico, unas veces envía datos y otras recibe (Ver esquema en Figura 6). La información que se intercambian es la comentada anteriormente (posición, energía, potencia y tiempo). Este tipo de comunicación es más lenta y menos eficiente y en caso de que los datos se envían demasiado rápido, la comunicación puede saturar, lo cual llevaría a una pérdida de información y a que o bien no se calculase bien la función de recompensa o a que no se enviasen todas las posiciones del robot. Lo que probablemente bloquease el robot por completo.

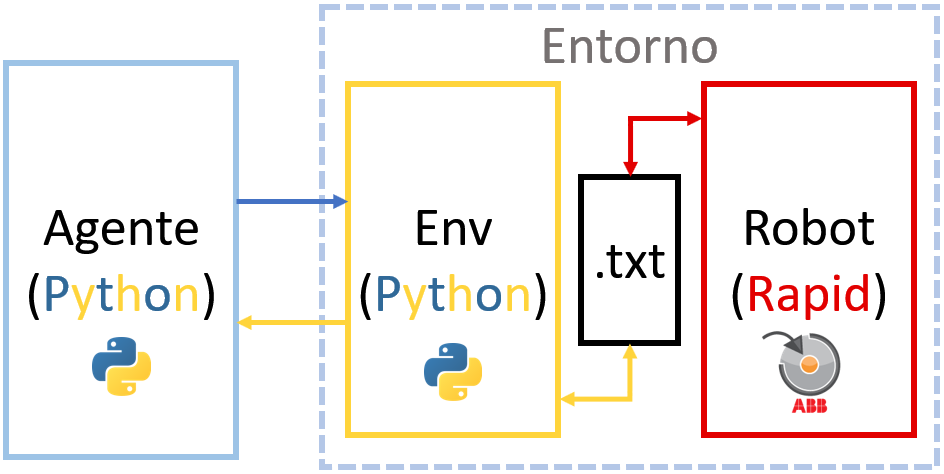


Figura 6: Esquema de la comunicación entre Python y RobotStudio

4.2 MOVIMIENTOS DEL ROBOT EN RAPID

En esta segunda parte del capítulo se van a explicar los comandos básicos de RAPID que se han usado para programar el código del robot.

Existen varios movimientos del brazo robótico: MoveJ, MoveL y MoveC. MoveJ corresponde con el movimiento que busca la mayor optimización de la ruta. MoveL permite al robot moverse solo linealmente en los ejes de coordenadas X, Y y Z. MoveC únicamente deja al robot moverse en movimientos circulares. El primer movimiento que se va a probar es el MoveJ, por ser el idónea para el proyecto.

El código empleado para el movimiento del robot se ha programado dentro de la aplicación de RobotStudio, está en lenguaje RAPID y es el siguiente:

```
MODULE Module1

   PERS robtarget ini := [[364.35,0,594],[3.2819E-8,0,1,0],[0,-1,-
1,0],[9E+9,9E+9,9E+9,9E+9,9E+9,9E+9]];
   PERS pos p_mov_ini := [100,250,300];
   PERS pos p_mov_fin := [250,-300,100];


   PROC main()

      var robtarget p10;
      var robtarget p20;
      var robtarget p30;
      var robtarget p40;
      var num i;

      p10 := CRobT(\Tool:=tool0 \WObj:=wobj0);
      p20 := p10;
      p20.trans := p_mov_ini;
      p30 := p10;
      p30.trans := p_mov_fin;

      MoveJ ini, vmax, fine, tool0;
      MoveJ p20, vmax, fine, tool0;
      WaitRob \ZeroSpeed;
      WaitTime 3;

         FOR i FROM 1 TO 150 STEP 1 DO
         MoveJ p30, vmax, fine, tool0;
         WaitRob \ZeroSpeed;
         MoveJ p20, vmax, fine, tool0;
         WaitRob \ZeroSpeed;
         WaitTime 1;
```

```
    ENDFOR
  ENDPROC
ENDMODULE
```

En este código se comienza fijando tres posiciones: una primera posición para empezar el movimiento, una posición de inicio de cada ciclo y una posición que tiene que alcanzar en mitad del ciclo. Estas posiciones se han elegido según los límites de funcionamiento del robot. Una primera posición `PERS robtarget ini := [[364.35,0,594],[3.2819E-8,0,1,0],[0,-1,-1,0],[9E+9,9E+9,9E+9,9E+9,9E+9,9E+9]];` que se corresponde con un punto en posición vertical, con desplazamiento en el eje X, ya que si no, se daría con el propio robot, y nada de desplazamiento en el eje Y, esto es básicamente para diferenciar el primer movimiento del resto. Una segunda posición `PERS pos p_mov_ini := [100,250,300];` que se define como la posición inicial de cada ciclo que se encuentra, viendo el robot de frente, a la derecha. Y una tercera posición `PERS pos p_mov_fin := [250,-300,100];` que se define como la posición final de cada ciclo que se encuentra, viendo otra vez el robot de frente, a la izquierda y un poco más abajo y acercada a nosotros que la posición inicial.

A continuación, se asigna a cada variable p (p10, p20, p30…) una posición. Esta línea de código `p10 := CRobT(\Tool:=tool0\WObj:=wobj0);` nos indica cómo queremos que se mueva el robot y esta otra `p20.trans := p_mov_ini;` nos indica a dónde se va a mover, no es la función de movimiento. Con esta línea `MoveJ ini, vmax, fine, tool0;` sí que se da la orden de hacer el movimiento que lo va a llevar a la posición primera a velocidad máxima y con un movimiento fino. Después se mueve a la posición inicial de cada ciclo y se esperan 3 segundos hasta comenzar una secuencia de 150 ciclos `FOR i FROM 1 TO 150 STEP 1 DO`. En esos 150 ciclos se va a mover de la posición `[100,250,300]` a la posición `[250,-300,100]`. El cambio de destino de posición se produce cuando el brazo robótico alcanza velocidad 0 `WaitRob \ZeroSpeed`. Entre ciclo y ciclo se va a esperar 1 segundo que se podrá acortar varias décimas de segundos. Sirve para saber cuándo se ha terminado un ciclo y poderlos filtrar en el código de MATLAB. Este tiempo se podría disminuir hasta que todavía se pueda distinguir cuando ha caído la energía debida a la pausa (ver Figura 10 el final y el principio). [17]

Para poder mover el robot se necesita una herramienta que es el FlexPedant, que consiste en un controlador manual en el que se puede elegir la potencia de los motores y se puede tener acceso al código en RAPID, lo cual moverá el robot según el código anterior. El FlexPedant sirve también para controlar el robot manualmente, tiene un botón que sirve de seguro de tal forma que si no se aprieta o si se aprieta demasiado el robot se para evitando que golpee alguna parte del entorno. Sin embargo, para poner el robot en modo automático hay que conectarlo a través de la aplicación de RobotStudio y ya no hacía falta usar el FlexPedant y tener apretado el botón que evite colisiones. Hay que trabajar con una secuencia de movimientos que estén fuera del alcance de otros elementos del entorno. Esta conexión se realiza a través de una dirección IP a la cual se conectan permitiendo un acceso en el controlador.

Capítulo 5. BENCHMARK Y ANÁLISIS DE RESULTADOS

Un *benchmark* es un proceso en el cual se toman como referencias una serie de productos, servicios o características para posteriormente compararlos con un trabajo posterior. En términos informáticos son un conjunto de pruebas que se le realizan a un ordenador para saber cómo de bueno es en comparación con otros y ver en qué puntos flaquea con respecto a los demás.

5.1 DISEÑO DE UN BENCHMARK

En este proyecto se va a diseñar un *benchmark* o marco de referencia que tiene como objetivo estimar el consumo energético de un robot para después aplicar técnicas de aprendizaje por refuerzo sin ver comprometido su tiempo de ciclo. El marco de referencia consiste en una serie de campañas de medida en diversas situaciones que simulen lo que ocurre en una fábrica industrial. De bajada y subida, de desplazamiento lateral, de desplazamiento lateral y subida a la vez, de coger y dejar un objeto, etc. Estas campañas de medida tienen como objetivo obtener el consumo energético y el tiempo de ciclo del robot industrial.

Una vez diseñadas estas campañas de medida se procede a la acción, se va a utilizar un brazo robótico de 6 grados de libertad, el IRB120 del laboratorio de Automatización y Sistemas Digitales del ICAI con un programa que lleva incorporado llamado *RobotStudio*. A la hora de poner en marcha el robot pueden aparecer problemas como que el robot no puede llegar pasar por una posición debido a la orientación de sus ejes, tiene unos límites de funcionamiento en los cuales hay que trabajar. Estos límites se muestran en las Figura 7 y Figura 8, superar los mismos puede hacer que el brazo robótico se bloquee. Por ello, hay que tenerlos en cuenta cuando se programen los puntos iniciales y finales.

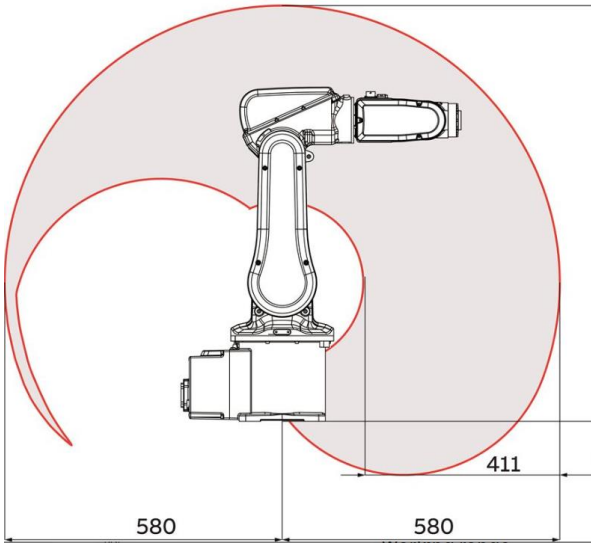


Figura 7: Vista de perfil de los límites de funcionamiento del brazo robótico IRB 120

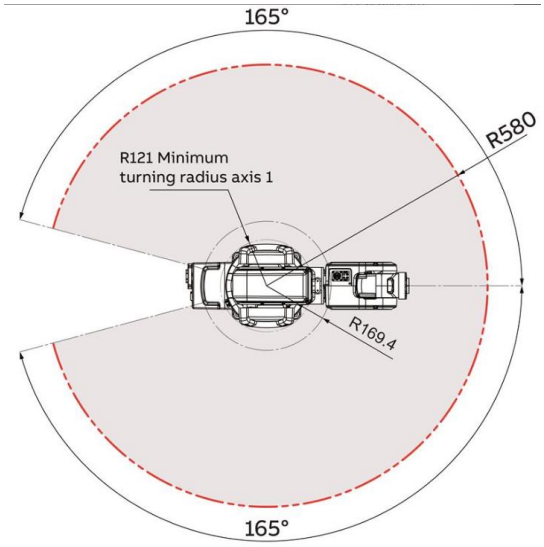


Figura 8: Vista de planta de los límites de funcionamiento del brazo robótico IRB 120

RobotStudio lleva incorporada una herramienta que permite obtener de manera remota las diferentes señales del brazo robótico. Se cogerán la potencia de motor y la energía consumida por el mismo, así como el tiempo. La herramienta se llama "signal analyzer online" y permite posteriormente exportar los datos a un archivo .csv, .txt o .xml. Esta aplicación también permite obtener los ángulos de orientación del brazo robótico, que pueden ser útiles de cara luego a optimizar el consumo energético.

Para el posterior análisis de los resultados se va a optar por la opción de .xml ya que es más accesible, ordena por columnas los resultados y nos da una primera imagen de los resultados obtenidos. Para el procesado de los datos se va a utilizar MATLAB por el alto nivel de procesado que tiene y la gran capacidad de computación.

```
tiempo=xlsread('Prueba2MoveJ.xlsx','A892:A17994');
potencia=xlsread('Prueba2MoveJ.xlsx','C892:C17994');
energia=xlsread('Prueba2MoveJ.xlsx','B892:B17994');

%gráfica de la potencia consumida

hold on
plot(tiempo,potencia)
j=1;
```

```
primer_num=1;
vec_energia(1,1)=energia(1);
vec_tiempo(1,1)=tiempo(1);

%iteracion para obtener el consumo energético y tiempo por ciclo

for i=1:length(energia)
    if primer_num==1
        if potencia(i)<6
            primer_num=0;
            vec_energia(j,2)=energia(i);
            vec_energia(j+1,1)=energia(i);
            vec_tiempo(j,2)=tiempo(i);
            vec_tiempo(j+1,1)=tiempo(i);
            j=j+1;
        end
    else
        if potencia(i)>25
            primer_num=1;
        end
    end
end

% elimina primer y ultimo elem del vector

vec_energia(j,:)=[];
vec_tiempo(j,:)=[];
vec_energia(1,:)=[];
vec_tiempo(1,:)=[];

%resta para obtener el intervalo

[m,n]=size(vec_energia);
for k=1:m
    vec_energia_step(k,1)=vec_energia(k,2)-vec_energia(k,1);
    vec_tiempo_step(k,1)=vec_tiempo(k,2)-vec_tiempo(k,1);
end

mod2 = fitlm(1:length(vec_energia_step),vec_energia_step);

% consumo por ciclo

figure; hold on;
plot(1:length(vec_energia_step),vec_energia_step);
plot(1:length(vec_energia_step), mod2.Fitted,'*');
grid;

% tiempo de ciclo

figure; hold on;
plot(1:length(vec_energia_step),vec_tiempo_step)
grid;
```

Este código coge los datos de potencia, energía y tiempo almacenados en un Excel (`'Prueba2MoveJ.xlsx'`). Se muestra una primera gráfica (Ver Figura 9) sobre cómo ha sido la potencia requerida por el robot para moverse de un punto a otro. Se ve cuáles son los

valores límite que van a servir para filtrar las secuencias y calcular después la energía consumida en cada ciclo, así como el tiempo de operación de dicho ciclo. En la Figura 10 se puede ver cómo es un ciclo de consumo energético de un movimiento completo.

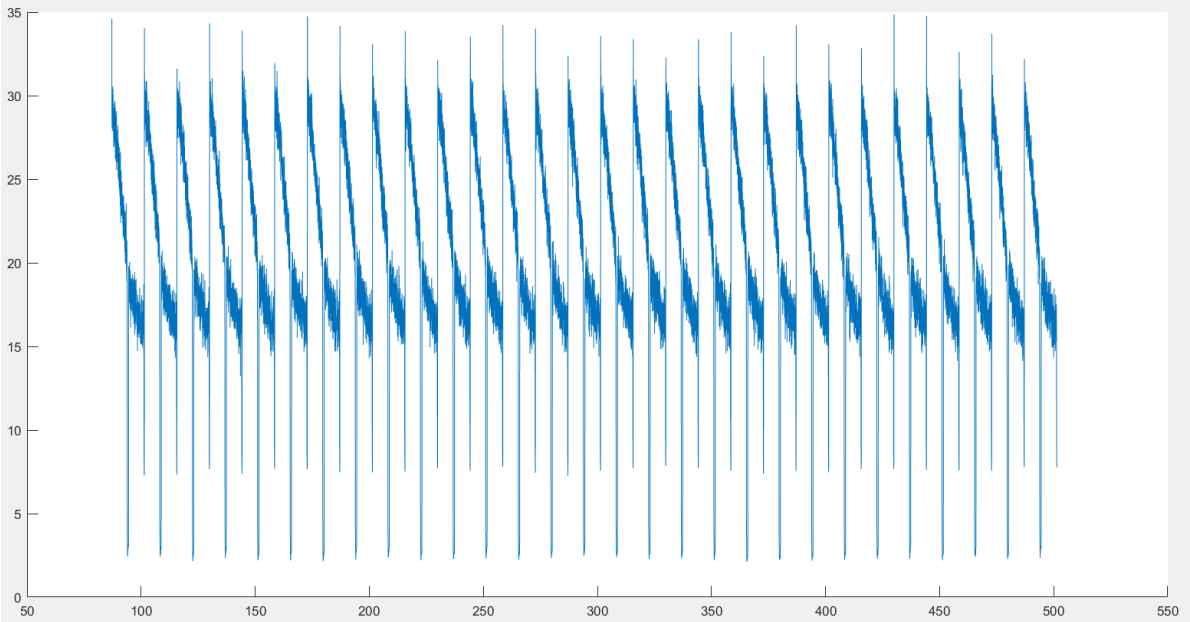


Figura 9: Potencia en W aplicada por el brazo robótico en toda la secuencia usando MoveJ

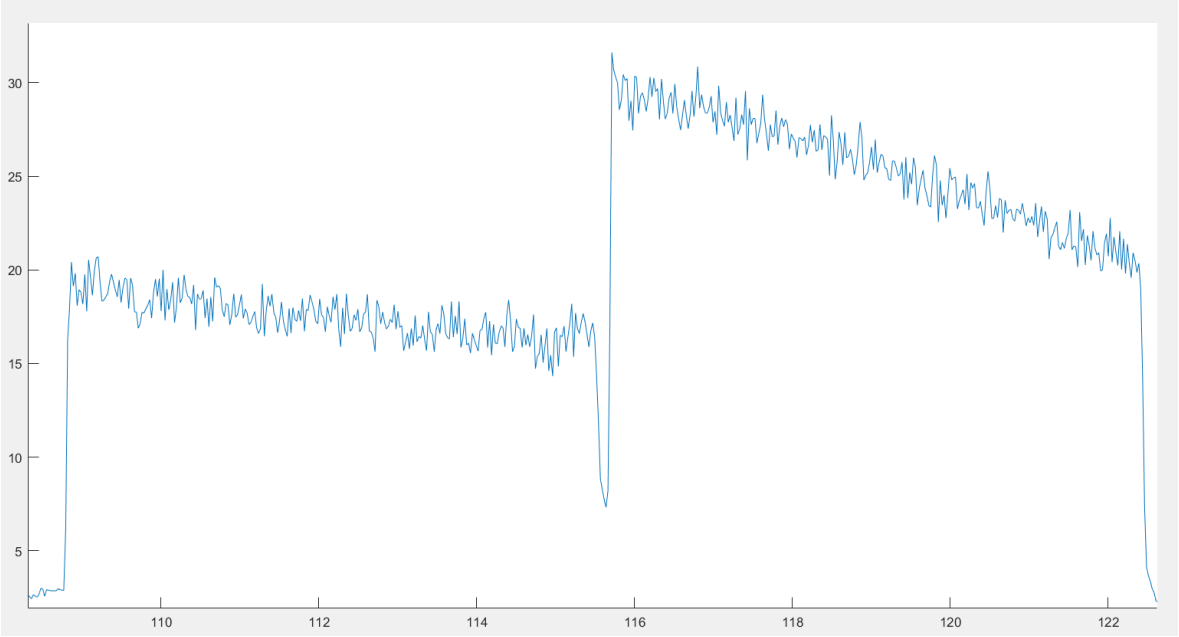
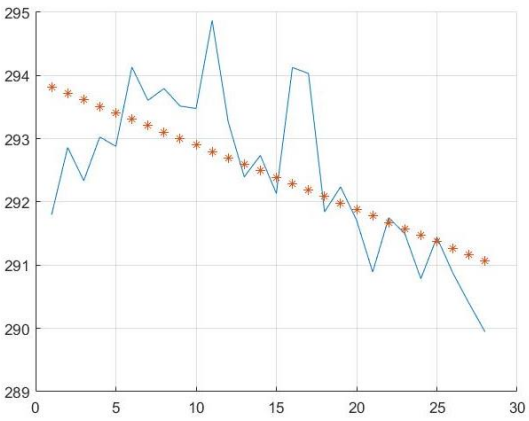


Figura 10: Potencia en W aplicada por el brazo robótico en un ciclo usando MoveJ

Siguiendo con la explicación del código, se almacenan los primeros valores de energía y de tiempo en una variable. Después se recorre un bucle for a lo largo de todos los valores extraídos del Excel. El filtro se basa en que una vez baja de un determinado valor (en este caso 6W `if potencia(i)<6`) se desactiva un flanco de subida (`primer_num=0;`) y se toman los valores de energía y tiempo del instante inicial del ciclo que coinciden con el instante final del anterior ciclo. Se va a ir incrementando un contador `j` cada vez que se recorra un ciclo. Después, una vez suba de un determinado valor (en este caso 25W `if potencia(i)>25`) se activa el flanco de subida (`primer_num=0;`) para indicar que el robot ha alcanzado su consumo máximo de energía y se dispone a volver a la posición inicial; comienza entonces la fase de subida.

A continuación, se restan los valores de inicio y fin de un ciclo tanto en energía como en el tiempo (`vec_energia_step(k,1)=vec_energia(k,2)-vec_energia(k,1);`) Posteriormente, se eliminan el primer y último vector por quedar distintos a los demás. Esto se debe a que cuando se dejan de tomar los datos de potencia, energía y tiempo no se sabe exactamente en qué punto corta y son menores de un ciclo completo. Y al final del código se muestran el consumo energético (Ver nFigura 11) y el tiempo de ciclo (ver Figura 12). Para el consumo energético se muestra una recta de regresión lineal para ver la tendencia de los datos. Para ello, se emplea una función de MATLAB (`fitlm`) que calcula una recta de regresión según el modelo deseado, bien podría ser cuadrático, potencial, exponencial o logarítmico [18]. En vista de otros gráficos obtenidos se ha decidido optar por una aproximación lineal para ver principalmente la tendencia.

A continuación, se muestran los datos de potencia total y la potencia de un ciclo:



nFigura 11: 30 ciclos de energía en J

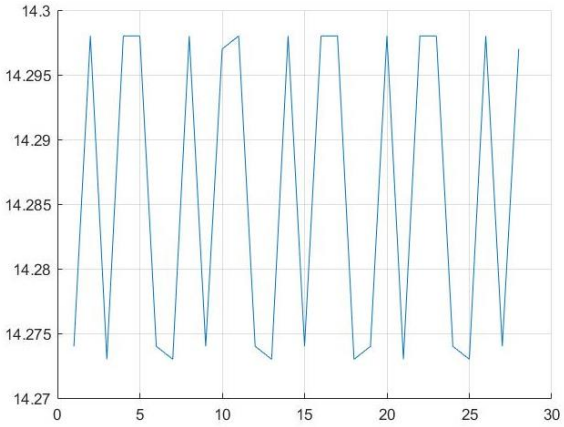


Figura 12: 30 ciclos de tiempo en s

Se puede observar como en las gráficas anteriores la energía parece que disminuye un poco mientras que el tiempo parece que fluctúa en torno a 2 valores. Esto es debido a que no se puede calcular con total exactitud cuando empieza y cuando termina un ciclo y se depende del tiempo de muestreo (25ms).

Si a continuación se toman más muestras se puede ver una tendencia más generalista (Figura 13) y se pueden sacar conclusiones. No se incluye la gráfica del tiempo por ser semejante a la Figura 12.

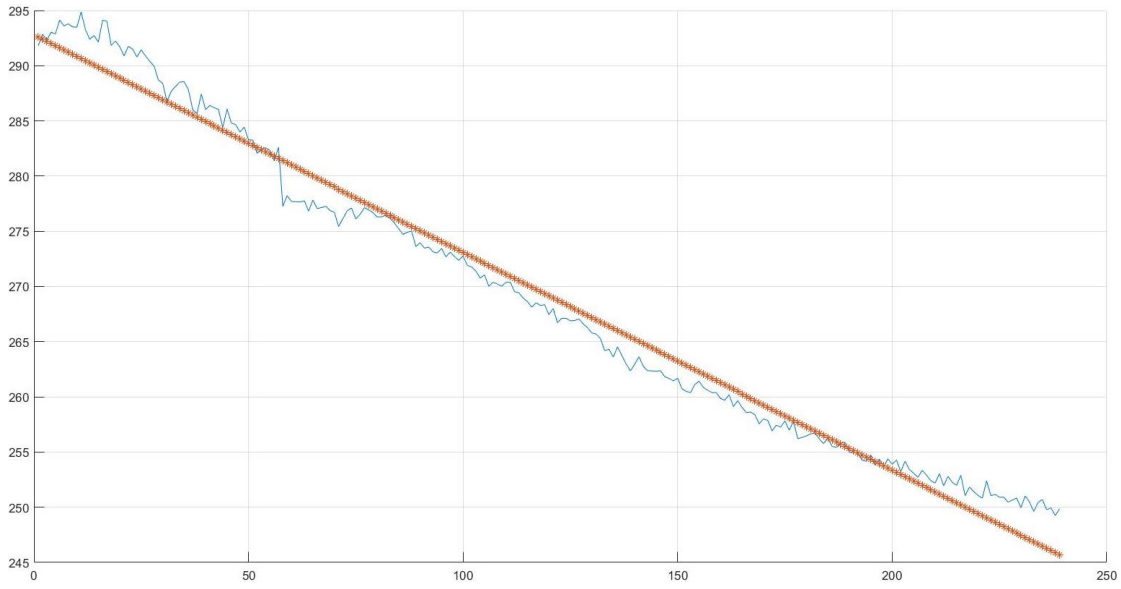


Figura 13: 240 ciclos de energía usando MoveJ

En vista de que se ha multiplicado aproximadamente por 10 el número de ciclos y haber obtenido una muestra más considerable, se puede decir que se confirma la tendencia de bajar el consumo. Se comprobará más adelante si esto es debido al movimiento con Move J que busca optimizar el movimiento o si se debe a temas de temperatura y rozamiento.

A continuación, se va a emplear el mismo código utilizado anteriormente y se van a mostrar los resultados obtenidos con el movimiento MoveL Un ciclo de potencia queda de la siguiente forma (Ver Figura 14):

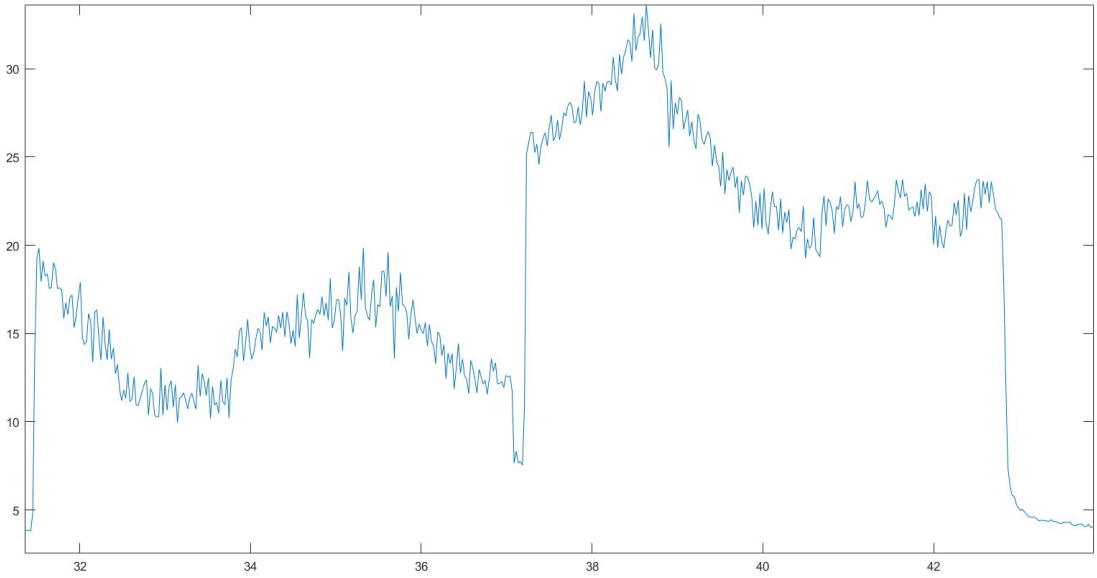


Figura 14: Potencia en W aplicada por el robot en un ciclo usando MoveL

Varios ciclos de energía quedan de la siguiente forma (Ver Figura 15):

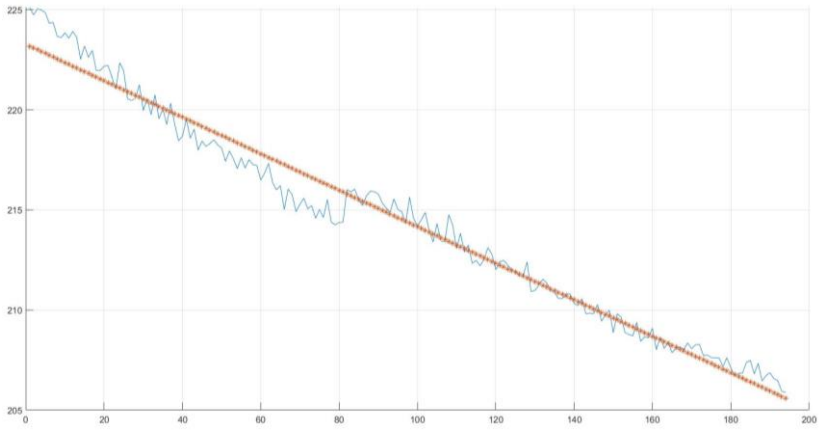


Figura 15: 200 ciclos de energía usando MoveL

Una vez más se puede ver como hay una línea de tendencia que decrece según va pasando el tiempo. En esta ejecución se han tardado 12.52*200=42 minutos aproximadamente, pero no parece que vaya estabilizando. Se va a hacer una última prueba con el robot en modo automático ya que trabaja a mayor velocidad y potencia y simula mejor el trabajo en fábrica. En vista de estos resultados se descarta la hipótesis de pensar que la energía disminuía gracias al uso del movimiento MoveJ.

Como última prueba se va a realizar una puesta en marcha del robot en modo automática, usando MoveJ y empleando el tiempo necesario hasta que llegue al régimen permanente para así poder calcular los valores medios de energía y tiempo de cada ciclo que realiza el brazo robótico.

Una primera gráfica para mostrar cómo es el ciclo de potencia de un solo movimiento (Ver Figura 16).

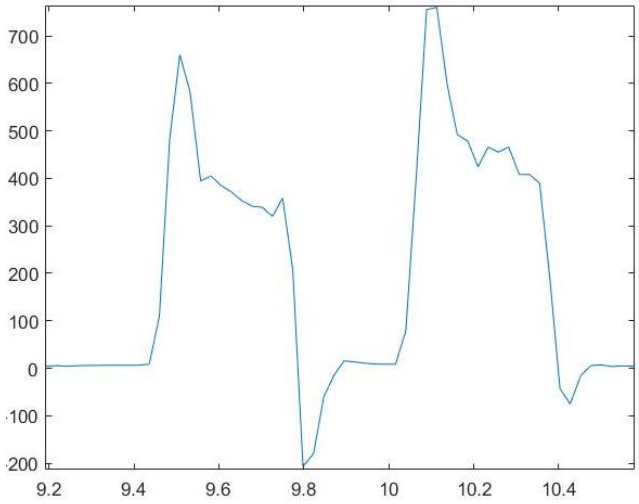


Figura 16: Potencia en w aplicada por el robot en modo automático en un ciclo usando MoveL

En comparación con las Figura 13 y Figura 15 se puede observar como el tiempo de esta es mucho menor (en torno a 1,4 s comparado con los 14 y 12,5 de gráficas anteriores). Debido a ello, el número de muestras tomadas por ciclo es mucho menor. Es cierto que se ha disminuido el ruido en la gráfica, pero al ser tan grande el tiempo de muestreo, en comparación con el tiempo de ciclo, se puede estar tomando puntos en los que el ruido momentáneo sea máximo ya que este ruido sigue estando solo que no se aprecia en la gráfica.

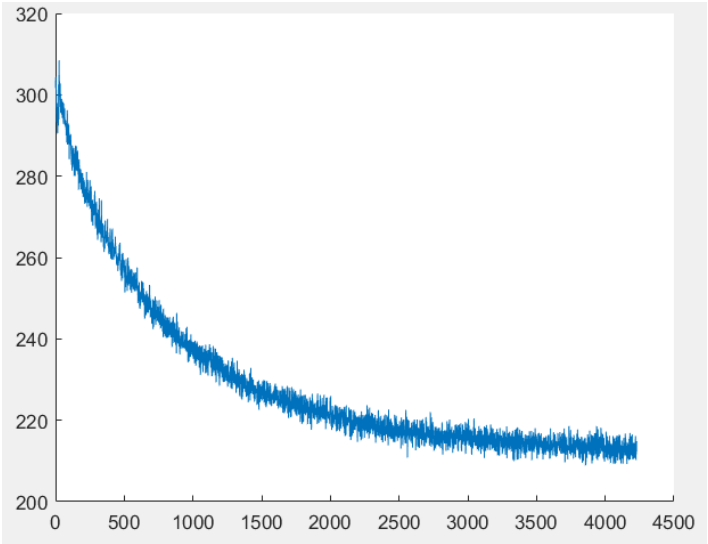


Figura 17: 4250 ciclos de energía usando el robot en modo automático y MoveL

Se puede observar como la energía disminuye como si de un régimen transitorio y exponencial se tratase. El tiempo para obtener estos resultados ha sido 1.4*4250 = 1 hora y media. Con lo cual, queda descartada la hipótesis de pensar que era por no poner el robot en modo automático, quedando como única hipótesis la de que a medida que pasa el tiempo el robot se calienta y al calentarse el coeficiente viscoso del aceite y las grasas de los engranajes y los ejes del brazo robótico disminuye. En consecuencia, hay menos perdidas por rozamiento y fricción y el brazo robótico consume menos. De ahí la necesidad de llevarlo a un régimen permanente a partir del cual se trabaja.

5.2 ESTIMACIÓN ESTADÍSTICA

Una vez obtenidos el consumo energético y el tiempo de operación de cada ciclo se procede a realizar una estimación estadística para establecer cuáles son los valores que se van a usar de referencia en la función de recompensa de la sección 2.2.4. En este caso, se va a estimar el tiempo medio y el consumo energético de cada ciclo con un intervalo de confianza al 95%. Cuanto más grande sea la muestra (nº de ciclos) más precisos serán los resultados. Se va a usar la siguiente ecuación.

$$\mu \in \left[\bar{X} \pm t_{n-1,\frac{\alpha}{2}} \cdot \frac{S_x}{\sqrt{n}}\right]_\gamma$$

Estimación de la media desconocida la varianza de una población

Donde:

μ es la estimación de la media

$\bar{X}$ es la media muestral de los ciclos obtenidos

$t_{n-1,\frac{\alpha}{2}}$ es una distribución T-Student de n-1 grados de libertad

S_x es la cuasidesviación típica muestral

n es el número de datos de la muestra

γ es el nivel de confianza del intervalo

Para ello, se van a coger los últimos valores de la Figura 17 que corresponden al régimen permanente. Se van a introducir en una hoja Excel, debido a la facilidad que da para manejar funciones estadísticas, y se va a calcular la media del consumo energético un ciclo de trabajo. Esta hoja de Excel es completamente reutilizable para estimar otros resultados de otras campañas de medidas. Los resultados obtenidos se pueden ver en la siguiente figura (Ver Figura 18)

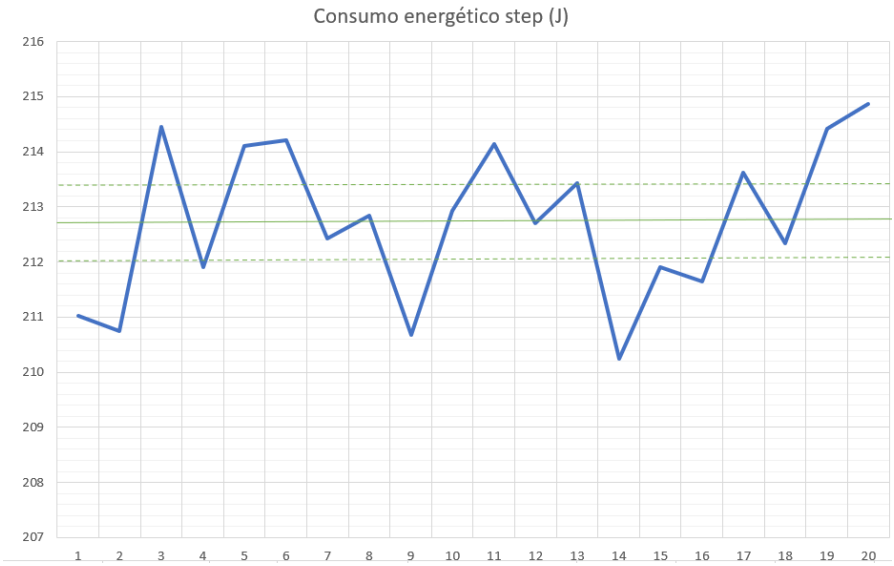


Figura 18: Estimación de la media muestral del consumo energético en J

Se han cogido 20 muestras cuya media ha sido de 212.73 J, cuyo intervalo de confianza ha resultado ser [212.07 - 213.39]. Como se puede observar, hay pocos valores dentro del intervalo, sin embargo, eso no es importante, lo que quiere decir el intervalo de confianza es que la media de la muestra va a estar entre esos valores. Si se aumentase al 99% de confianza el intervalo quedaría [211.83 - 213.63]. Y si se disminuyese al 90% de confianza [212.19 - 213.28] se acotaría aún más el valor de la media muestral que es el que se va a usar de referencia. Otra forma de estimar es coger la media y sumarle y restarle 2 desviaciones típicas y el intervalo queda [210-215.5] que puede ser ampliado a [209,32 - 216,14] si se toma como valores límite los del primer intervalo calculado. De esta manera se sabría que el 95% de las ocasiones el valor de un ciclo va a estar dentro del intervalo.

Capítulo 6. CONCLUSIONES Y TRABAJOS FUTUROS

En este trabajo se ha diseñado un controlador por aprendizaje por refuerzo donde se ha tenido que atender a dos temas fundamentales: las comunicaciones entre el controlador y el brazo robótico y el diseño de un *benchmark*.

El principal problema que se ha encontrado en el tema de las comunicaciones ha sido la implementación de los sockets, que son unos protocolos de comunicación, que permiten una comunicación fluida y eficiente. Los sockets han dado problemas debido a que el agente de Python bloqueaba este tipo de comunicación. Para poder continuar con el proyecto se ha optado por una comunicación mediante archivos .txt, una solución rudimentaria que ha permitido seguir desarrollando este proyecto.

Sobre el estudio que se ha realizado acerca de los algoritmos de aprendizaje por refuerzo para determinar cuál es el que mejores resultados ofrecería en este proyecto, se ha determinado que el que ofrecería unos resultados más prometedores sería HER. Este algoritmo se podría combinar con otro como DDPG que trabaja en espacios continuos y que podría dar más robustez al aprendizaje.

Realizar unos diseños de referencia o *benchmarks* es algo básico y esencial en el aprendizaje por refuerzo. Siempre se trabaja con este tipo de diseños para comprobar si realmente ha sido efectivo el aprendizaje automático. Este tipo de diseños también se emplean para comparar unas políticas con otras y unos algoritmos con otros. Se emplean para saber cuál es el porcentaje de éxito de un algoritmo y comparar también la curva de rendimiento con otros. Por ello, es necesario trabajar en régimen permanente y no con tendencias decrecientes de energía. El principal motivo por el que había un régimen transitorio en el que la energía disminuía era porque al calentarse el aceite y la grasa de los ejes y las juntas de unión de los mismos, el coeficiente de viscosidad de estos disminuye, lo cual reduce las pérdidas por fricción y en consecuencia, el consumo de energía. En este caso el diseño del *benchmark* no se ha podido finalizar al completo con resultados satisfactorios debido a que no se ha podido

acceder al laboratorio de ICAI. Habría que realizar más campañas de medidas para poder concluir que los resultados obtenidos son totalmente válidos.

En futuros proyectos sería conveniente retomar la comunicación con sockets ya que aumentan la eficiencia de manera considerable y tienen menos posibilidades de saturar y perder información. Y en cuanto al tema del aprendizaje por refuerzo se va a seguir trabajando con el controlador del brazo robótico con imágenes ya que ofrecen una visión más real sobre el entorno. Las imágenes obtenidas por la cámara darán un enfoque sobre el *Deep Reinforcement Learning*, más en línea con los conceptos generales del aprendizaje por refuerzo que ya se están aplicando en las investigaciones más punteras.

BIBLIOGRAFÍA

[1] "Executive Summary World Robotics 2019 Industrial Robots", International Federation of Robotics, septiembre 2019. Último acceso 26/06/2020 https://ifr.org/downloads/press2018/Executive%20Summary%20WR%202019%20Industrial%20Robots.pdf

[2] Okuma, T.," Editorial WR 2019 Industrial Robots", International Federation of Robotics, septiembre 2019. Último acceso 26/06/2020 https://ifr.org/downloads/press2018/Editorial_WR_2019_Industrial_Robots.pdf

[3] Torrent-Sellens, J., "Las empresas industriales en 2014 y 2015", fundación SEPI, F.S.P., octubre de 2017. Último acceso 26/06/2020 https://www.fundacionsepi.es/investigacion/esee/LasEmpresasIndustriales2015.pdf

[4] Torrent-Sellens, J., "Las empresas industriales en 2016, Robótica, productividad y empleo en la empresa industrial", fundación SEPI, F.S.P., junio de 2018. Último acceso 26/06/2020 https://www.fundacionsepi.es/investigacion/esee/LasEmpresasIndustriales2016.pdf

[5] Wiering, M. y Van Otterlo, M. "Reinforcement Learning State-of-the-Art", Springer-Verlag Berlin Heidelberg 2012, páginas 1-31. Último acceso 23/06/2020 https://link.springer.com/content/pdf/bfm%3A978-3-642-27645-3%2F1.pdf

[6] Sancho Caparrini, F. "Aprendizaje por refuerzo: algoritmo Q Learning" Última modificación Marzo 2019. Último acceso 23/06/2020. http://www.cs.us.es/~fsancho/?e=109

[7] Li, Y. "Deep Reinforcement Learning: an overview" Noviembre 2018. Último acceso 23/06/2020 https://arxiv.org/pdf/1701.07274.pdf

[8] Plappert, M., et al. "Multi-goal Reinforcement Learning: Challenging Robotics Enviroments and Request for Research", openAI, marzo 2018. Último acceso 26/06/2020 https://arxiv.org/pdf/1802.09464.pdf

[9] S. Amarjyoti, "Deep reinforcement learning for robotic manipulation-the state of the art", 2017, Robotics Institute, School of Computer Science, Carnegie Mellon University. Último acceso 26/05/2020 https://arxiv.org/pdf/1701.08878.pdf

[10] T. Johannink et al., "Residual Reinforcement Learning for Robot Control," 2019 International Conference on Robotics and Automation (ICRA), Montreal, QC, Canada, 2019, pp. 6023-6029. Último acceso 18/05/2020 https://ieeexplore.ieee.org/abstract/document/8794127

[11] R. Meyes et al., "Motion Planning for Industrial Robots using Reinforcement Learning," Procedia CIRP, Aachen, Germany, Volume 63, 2017, Pages 107-112. Último acceso 27/05/2020 https://www.sciencedirect.com/science/article/pii/S221282711730241X

[12] P. Kormushev, B. Ugurlu, S. Calinon, N. G. Tsagarakis and D. G. Caldwell, "Bipedal walking energy minimization by reinforcement learning with evolving policy parameterization," 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems, San Francisco, CA, 2011, pp. 318-324. Último acceso 27/05/2020 https://ieeexplore.ieee.org/document/6094427

[13] Silver, D. et al. "Deterministic Policy Gradient Algorithms", DeepMind Technologies, 2014. Último acceso 25/06/2020 http://proceedings.mlr.press/v32/silver14.pdf

[14] Lillicrap, T. et al. "Continuous control with deep reinforcement learning", Google Deepmind, julio 2019. Último acceso 25/06/2020 https://arxiv.org/pdf/1509.02971.pdf

[15] Andrychowicz, M. et all, "Hindsight Experience Replay", OpenAI, Febrero 2018 https://arxiv.org/pdf/1707.01495.pdf

[16] OpenAI. Último acceso 22/06/2020 https://github.com/openai/baselines/tree/master/baselines

[17] ABB, "Operating Manual RobotStudio" 2020. Último acceso 9/03/2020

https://library.e.abb.com/public/041fef81c5344fd589bf1b499779149a/3HAC032104%20OM%20RobotStudio-en.pdf

[18] Mathworks. Último acceso 20/02/2020
https://es.mathworks.com/help/stats/fitlm.html

ANEXO I-OBJETIVOS DE DESARROLLO SOSTENIBLE

Los Objetivos de Desarrollo Sostenible (ODS) son un conjunto de 17 propuestas globales que se crearon en 2015 para ser cumplidos en 2030. Fueron diseñados bajo el lema de ser "un plan de trabajo para logar un mundo mejor y más sostenible para todos". Los ODS persiguen como objetivos principales poner fin a la pobreza, proteger el planeta y mejorar la vida de las personas. Hoy en día, las medidas para lograr dichos objetivos están por detrás de lo esperado y avanzan a un ritmo más lento del esperado. En esta nueva década que ha empezado, gobiernos y actividad privada se han propuesto acelerar estas propuestas mediante la movilización de financiación y una mayor implicación a nivel nacional.

En este proyecto el principal Objetivo de Desarrollo de Sostenible que se ha abordado ha sido el 9: Industria, Innovación e Infraestructuras. Este ODS persigue "construir infraestructuras resilientes, promover la industrialización inclusiva y fomentar la innovación porque el crecimiento económico, el desarrollo social y la acción contra el cambio climático dependen en gran medida de la inversión en infraestructuras, desarrollo industrial sostenible y progreso tecnológico". Por ello, este proyecto busca mejorar la eficiencia de la industria, ya que busca el ahorro del consumo energético y la optimización de los recursos con vistas a una industria más sostenible. Este proyecto además es un proyecto de investigación e innovación que es escalable, lo cual puede ser aplicable a industrias menos desarrolladas que podrán crecer sosteniblemente. Además, cumple la meta del objetivo 9.2 que es duplicar la contribución en los países menos desarrollados, este proyecto favorece la inclusividad de la industria globalmente, favoreciendo también la creación de empleo en estas industrias menos desarrolladas. [1]

La industria se trata del principal motor de la economía, se estima que por cada empleo creado en la industria se crean 2.2 en otros sectores económicos. Y con este proyecto se cumple el objetivo 9.1 que propone desarrollar infraestructuras que apoyen el desarrollo económico y el bienestar humano, ya que cumple con que los robots favorecen las

condiciones de trabajo de las personas, apartándolas de los trabajos forzosos y de aquellos empleos que supongan un riesgo para su salud. 46[2]

En cuanto a la cuantificación de lo que supondría este proyecto se van a hacer una serie de con los datos de los que se disponen. Se sabe que la potencia de este robot es de 240 W y que no es de los robots industriales más potentes. Los hay que consumen más y que consumen menos y por ello, se va a tomar ese valor como referencia. Se va a tomar un tiempo medio de operación de 12 horas, medio día y se va a suponer que un 80% de los robots mundiales (3,2 millones) están operativos. Esto supondría una energía total de 26.542.080 millones de julios al día y 36.864 MWh. Con una optimización esperada de entre un 1% y un 5%, cogiendo un 2% como valor referencia, se ahorrarían 530.842 millones de julios al día y de 737,28 MWh. Que a un precio de 100 € el MWh se estarían ahorrando al día 73.000 euros y al año 27 Millones de euros. Además, toda esta energía ahorrada supondría también un ahorro en la expulsión de CO2 a la atmósfera de aproximadamente 96 toneladas a la hora. Lo cual estaría promoviendo una industria más sostenible, cumpliendo así con el objetivo número 9.

Este proyecto también está muy ligado con el ODS número 12 que promueve una producción y un consumo sostenible gracias al ahorro de energía y con el número 1 que tiene como fin erradicar la pobreza gracias al ahorro económico que se podría destinar a distribuir la riqueza.

[1] Naciones Unidas, “Industria, Innovación e Infraestructura: por qué es importante” Último acceso 28/06/2020. https://www.un.org/sustainabledevelopment/es/wp-content/uploads/sites/3/2016/10/9_Spanish_Why_it_Matters.pdf

[2] Naciones Unidas, Objetivos de Desarrollo Sostenible. Último acceso 28/06/2020. https://www.un.org/sustainabledevelopment/es/infrastructure/