PLATAFORMA DE PROGRAMACIÓN QUE FACILITE LA CONFIGURACIÓN DE COBOTS POR MEDIO DE LENGUAJE NATURAL

JUAN PABLO BERRUECOS ROLDÁN

Trabajo de grado para optar al título de

INGENIERO MECATRÓNICO

MSc. Andrés Felipe Valle Pérez

Asesor: Juan Camilo Tejada Orjuela

UNIVERSIDAD EIA

INGENIERÍA MECATRÓNICA

ENVIGADO

2019

AGRADECIMIENTOS

A mi familia, por su gran apoyo y amor que me brindaron durante el tiempo que estuve cursando mi pregrado.

A mi mamá, por ser mi ejemplo incondicional y ser la persona que me motivo en los momentos más duros y me felicitó en los mejores momentos.

A mi novia, por ser parte fundamental en mi vida y por ser la persona que luchó codo a codo los momentos difíciles.

A mis profesores, por ser una excelente guía y por permitirme enamorarme más de mi carrera.

CONTENIDO

pág.

LISTA DE TABLAS

pág.

LISTA DE FIGURAS

pág.

LISTA DE ANEXOS

pág.

RESUMEN

El gran avance en la tecnología demanda un recambio continuo de los productos y de los métodos de fabricación, lo que genera que las grandes y pequeñas empresas productoras deban migran de la gran producción por volumen a una producción flexible que permita satisfacer las necesidades cambiantes de su entorno.

Los robots colaborativos son una gran solución para las empresas que requieren automatizar procesos sin necesidad de modificar la línea de producción y que deseen tener un rápido retorno económico, sin embargo, estos robots pierden flexibilidad en la aplicación cuando se necesita que estén ubicados en otro punto de la línea de producción realizando una tarea diferente a la habitual, debido a que esta acción conlleva en cambios de programación. Por este motivo, la creación de una plataforma de programación que facilite la configuración de los robots colaborativos, permitiendo al operario reprogramar el robot simplemente escribiendo la tarea que desea que realice en un aplicativo web, sin ningún tipo de estructura de programación, garantizará la flexibilidad y adaptabilidad al robot colaborativo y una reducción de costos a la empresa.

Para desarrollar esta plataforma se utilizará una tecnología llamada RPA (Robotic Process Automation), con la cual se creará un Bot con la capacidad de interpretar las instrucciones de los usuarios, estructurar el código correspondiente y programar el robot colaborativo. Obteniendo así, una plataforma capaz de implementarse sin necesidad de hardware adicional.

Palabras claves: Robots colaborativos, RPA, lenguaje natural, programación.

ABSTRACT

The great advance in technology demands a continuous change of products and manufacturing methods, which means that large and small manufacture sector companies must migrate from large production by volume to flexible production that allows to meet the changing needs of their environment.

Collaborative robots are a great solution for companies that have to automate processes without modifying the production line and wishing to have a rapid economic return, however, these robots lose flexibility in the application when they need to be located in another point of the production line performing a different task than usual, because this action involves programming changes. For this reason, the creation of a programming platform that facilitates the configuration of collaborative robots, to the operator reprogram the robot simply by writing the task that you want to be performed in a web application, without any programming structure, compatible with the flexibility and adaptability to the collaborative robot and a reduction of costs to the company.

To develop this platform, a technology called RPA (Robotic Process Automation) will be used, with which a Bot will be created with the ability to interpret user instructions, structure the corresponding code and program the collaborative robot. Thus, obtaining a platform capable of implementing without the need for additional hardware.

Keywords: Collaborative robots, RPA, natural language, programming.

INTRODUCCIÓN

El presente trabajo de grado expone una manera de aumentar la implementación de los robots colaborativo (CoBots) en la industria, específicamente en empresas pequeñas o con una línea de producción dedicada a fabricar diferentes productos. Esto debido a que estos robots son capaces de interactuar con los operarios sin necesidad de barreras de seguridad, por lo que ocupan poco espacio en el sitio donde estén situados, y no interfieren de manera invasiva en la línea de producción. Este trabajo se enfoca en generar un canal de comunicación sencillo entre los operarios y los CoBots, con el objetivo de que estos puedan programar las tareas que demanden sus procesos sin necesidad de tener conocimiento técnicos ni teóricos sobre programación e ingeniería.

El desarrollo de esta plataforma de programación básica de CoBot consta de tres fases esenciales:

La primera fase consiste en el desarrollo de una plataforma amigable con el usuario que le brinde la posibilidad de asignar tareas a un CoBot mediante la utilización de comandos sencillos, sin necesidad de utilizar algoritmos de programación, para satisfacer una necesidad específica.

En la segunda fase, se encuentra el diseño y desarrollo de un asistente virtual, haciendo uso de la tecnología RPA, como mecanismo de comunicación entre el usuario y el CoBot; este se encargará de la traducción de órdenes y programación del CoBot. Cabe anotar que para esto se hará uso de subrutinas preprogramadas dentro del robot colaborativo.

Finalmente, se realizará la programación del CoBot BAXTER, siguiendo la lógica anterior, desde el asistente virtual que se encargará de procesar la información ingresada por el usuario.

1. PRELIMINARES

1.1 PLANTEAMIENTO DEL PROBLEMA

La automatización industrial es una importante estrategia de fabricación que emplea técnicas y prácticas modernas en la fabricación de productos para lograr una ventaja competitiva y sostenible en la industria. Por lo tanto, se ha convertido en un área de atención principal para los profesionales, gerentes e investigadores debido a su importante contribución al rendimiento comercial, el costo, la satisfacción del cliente y la rentabilidad *(Acharya, 2017)*.

A grandes rasgos la automatización industrial se puede distribuir en cinco niveles de forma piramidal: el nivel 0 son los sistemas que intervienen directamente en el proceso. El nivel 1 es un control automático y el nivel 2 es un control de supervisión. El nivel 3 es un control de producción y es responsable del mantenimiento, calidad, inventario, etc. Por último, el nivel 4 está principalmente enfocado en funciones de gestión, ventas, marketing, etc. *(Omer, 2014)*.

El nivel cero al ser la base de la pirámide es fundamental. Este nivel está compuesto por diferentes dispositivos de medida, encargados de tomar datos de las variables significativas, y los actuadores, estos últimos son los encargados de recibir la señal de control y de desempeñar las tareas de automatización. Uno de los actuadores más novedosos es el robot colaborativo (CoBot), este es un dispositivo capaz de integrarse con los empleados y adaptarse fácilmente en entornos de fabricación, permitiendo así una interacción directa entre el operario y el robot, eliminando restricciones de seguridad y logrando automatizar diversas tareas que se desarrollan en la industria. *(Koukkari, 2016)*

“Un área de crecimiento para los robots colaborativos está en las Pymes, ya que pueden beneficiarse de la escalabilidad y flexibilidad del robot, pero ve como problemática el hecho de que estas carecen de experiencia interna en robótica, por lo que deben ser fácilmente programables” *(El Financiero, 2018)*.

Como respuesta a esto, los robots colaborativos poseen una función de entrenamiento en la que el operario puede guardar rutinas con la simple acción de mover los brazos del robot en las posiciones deseadas *(Rethink robotics, s.f.)*. Sin embargo, en algunas ocasiones la función de entrenamiento se vuelve engorrosa debido a la complejidad de la tarea. Si se necesita que el robot realice una rutina diferente, sin importar su sencillez, se debe detener el proceso y realizar la programación correspondiente, esto implica gastos en tiempo y en personas especializadas. Asimismo, si se desea que el robot cumpla con diferentes tareas dentro de una línea de producción se debe realizar cada programación o realizar el entrenamiento que genera la rutuna, esto es una tarea repetitiva que le resta flexibilidad a la aplicación de los CoBots.

Por lo anterior, surge la necesidad de la creación de una plataforma de programación que permita configurar las tareas de los robots colaborativos con el simple hecho de describir la acción en un aplicativo web facilitando así la programación de rutinas complejas a personas que carezcan de experiencia en robótica.

1.2 OBJETIVOS DEL PROYECTO.

1.2.1 Objetivo General

Desarrollar una plataforma de programación que permita la configuración de instrucciones a un robot colaborativo por medio de lenguaje natural, utilizando tecnología RPA para la estructuración del código y una interfaz de usuario para la comunicación entre el robot y el usuario, facilitando así la implementación de estos en la industria.

1.2.2 Objetivos Específicos

- Caracterizar las cuatro tareas más comunes que cumplen los robots colaborativos en la industria.
- Programar subrutinas de las tareas caracterizadas en el robot colaborativo Baxter de la Universidad EIA.
- Crear una interfaz de usuario y chatbot para la toma y procesamiento de instrucciones del usuario, con la capacidad de comunicarse con el Bot.
- Construir un Bot para la estructuración de un código, basándose en subrutinas preprogramadas, que cumpla con las instrucciones dadas por el usuario.
- Verificar el funcionamiento del sistema en el robot colaborativo de la Universidad EIA bajo un ambiente de laboratorio.

1.3 MARCO DE REFERENCIA

1.3.1 Antecedentes

Con el objetivo de tomar como punto de partida los hallazgos, conclusiones y resultados de trabajos de grados, investigaciones y patentes relacionadas con el tema a desarrollar, se realizó una búsqueda literaria con el fin de adquirir un conocimiento más amplio sobre la metodología con la que se estructuraron y los objetivos planteados.

El trabajo de grado “Aplicación web móvil con geolocalización para mejorar la experiencia de compra del consumidor de Trujillo en la búsqueda de promociones en supermercados en el año 2016” (Quiroz & Yarlequé, 2017), desarrollado en la facultad de ingeniería de la Universidad Privada Antenor Orrego, de la ciudad de Trujillo, Perú, crearon una aplicación que le permita al usuario visualizar los descuentos ofertados por los supermercados más cercanos sin necesidad de entrar a la página web de cada uno de estos. Para llevar a cabo esta aplicación se utilizó la metodología XP (eXtreme Programming), este enfoque es el más destacado de los procesos ágiles de desarrollo de software y se utiliza para proyectos con equipos pequeños o medianos (Quiroz & Yarlequé, 2017). En el proyecto uno de los objetivos específicos se enfocaba en la creación de un Bot, utilizando RPA, para automatizar el transporte de datos desde un servidor local a las diferentes páginas web de los supermercados (Quiroz & Yarlequé, 2017). En este trabajo de grado se utilizó la tecnología RPA, para automatizar un proceso repetitivo, combinado con un API que le permite obtener información, y se concluyó que Selenium API permite una fácil integración con RPA y posee basta documentación que ayuda en su realización (Quiroz & Yarlequé, 2017).

Para obtener una referencia de un proceso de implementación de un robot colaborativo en una línea de producción, enfocado en los métodos utilizados para la programación, se analizó el trabajo de grado “Implementación de robots colaborativos en línea de producción” (Tabuenca, 2017). En éste, se tenía como objetivo la implementación de un robot colaborativo en una de las líneas de producción de la planta de Zaragoza de Valeo Térmico. Para la aplicación se utilizó un CoBot comercial de Universal Robotics, este fue programado con la interfaz gráfica de usuario Polyscope. El autor definió las especificaciones de trabajo y las condiciones del entorno para construir una programación de forma convencional, incluso se concluyó que “se logró asignar diferentes tareas a los operarios y dejar las más difíciles o repetitivas para el robot” (Tabuenca, 2017), esto deja en evidencia que al utilizarse métodos convencionales de programación los CoBots pierden flexibilidad y deben ser relegados a tareas simples.

Por último, se analizó una patente, la cual se enfoca en el desarrollo de “Sistemas Robóticos colaborativos humano-máquina” (Patente nº US20130218340A1, 2011), este trabajo implementó un sistema robótico semiautomático e interactivo para realizar y/o simular sistemas una tarea de múltiples pasos (Patente nº US20130218340A1, 2011). Lo valioso de esta patente es la integración de diferentes sistemas, como sistemas de

control, sistemas de reconocimiento y sistemas de sensor-actuador. Esto demuestra la facilidad de los robots colaborativos para acoplarse a diferentes tareas basándose en la integración de diferentes sistemas.

1.3.2 Marco teórico

Este marco teórico brinda los conceptos básicos necesarios para el entendimiento del proyecto, dividiendo todo el sistema en hardware, software y esquema de comunicación, con el objetivo de desglosar los conceptos en forma más clara.

- **Hardware**

Robots colaborativos

Un robot colaborativo, también llamado CoBot, es un dispositivo que manipula objetos en colaboración con un operario. Este tipo de robots proporciona asistencia por medio del establecimiento de superficies virtuales que se pueden utilizar para restringir y guiar el movimiento. Los robots colaborativos son potencialmente adecuados para tareas en donde la seguridad es crítica o en aquellas que implican grandes fuerzas de interacción (Peshkin, Wannasuphoprasit, & Colgate, 1996).

Los CoBots permiten trabajar en armonía con el operario y ayudar en las líneas de producción, en cualquier lugar de estas, aportando el trabajo pesando, la resistencia y la precisión en las tareas, mientras que el operario se encarga de aportar la destreza, flexibilidad y la capacidad de resolver problemas (El Financiero, 2018).

La implementación de los robots colaborativos se puede dar en cualquier tipo de industria, desde las grandes empresas productoras hasta las pequeñas Pymes, esto debido a que las compañías están migrando de la automatización en gran volumen a una producción flexible que permita satisfacer las demandas de los consumidores que cambian constantemente y desean competir en un mercado global (Rethink robotics, s.f.).

Programación de robots colaborativos

La forma convencional de programar los CoBots es utilizando un software desarrollado por cada fabricante. En el caso del robot colaborativo de la universidad EIA, el software que permite controlar el sistema se llama Intera 5. Esta es la interfaz gráfica de usuario (IGU) y fue desarrollado por la empresa Rethink Robotics. Este software proporciona un enfoque de programación funcional y gráfico que garantiza confiabilidad en los desarrollos y en la implementación (Rethink robotics, s.f.).

- **Software**

El segundo componente general para describir el funcionamiento del sistema de adquisición de datos es el software, aquí se presentan los datos al usuario de la mejor

manera con el fin de que él pueda realizar un análisis de esta información y tomar las decisiones más acertadas.

Automatización robótica de proceso (RPA)

Es una tecnología que automatiza procesos rápidamente implementado Bots que imitan las acciones de los empleados en un ordenador. Es una gran alternativa para reducir o eliminar cargas de trabajo de procesos back – office tales como los financieros, de contabilidad y manejo de información, ayudando a las personas que realizan tareas repetitivas de gran volumen. Además, permite una integración sencilla con entornos IT de las empresas y con casi cualquier proceso existente, lo que permite su implementación en una amplia gama de la industria. Los procesos automatizados con RPA pueden ahorrar de un 25% a un 50% los costos necesarios para su realización y les proporciona continuidad a los procesos gracias a tener Bots habilitados para trabajar 24/7/365 (Institute for robotic process automation & artificial intelligence, 2015).

Dentro de la amplia gama de funciones que pueden adquirir los Bots se encuentran la administración de correos electrónicos, el diligenciamiento de formularios y formatos, la generación de cualquier tipo de documento, y el control de cualquier programa que se tenga instalado dentro de una computadora. Uno de los valores agregados que tiene la tecnología RPA es que sus grandes proveedores se están asociando con grandes compañías de tecnología cognitiva como IBM y Google, y están incorporando en sus ofertas tecnología como el procesamiento del lenguaje natural y el aprendizaje automático (David Schatsky, 2016).

IBM Watson

Es un sistema informático desarrollado por la empresa estadounidense IBM, esta tecnología responde a preguntas y solicitudes basándose en inteligencia artificial, además, presenta diferentes módulos que permiten realizar desarrollos con tecnología de punta, entre estos se encuentra los módulos de procesamiento de lenguaje natural, traducción de texto, convertidor de texto a voz y viceversa, entre otros. Estos servicios poseen costos elevados, sin embargo, la herramienta presenta una versión gratis, en donde los servicios son limitados en recursos y tiempos (IBM, s.f.).

- **Esquema de comunicación**

Módulo de comunicación OSI

El modelo de interconexión de sistemas abierto (OSI por sus siglas en inglés) es una herramienta de referencia que describe el proceso por el cual deben pasar los datos para ser comunicados entre dos sistemas en red. El modelo de comunicación OSI se divide en 7 capas, cada una cumple una función específica y soporta las capas superiores y ofrece servicio a las inferiores (Global Knowledge, 2006). A continuación, se describirán cada una de las siete capas.

Capa física.
Se encarga de definir las especificaciones del conector y del cable de conexión, así como el medio requerido. Además, comprende especificaciones eléctricas, mecánicas, funcionales y de procedimiento para enviar un flujo de bits en una red informática. Los componentes de la capa física incluyen a) componentes del sistema de cableado, b) adaptadores que conectan medios a interfaces físicas, c) diseño del conector y asignaciones de pines, d) especificaciones del concentrador, repetidor y panel de conexiones, e) componentes del sistema inalámbrico, f) interfaz del sistema de computadora pequeña e i) tarjeta de interfaz de red (NIC) (Global Knowledge, 2006).

Capa de enlace de datos.
La capa 2 proporciona las siguientes funciones: Permite que un dispositivo acceda a la red para enviar mensajes, ofrece una dirección para que los datos de un dispositivo puedan enviarse a la red, funciona con el software de red de un dispositivo al enviar y recibir mensajes y brinda la capacidad de detección de errores (Global Knowledge, 2006).

Capa de red.
El objetivo de la capa de red es transportar los datos desde la fuente hasta el destino sin importar que ambos no estén conectados directamente. Se encarga de identificar el enrutamiento existente entre una o más redes, las unidades de datos se denominan paquetes y se pueden clasificar en protocolos enrutables y protocolos de enrutamiento (Global Knowledge, 2006).

Capa de transporte.
Ofrece comunicación en extremo a extremo entre los dispositivos finales de una red. Algunas de las funciones de la capa de red son la identificación de la aplicación y de la entidad del lado del cliente, confirmación de que el mensaje llegué completo, segmentación de datos para el transporte de red, control de flujo de datos para evitar el exceso de memoria, entre otros (Global Knowledge, 2006).

Capa de sesión.
Esta capa de sesión permite generar aplicaciones en dispositivos para establecer, administrar y terminar un diálogo a través de una red. A demás, incluye funcionalidad de sincronizar el flujo de datos, creación de unidades de dialogo, negociación de parámetros de conexión, participación de servicios en funciones, entre otros (Global Knowledge, 2006).

Capa de presentación.
La capa 6 básicamente permite que una aplicación lea o comprenda el mensaje, algunas de las funciones de esta capa incluyen el cifrado y descifrado de mensajes para seguridad, comprensión y expansión de un mensaje para que viaje de manera eficiente, formateo de gráficos, traducción de contenido y especificación del sistema (Global Knowledge, 2006).

Capa de aplicación.
La capa final proporciona una interfaz para el usuario final, que opera un dispositivo conectado en red, además es el soporte para transferencias de archivos, da la posibilidad de imprimir en una red, tiene acceso a correo electrónico y menajes eléctricos y da la posibilidad de navegar por la World Wide Web (Global Knowledge, 2006).

Módulo de comunicación TCP/IP

El modelo TCP/IP usa cuatro capas para realizar las mismas funciones que el modelo OSI realiza en siete capas. La capa de acceso a la red funciona de manera igual a una combinación de la capa física y de la capa de enlace de datos. La capa de internet del modelo TCP/IP realiza las mismas funciones de la capa de red del modelo OSI. La capa de transporte y de sesión de OSI son sustituidas en el modelo TCP/IP por la capa host-to-host. En la capa final, si el usuario está usando TCP, la capa de proceso posee las mismas funciones que la capa 6 y 7 de OSI; cuando el usuario está usando un protocolo de transporte UDP, la capa de proceso posee las mismas funciones que las capas 5, 6 y 7 del modelo OSI (Global Knowledge, 2006).

2. METODOLOGÍA

El diseño metodológico para el desarrollo de un producto se puede dividir en cinco componentes que dan como resultado el dispositivo terminado (Ulrich & Eppinger, 2008).

2.1 IDENTIFICACIÓN DE LOS REQUERIMIENTOS

Antes de realizar la construcción del prototipo se debe hacer una recolección de datos que permita identificar las tareas más comunes que realizan los robots colaborativos en las industrias, esto con el fin de utilizar las cuatro tareas más relevantes como base para el trabajo de este proyecto. Además, estos datos brindarán información acerca de las condiciones de trabajo en las que operan los robots colaborativos, lo que ayudará en el diseño del aplicativo web.

Para realizar esta recolección de datos se usará el método de vigilancia tecnológica, esta no se enfoca solamente en la recolección de información sobre un problema a resolver, sino que sistematiza su obtención con respecto al estado actual y la ejecución del desarrollo científico-tecnológico, dando valor agregado a los datos recolectados y generando reportes que orientan la toma de decisiones (Universidad de Chile, 2012). Previo a la búsqueda de información se definirán criterios de inclusión y de exclusión, además de procesos de selección de estudios primarios y posteriormente se organizará por orden de importancia y se realizará un respectivo análisis y síntesis.

2.2 DISEÑO DE CONCEPTO

Para diseñar las funciones específicas de cada componente se debe definir primero la función general de la plataforma, para lograr esto se construirá una caja negra, en la que se tienen en cuenta las entradas y las salidas del sistema, sin considerar el proceso interno. Posteriormente se construirá una caja trasparente en donde se definirán cada una de las funciones internas de cada componente y las conexiones entre ellas. Después de estos procedimientos fundamentales y tener claridad de las funciones internas y externas se seleccionarán los métodos y componentes de cada función utilizando una matriz morfológica. En esta se prueban diferentes combinaciones entre las posibilidades que posee cada función para ser realizada y se evalúan variables significativas, descritas en la identificación de requerimientos, para seleccionar la mejor.

2.3 DISEÑO DE DETALLE

Después de tener claridad sobre las funciones que desempeñará cada componente del dispositivo y los materiales o aplicaciones que se usarán, se procederá al desarrollo.

Se desglosarán las tareas más comunes de los robots colaborativos, seleccionadas previamente, en subprogramas o subrutinas de programación, esto con el objetivo de crear una biblioteca en donde el Bot, creado con RPA, tenga fácil acceso a ellas y utilizándolas pueda construir la programación que cumpla con la tarea que el usuario está solicitando. Realizar este paso de una manera óptima garantizará flexibilidad en la combinación de tareas y le permitirá al Bot realizar programaciones mucho más complejas. Posteriormente, se programarán estos subprogramas en el robot colaborativo de la Universidad EIA, esta cuenta con un CoBot Baxter diseñado por *Rethink Robotics*.

Además de los subprogramas previamente estructurados, se programarán secuencias muy comunes de los robots colaborativos como lo son el reconocimiento visual, utilización de los actuadores y mecanismos de seguridad.

El aplicativo web se diseñará de tal forma que el usuario pueda ingresar sus instrucciones por medio de lenguaje natural, escribiendo la tarea que desea que el robot realice. Además, este aplicativo web tendrá la capacidad de ingresar a un API, capa siete del modelo de comunicación OSI, la cual será el lazo de comunicación entre el aplicativo web y el Bot. El objetivo del Bot es automatizar la tarea repetitiva de reprogramar el CoBot cada vez que se desee que cambie de actividad, este se enfocará en acceder a la API, extraer la información del usuario, interpretar las instrucciones, estructurar el código y programar el CoBot para que realice la tarea.

2.4 INTEGRACIÓN DE LOS DESARROLLOS

Teniendo el funcionamiento de cada componente por separado se realizarán pruebas de funcionamiento en cada etapa, posteriormente se construirá el prototipo, en donde se unirán los tres componentes y realizarán pruebas para verificar el funcionamiento del sistema acoplado y de sus conexiones.

2.5 PRUEBAS Y REFINAMIENTO

Después de las pruebas de funcionamiento del prototipo y de sus respectivas correcciones, se comprobará su funcionamiento en un entorno académico controlado, en donde el dispositivo será programado por diferentes personas y se evaluará su eficiencia para realizar la tarea, comparándola con la pedida por el usuario. Por último, se llevará a

cabo una encuesta en la que participen los usuarios para obtener una realimentación con respecto a la interacción con el aplicativo.

3. PRESENTACIÓN Y DISCUSIÓN DE RESULTADOS

3.1 IDENTIFICACIÓN DE REQUERIMIENTOS

Con el objetivo de caracterizar las tareas más comunes que desempeñan los robots colaborativos dentro de la industria, se realizó una recopilación de información en diferentes fuentes de datos académicas, como lo son Science Direct y Scopus, además de usar las herramientas Google patentes y académico. En esta búsqueda se seleccionaron las tareas a ser caracterizadas, esto fue realizado bajo una vigilancia tecnológica, la que se encuentra en el anexo 1, en la que se tenían parámetros de búsqueda como el tipo de tareas que realizan los CoBots, industrias en las que más se utilizan, cómo es la programación y la comunicación del usuario con el CoBot, entre otros.

Se seleccionaron las siguientes tareas:

- **Pick & Place**

Esta tarea es una de las más utilizadas (Mabie, s.f.), debido a su flexibilidad en la industria. En este proceso el robot debe seleccionar un objeto, tomarlo y depositarlo o situarlo en otro lugar o dentro de otro componente.

- **Cambio de herramienta**

El cambio de herramienta o pieza en un proceso de fabricación utilizando CNC, es una tarea repetitiva que desgasta al empleado. Aquí el robot, teniendo en cuenta todos los procedimientos de seguridad necesarios, debe seleccionar la herramienta o pieza a cambiar y el lugar en donde debe hacerlo.

- **Pintar y lijado piezas.**

El proceso de pintar una pieza de forma industrial puede ser asistido por un robot colaborativo, debido a que representa una tarea repetitiva, en donde se tiene una rutina y una trayectoria de pintado o lijado determinada, además de que requiere de condiciones de seguridad altas debido a los químicos con los que se trabajan.

- **Empaquetado**

Otra tarea repetitiva que se da en muchas empresas de producción industrial que fabriquen cualquier producto, independiente del tamaño. El empaquetado de productos es uno de los procesos donde más se utilizan los robots colaborativos (Universal Robots, 2017).

Luego de identificar las cuatro tareas más comunes, se procede a realizar una caracterización de cada una, es decir, dividir ese proceso en subtareas más pequeñas, esto con el objetivo de poder generar cualquier proceso, diferente a las tareas seleccionadas, con la combinación de esas subtareas. A continuación, se muestran las subtareas seleccionadas, enfocadas a las acciones que debe realizar el robot colaborativo.

- **Pick and place:**
    a. Capturar y procesar imágenes.
    b. Mover brazos a posiciones deseadas.
    c. Posicionar los brazos en lugares estándar (Homing).
    d. Tomar un objetivo con el actuador.
    e. Soltar un objeto con el actuador.
    f. Calcular trayectorias de los brazos.
- **Pinta y lijar piezas:**
    g. Reconocer área de un objeto plano.
    h. Calcular trayectorias para pintar o lijar.
    i. Posicionar los brazos en lugares estándar (Homing).
    j. Capturar y procesar imágenes.
    k. Mover brazos a posiciones deseadas.
- **Cambio de piezas en la CNC**
    l. Capturar y procesar de imágenes
    m. Mover brazos a posiciones deseadas
    n. Posicionar los brazos en lugares en lugares estándar (Homing)
    o. Tomar un objetivo con el actuador.
    p. Soltar un objeto con el actuador.
    q. Calcular trayectorias de los brazos.

- **Empaquetado.**

    r. Capturar y procesar de imágenes.

    s. Mover brazos a posiciones deseadas.

    t. Posicionar los brazos en lugares en lugares estándar (Homing).

    u. Tomar un objetivo con el actuador.

    v. Soltar un objeto con el actuador.

    w. Calcular trayectorias de los brazos.

    x. Trasladar objetos a posiciones deseadas.

Para evidenciar la correlación entra las tareas seleccionadas con base en las subtareas identificadas, este diagrama se muestra en la figura 1.

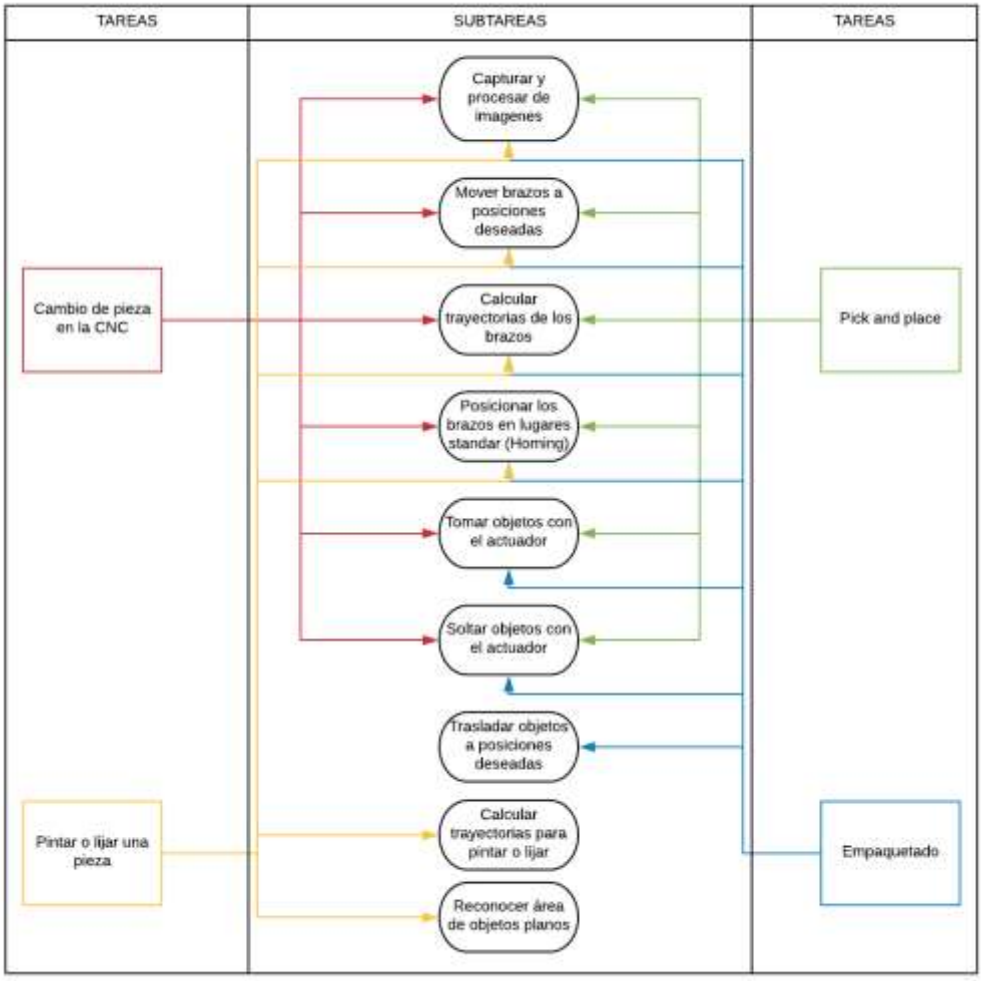

Figura 1 Diagrama de correlación de tareas

El diagrama muestra que las tareas seleccionadas se basan en la identificación de objetos por medio de las cámaras que posee el robot colaborativo y generación de las diferentes trayectorias.

Como parte del diseño preliminar del proyecto se realizó un matriz de necesidades enfocada en clasificar las necesidades identificadas en la vigilancia tecnológica y la documentación recolectada para realizar el planteamiento del problema y la justificación del proyecto. En esta matriz se pondera cada necesidad por nivel de importancia, se le asignan características generales y se da un estándar de medida.

Tabla 1 Matriz de necesidades

#	Necesidad	Importancia	Característica	Medida
1	Los usuarios puedan ingresar la tarea por medio de lenguaje natural	5	Facilidad de programación	Subjetivo
2	La plataforma pueda reprogramar los CoBots.	5	Facilidad de programación	Binario
3	Flexibilidad de los CoBots en la industria.	5	Rapidez para reasignar tareas en entornos industriales	Subjetivo
4	Reducir costos de mano de obra calificada	4	Costos de reprogramación	COP
5	Capacidad para programar el CoBot de forma remota	3	Facilidad de programación	Binario
6	La plataforma de programación tenga un interfaz de usuario amigable.	4	Facilidad de programación	Subjetivo
7	La plataforma pueda programar cualquier tarea o combinación de ella que se encuentre entre las seleccionadas	5	Versatilidad de la plataforma	Lista
8	La instalación de la plataforma sea simple	3	Complejidad en la instalación	Subjetivo

Como resultado de la matriz de necesidades se concluyó que la simpleza para ingresar la tarea al CoBot, es una de las más importantes, debido a que la correlación entre esta necesidad y la flexibilidad del CoBot en la industria, es directamente proporcional, generando así un impacto en dos de las necesidades con más ponderación.

3.2 DISEÑO DE CONCEPTO

Luego de tener claridad sobre las diferentes necesidades que pretende desarrollar el proyecto y de haber caracterizado las tareas más comunes en la industria, se procedió a realizar una caja negra, este método permite visualizar el sistema a desarrollar de una manera sencilla, contemplando únicamente entradas y salidas del sistema.

Figura 2 Caja negra del sistema

La figura 2 muestra las entradas que tendrá el sistema, energía eléctrica y la orden del usuario en lenguaje natural, y como salida, después del procesamiento de toda la información, la orden o tarea ejecutada en el CoBot. Esta herramienta permite tener una vista a alto nivel del proyecto y sirve como insumo fundamental en la construcción de la caja transparente, un diagrama que contempla las mismas entradas y salidas de la caja negra, pero que describe y esquematiza el procesamiento interno en subsistemas o subfunciones.

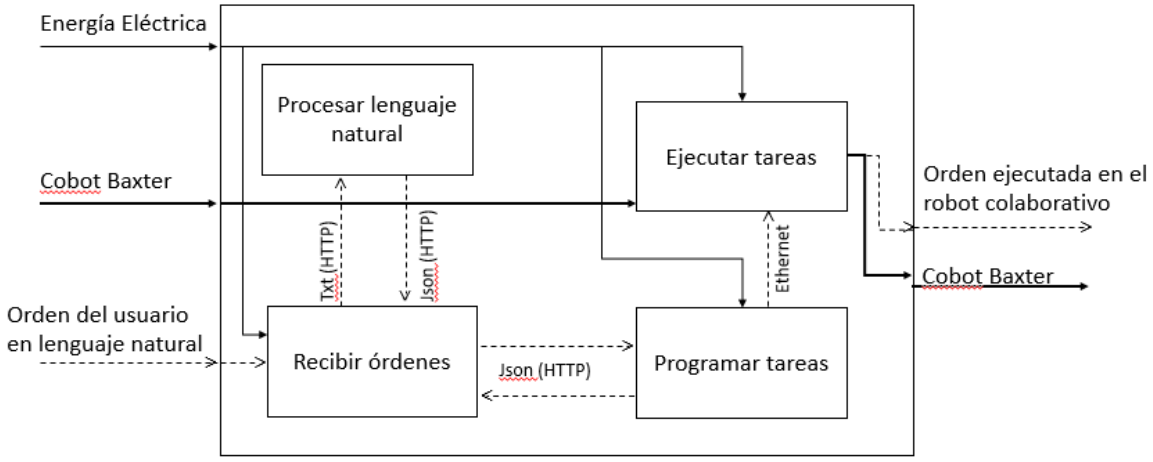

Figura 3 Caja transparente del sistema

En la caja transparente se muestran cuatro subsistemas, el primero es el módulo encargado de recibir las ordenes de usuario, este debe ser de forma gráfica y que permita un entendimiento intuitivo para el usuario. El segundo es el módulo de procesamiento de lenguaje natural, este debe recibir el mensaje que ingresa el usuario y desglosarlo en

diferentes componentes entendibles para el Bot y que le permitan identificar qué combinación de subtareas permiten cumplir con la tarea solicitada por el usuario. El tercero es el módulo encargado de programar el CoBot, este será realizado con un software de RPA, recibirá la información del usuario procesada y estructurará la programación en el software del CoBot. Por último, el cuarto módulo es el encargado de ejecutar las tareas, este está compuesto por un robot colaborativo.

3.2.1 Registro de soluciones

Con el fin de consolidar un punto de partida para los diferentes conceptos de diseño a desarrollar, se realiza una búsqueda de tecnologías que cumplan con las especificaciones y tareas que debe desempeñar cada subsistema de la caja transparente.

Recibir órdenes

Este subsistema debe contar con un entorno de desarrollo que permita crear una interfaz gráfica de usuario, amigable e intuitiva para el usuario, además, de poder generar una comunicación por medio de APIs entre el subsistema de procesamiento de lenguaje natural y el encargado de programar las tareas en el CoBot. Con base en estas características de ingeniería se seleccionó el software especificado en la tabla 2.

Tabla 2 Registro de soluciones – Recibir órdenes

Alternativa de solución	Definición	Ventajas	Desventajas
(Node JS, s.f.)	Node JS	Este entorno permite ejecutar aplicaciones de JavaScript orientado a eventos asíncronos. Permite generar interfaces de usuarios web y aplicaciones, además, permite administrar la conectividad y el consumo de recursos de diferentes aplicaciones por medio de APIs.	Node JS no posee librerías estándar como las tienen otros entornos de programación.
Alternativa de solución	**Definición**	**Ventajas**	**Desventajas**
(Node RED, s.f.)	Node RED	Esta aplicación está basada en Node Js y posee una interfaz de programación de gráfica, por medio de bloques, esto facilita la programación.	Posee una desventaja similar al node js, no cuenta con librerías de aplicaciones disponibles.

Alternativa de solución	Definición	Ventajas	Desventajas
 (HTML 5, s.f.)	HTML 5	La facilidad para adaptarse a diferentes pantallas y entorno, las aplicaciones web desarrolladas en HTML, es muy grande, además estás aplicaciones se pueden implementar como páginas web en servidores locales y también en navegadores (Intel software, 2014).	La eficiencia del código depende del motor de traducción debido a que los lenguajes no son nativos para las plataformas.

Procesamiento de lenguaje natural

El componente seleccionado en este subsistema debe poseer una capa de uso libre, debido a la naturaleza académica del proyecto, además, tener la capacidad de ingresar información por medio de texto en párrafos de gran tamaño, procesarlos y extraer información fundamental como intenciones, análisis sintáctico y clasificación de contenido, los softwares seleccionados y sus características se muestran en la tabla 3.

Tabla 3 Registro de soluciones – procesamiento de lenguaje natural

Alternativa de solución	Definición	Ventajas	Desventajas
 (IBM, s.f.)	IBM Watson:	Esta herramienta de IBM además de prestar servicios de procesamiento de lenguaje tiene sistemas de conversaciones basado en machine learning.	Es un servicio privado y para consumir este servicio se debe pagar. Sin embargo, posee una versión gratuita que brinda los mismos servicios, pero de forma limitada.
Alternativa de solución	**Definición**	**Ventajas**	**Desventajas**
 (Python, s.f.)	Python:	Python es un software Open Source que posee librerías especializadas en el procesamiento de lenguaje.	Genera resultados genéricos y a muy alto nivel. No permite tener un procesamiento tan completo.

Alternativa de solución	Definición	Ventajas	Desventajas
 (Google , s.f.)	Google cloud:	Posee servicios de almacenamiento en la nube, de procesamiento de lenguaje natural, análisis de datos, entre otros.	Es un servicio privado y para consumir este servicio se debe pagar. Sin embargo, posee una versión gratuita que brinda los mismos servicios, pero de forma limitada.

Ejecutar tareas

Como subsistema final, en el que se desplegarán las tareas solicitadas por los usuarios, se busca un robot colaborativo que posea cámaras y sensores que le permitan ubicarse de manera correcta en el espacio y reconocer diferentes objetos, además, que cuente con un entorno de programación sencillo que facilite la integración entre con el Bot. Con base en las necesidades del proyecto se seleccionaron las plataformas de robots colaborativos descritos en la tabla 4.

Tabla 4 Registro de soluciones – Ejecutar tareas

Alternativa de solución	Definición	Ventajas	Desventajas
 (Rethink robotics, s.f.)	Baxter:	Robot colaborativo con dos brazos que poseen 7 grados de libertad cada uno, además de cámaras en cada uno de los brazos y en la pantalla principal.	Alto costo de adquisición y de actuadores, además, posee gran tamaño, lo que le dificulta su ubicación.
Alternativa de solución	**Definición**	**Ventajas**	**Desventajas**
 (Universal Robots, s.f.)	UR10:	Este robot posee un alcance de 1300mm, una carga útil de 10kg, una huella de 190mm y un peso de 28, 9 kg. Es de dimensiones pequeñas y facilidad de instalación (Universal Robots, 2017).	Alto costo de adquisición y de actuadores.

Programar tareas

En este módulo se debe seleccionar un software de RPA con la capacidad de integrarse a diferentes aplicaciones por medio de APIs, además de ser capaz de realizar consultar a bases de datos y reconocer diferentes entidades que brindan información para estructurar un código con subtareas, para cumplir con las ordenes de los usuarios. El registro de soluciones de los softwares encargados de la creación de los Bots se muestra en la tabla 5.

Tabla 5 Registro de soluciones – Programar tareas

Alternativa de solución	Definición	Ventajas	Desventajas
 (Automation Anywhere, s.f.)	Automation Anywhere:	Permite integrarse con diferentes softwares y aplicaciones, realizar peticiones REST y SOAP y posee una arquitectura robusta que facilita la ejecución y programación de tareas de manera más controlada.	La estructura de programación es a alto nivel, sin embargo, la estructuración de comandos se dificulta. Además, el costo de adquisición es alto.
Alternativa de solución	**Definición**	**Ventajas**	**Desventajas**
 (UiPath, s.f.)	Uipath:	Permite programar en diagrama de flujo y crear Bots backoffice.	Capacidad limitada de integración y de ejecución.
Alternativa de solución	**Definición**	**Ventajas**	**Desventajas**
 (Python, s.f.)	python:	Python es un software Open Source que posee librerías para el manejo del teclado y el mouse del computador.	No tiene tantas funcionalidades como un software especializado.

3.2.2 Matriz morfológica

Para continuar con el diseño de concepto se construye una matriz con las tecnologías reportadas en cada registro de soluciones, para las diferentes funciones del dispositivo, y se trazan rutas que definen los componentes que conforman los conceptos preliminares.

A B C	Alternativas solución		
Recibir órdenes	node JS	Node-RED	HTML 5
Procesar leguaje natural	IBM Watson		
Programar tareas	AUTOMATION ANYWHERE	UiPath	
Ejecutar tareas	rethink robotics	UNIVERSAL ROBOTS	

Figura 4 Matriz morfológica

Descripción de conceptos

Todos los conceptos seleccionados incluyen los mismos componentes en los subsistemas de programar y ejecutar las tareas, debido a la disponibilidad de licencias y de componentes que brinda la universidad. El entorno de programación de RPA será Automation Anywhere, esto debido a que se posee una licencia Enterprise, lo que da acceso total a las funciones y aplicaciones del software. Esta herramienta permite ejecutar los Bot con base en la aparición de un evento o programarse para que sea desplegado en días y horas específicas. Además, posee gran facilidad para comunicarse con páginas web, por medio de peticiones REST, a bases de datos e integrarse con programas que se estén ejecutando dentro de un ordenador. Esto genera gran facilidad para programar el

Robot colaborativo, que en este caso será Baxter, el robot colaborativo de Rethink Robotics, esto debido a que la universidad EIA cuenta en sus laboratorios de mecatrónica con este insumo. La capacidad de trabajo de Baxter, sus grados le libertad en cada brazo, sus sensores y cámaras lo convierten en una herramienta robusta para la automatización de procesos de lotes pequeños.

Concepto A

El uso de la plataforma de programación Node Red permitirá crear una interfaz de usuario que permita que se ingrese la orden por parte del operario que está programando el Robot colaborativo. Por otra parte, Node-RED administrará el flujo de información hacia la plataforma cloud, que se encargará de almacenar y procesar la información de las órdenes del operario, y el flujo de información hacia el Bot. La gran ventaja de este software es su plataforma de programación por bloques, esto facilita en gran medida la configuración de instrucciones. Por último, la plataforma Cloud de Google ofrece servicios gratuitos de procesamiento de lenguaje natural enfocados al análisis de entidades, sentimiento, sintáctico, sentimiento de entidades y clasificación de contenido, esto si no se consumen más de 5000 unidades al mes (Google , s.f.).

Concepto B

La plataforma de Node-RED posee una librería que permite utilizar todas las herramientas de IBM cloud, específicamente las de Watson, siendo así uno de los conceptos más oportunos a seleccionar debido a la facilidad de integración. IBM Watson posee un módulo de procesamiento de lenguaje natural y permite crear *chatbots*, y posee una franja gratuita para desarrollos, siempre y cuando no se exceda de una cantidad especifica de solicitudes al API. Por último, Python posee una librería llamada pyAutoGUI que permite controlar el mouse y el teclado.

Concepto C

Para la administración el flujo de información entre los usuarios de la plataforma, los servicios de procesamiento de lenguaje natural y Cloud, y el Bot. La interfaz creada en Node JS recibirá las ordenes de los operarios y consumirá los recursos de procesamiento de lenguaje natural para que el Bot de RPA pueda realizar peticiones REST a la página y obtener la información necesaria para programar la tarea en el robot colaborativo. La plataforma Cloud que se utilizará es IBM Cloud Foundry, esta plataforma posee el producto IBM Watson, el cual se encargará del procesamiento.

3.3 SELECCIÓN DEL CONCEPTO

Con el objetivo de seleccionar el concepto que permita desarrollar el proyecto de la mejor manera y obtener los mejores resultados, se utiliza una matriz de decisión, en la que se especifican diferentes criterios de decisión y su peso, se le da una calificación a cada

concepto sobre el criterio y se saca un ponderado, al final, el criterio con mayor puntaje en la suma de los ponderados será el elegido para desarrollar el proyecto.

Los pesos de los criterios de decisión están distribuidos de tal manera que sobresalga la capacidad de integración entre componentes, debido a que el objetivo del proyecto es integrar tecnologías que pueden considerarse nuevas y generar una única solución, y finalmente los costos. Siguiendo la línea de jerarquía de los criterios se tiene la complejidad de desarrollos, y por último la disponibilidad, este presenta un componente diferenciador ya que se tiene acceso al software de RPA y el robot colaborativo.

Tabla 6 Matriz de decisión

		Concepto A		**Concepto B**		**Concepto C**	
Criterio de decisión	**Peso**	**C.**	**P.**	**C.**	**P.**	**C.**	**P.**
Disponibilidad tecnológica	10%	5	0,5	5	0,5	3	0,3
Accesibilidad a documentación	10%	3	0,3	4	0,4	4	0,4
Capacidad de integración entre componentes	30%	4	1,2	5	1,5	5	1,5
Costos	30%	2	0,6	2	0,6	2	0,6
Complejidad de los desarrollos	20%	4	0,8	3	0,6	2	0,4
Total		3,4		3,6		3,2	
Posición		2		1		3	
¿Desarrollar?		NO		**SI**		NO	

Se evidencia que el concepto B es seleccionado debido que cumple, de una mejor manera, con las necesidades del proyecto y, por tanto, será el concepto por desarrollar. Esto debido a la facilidad de integración entre componentes, una de las dificultades o grandes retos que posee el desarrollo de este proyecto. Además, posee una accesibilidad grande a documentación, lo que permite avanzar de forma óptima en el cronograma del proyecto.

3.4 DISEÑO DE DETALLE

Luego de la selección del concepto, se procede a la construcción de cada uno de los componentes y su adecuada integración, teniendo presente la solución y el cumplimiento satisfactorio de los requerimientos de usuario identificados en el diseño de concepto. El orden de desarrollo está estructurado por los objetivos específicos, de donde salió el cronograma de actividades.

3.4.1 Diseño y creación interfaz gráfica

Un pilar fundamental en el funcionamiento de la plataforma de programación es la interfaz gráfica, este será el primer acercamiento de los usuarios con el dispositivo y debe ser sencilla de entender y que se pueda navegar por ella de manera intuitiva, además, que permita programar tareas en robots colaborativos sin necesidad de ningún conocimiento en programación ni en robótica.

Para el desarrollo de esta interfaz se seleccionó la herramienta Node-RED, esta plataforma, basada en NodeJS y JavaScript, permite la creación de plataformas web de manera sencilla, esto debido a que posee una librería enfocada al desarrollo de *dashboard*, lo que permite crear botones, cuadros de mensajes, franjas de textos, entre otros.

Para la interfaz de este trabajo se crearon dos ventanas, la primera está enfocada en generar un espacio para la autenticación de los usuarios. Esto garantiza seguridad y control de las personas que pueden ingresar e interactuar con los robots colaborativos de la empresa, esta pestaña se muestra en la figura 5.

Figura 5 Ventana de autenticación de usuarios

Esta ventana posee una función de autenticación, programada en JavaScript, que principalmente verifica el usuario y la contraseña en una lista guardada dentro del programa. La ubicación dentro del flujo de este nodo se muestra en la figura 6. Al realizar un registro exitoso se pasa automáticamente a la próxima ventana, sino envía un mensaje de error al usuario.

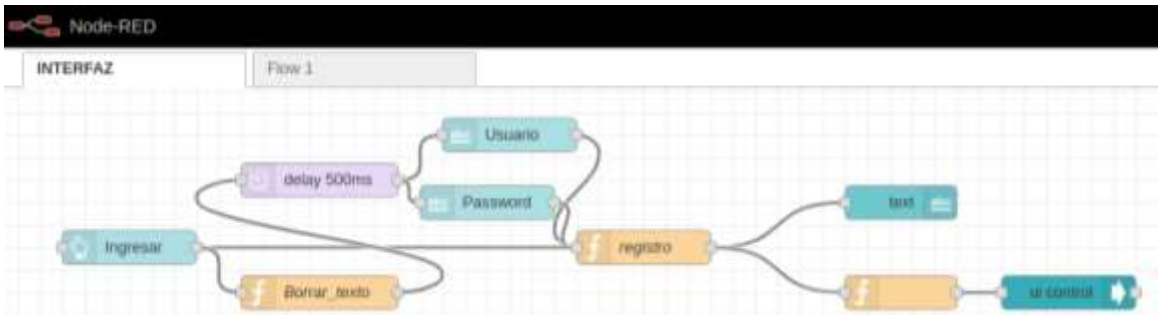

Figura 6 Programación de la ventana de autenticación

La segunda ventana se compone de cuatro módulos, mostrados en la figura 7, estos componen todo el dashboard o la interfaz completa con la que interactúa el usuario. En el primero llamado “Ingrese tarea” es donde se encuentra el asistente virtual, en este caso llamado *Baxter Assistant,* el usuario puede ingresar la orden que se desea programar. Esta se recibe en lenguaje natural, es decir, sin ningún comando de programación específico o ni ninguna estructura, se debe escribir el comando y dar click en el botón “Enviar” para interactuar con el agente. Baxter Assistant ayuda al usuario a reunir todos los datos que se necesitan para poder realizar un programa en el robot. Para facilitar este proceso se incluyeron imágenes dentro del dialogo, que funcionan como apoyo visual para que el usuario se sienta más cómodo al momento de programar y proveen información relacionada con el nombre de las articulaciones y el sentido de la orientación de cada una.

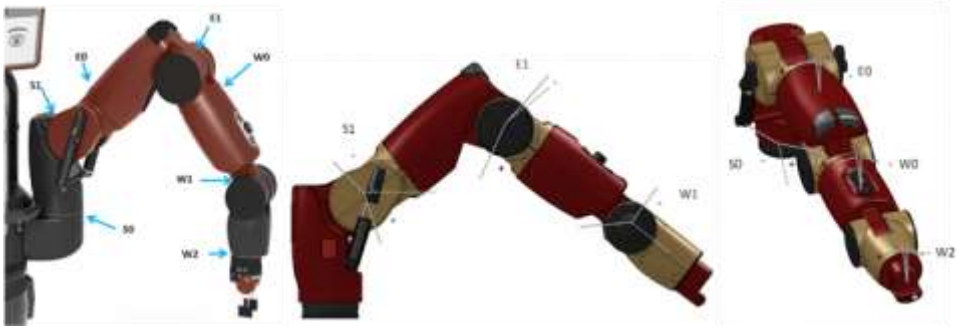

Figura 7 Imágenes de guía

La segunda, llamada “configuraciones” permite que los usuarios por medio de botones puedan realizar configuraciones como iniciar, apagar, llevar a posición de *home* al robot y guardar las programaciones que se están realizando, este último proceso se lleva a cabo debido a que cada vez que se termina de ejecutar una acción en el CoBot, el asistente virtual guarda la orden y le asigna un número, al momento de oprimir el botón de guardar, se seleccionan la cantidad de ordenes enviadas hasta ese momento y se pide al usuario que ingrese el nombre del programa para terminar el proceso de guardar. El último componente de este segundo módulo es un menú desplegable que permite seleccionar

los programas guardados por el usuario, se actualiza después de cada orden guardada, cuando se oprime el nombre del programa se ejecuta en el robot.

El tercer y cuarto módulo de la interfaz se creó con el objetivo de que se pueda tener información del robot en tiempo real, en estas dos secciones se puede ver los valores, en ángulos, de cada articulación de robot, tanto para el brazo derecho como para el izquierdo, estos valores se actualizan cada vez que hay comunicación entre el servidor web de Python y la interfaz en node-red. Estas secciones se crearon como valor agregado al desarrollo planteado inicialmente debido a que permite tener un monitoreo más sencillo del estado del CoBot.

Figura 8 Ventana de programación

Para la comunicación entre al interfaz y Baxter Assistant se utilizó la librería de IBM Watson en Node-red, este paquete posee diferentes nodos que permiten acceder a las herramientas que provee el servicio cloud de IBM. En este caso se utilizó el nodo *Assistant,* este recibe como entrada un valor tipo texto y retorna los datos en formato JSON, en este formato se encuentran todas las variables que extrajo el asistente de la conversación con el usuario. Sin embargo, se debe realizar un procesamiento posterior antes de enviarlo al servidor web. La conexión del nodo se muestra en la figura 9.

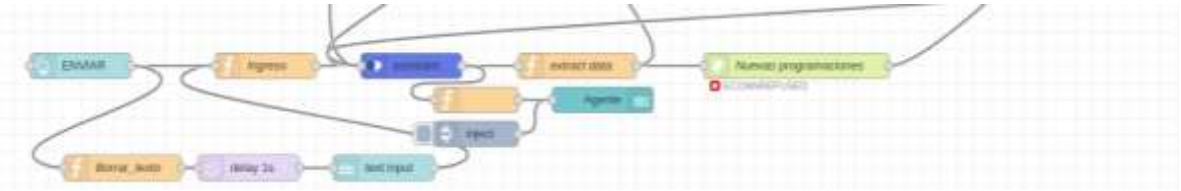

Figura 9 Programación del procesador de lenguaje

Por último, para enviar la información al servidor web, donde se alojan el bot de RPA y los programas requeridos para programar el Baxter, se utilizó un nodo de comunicación HTTP, este permite hacer peticiones POST al servidor con la información requerida para la programación, esta está construida en formato JSON.

3.4.2 Diseño agente virtual

El asistente virtual cumple un papel fundamental dentro del desarrollo del proyecto, ya que permite que los usuarios interactuaren de manera simple con la plataforma, esto afecta directamente uno de los ítems de la lista de métricas del proyecto. Para crear este asistente virtual, se deben tener en cuenta dos variables, la primera son las intenciones, estás son objetos o propósitos que el usuario implementa para expresar su necesidad o consulta al chatbot (IBM, 2019). La segunda variable a tener encuenta son las entidades, estas representan la información complementaria para que el usuario cumpla con su interacción (IBM, 2019). Estas ayudan a que Baxter Assistant pueda entender el lenguaje natural proveniente del usuario y a guiarlo en que ingrese los datos necesarios para realizar la programación. La tabla 7 muestra las intenciones y la tabla 8 las entidades utilizadas para este proyecto, para realizar un buen entrenamiento del agente se deben agregar ejemplos, los cuales también se incluyen en la tabla mencionada. Para el desarrollo del proyecto se usaron 3 ejemplos en promedio por intenciones.

Tabla 7 Intenciones del agente virtual

Valor	Ejemplos
Borrar	Borrar comandos actuales, borrar programación.
Calcular	Calcular área del cuadrado verde, calcular el tamaño del cubo rojo.
Capturar	captura una foto con la cámara del brazo derecho, tomar una foto con la cámara principal, toma un foto con la cámara de la cabeza.
Despedir	Chao, bye, hasta luego, nos vemos pronto.
Guardar	Guardar datos, guardar estado del robot, guardar programación.
Mover	Move, mover brazo derecho, por favor mover el brazo izquierdo.
Programar	Deseo programar el baxter, programar el baxter, quiero programa.
Saludar	Buenos días, buenas noches, buenas tardes, hola.
Soltar	Liberar el objeto que tienes en el brazo derecho, soltar lo que tengas en la mano derecha.
Tomar	Tomar el cubo rojo, toma el objeto con la mano derecha.

Tabla 8 Entidades del agente virtual

Valor	Ejemplos
Acción_garra	Cerrar,Abrir
Articulaciones	E0, E1, S0, S1, W0, W1, W2.
Cámaras	Cámara principal, cámara
Confirmación	Sí, no
Dirección	Derecha, izquierda
Partes	Brazo, cabeza, Gripper

Por último, para que Baxter Assistant se pueda comunicar con el usuario, se debe crear un dialogo, aquí se establecen las respuestas del agente virtual y la lógica necesaria para que se cumpla la orden que el usuario desea. Este dialogo se crea por medio de nodos estructurados de manera jerarquica, tal y como se ve en la figura 10, en los cuales el usuario navega en cada interacción con el agente virtual. En el desarrollo de este trabajo el dialogo se enfocó en atender dos solicitudes del usuario.

La primera es generar un movimiento de las partes del CoBot, como brazo, cabeza o actuador, para esto el dialogo solicita que siempre se ingresen los datos necesarios, en el caso del movimiento de los brazo el usuario debe ingresar el brazo a mover, la articulación deseada, la cantidad de grados y la orientación. Para mover la cabeza el asistente virtual pide que ingresen el sentido en que la quiere mover y los grados, por último, para mover el actuador, que en este caso es un garra, se pregunta si se quiere abrir o no.

Figura 10 Dialogo del asistente virtual

Depués de cada orden, el asistente virtual envía un mensaje de confirmación al usuario para verificar que la información recolectada es correcta, luego de eso envía la información a la interfaz para que posteriormente sea enviada al servidor web y al bot de RPA. Después de cada orden el asistente virtual pregunta si se desea hacerlo otra vez. Como valor agregado, al momento de que una persona ya sepa qué variables debe ingresar lo puede hacer en un solo mensaje y una persona que no tenga estos conocimientos será guiada por el asistente.

Por último, para garantizar la comunicación entre el agente y la interfaz se configura dentro del nodo Assistant las credenciales del API, eso permite que se pueda interactuar con Baxter Assistant desde la interfaz creada en Node-red. Las credenciales se muestran a continuación.

- **la clave del API:** *"GxiLDB991RqZByf1AY7E8bUCxEgjrpSje1lHgI707cHl"*
- **Endpoint o URL:** *https://gateway.watsonplatform.net/assistant/api*.

Este desarrollo potencia la aplicación del trabajo de grado, debido a que amplía la capacidad cognitiva de la herramienta y eso genera que cualquier persona, tenga conocimientos en programación o no, pueda programar el CoBot y realizar tareas como tomar objetos, posicionar el robot en un punto espacial (x,y,z) especifico, llevar a un ángulo cualquiera de las articulaciones y la cabeza.

3.4.3 Creación y conexión del Bot RPA

En la arquitectura del proyecto, descrita en la caja trasparente (figura 3), se observa que el bloque de recibir las ordenes, que es la interfaz creada en Node-red y el bloque de programar tareas, que es un ordenador, están conectados mediante un servidor web, el cual recibe peticiones HTTP. El servidor se aloja en el computador que está conectado vía ethernet al CoBot y está creado en el lenguaje de programación Python y apoyado en una librería llamada FLASK, esta librería permite crear diferentes rutas o *Endpoints* en los cuales se pueden configurar peticiones de cualquier tipo. Para el proyecto se utilizaron dos rutas tipo POST, una encargada de recibir las ordenes enviadas desde los botones ubicados en el módulo de "configuraciones" llamada "programaciones guardadas" y otra encargada de recibir la información del asistente virtual, llamada "nuevas programaciones".

Al momento de realizarse una petición desde la interfaz se ejecutan los comandos necesarios para efectuar la programación pedida por el usuario. En el caso de la ruta de "programaciones guardadas", al invocarse esta función se ejecuta un Bot de RPA. Está encargado de abrir una terminal o consola y ejecutar dos comandos de configuración necesarios para establecer la comunicación con el CoBot. Este desarrollo garantiza una automatización total del proceso y garantiza que llegue siempre la orden desde el usuario al robot.

Para esta aplicación, el Bot de RPA está hecho en Python, lo cual genera dos beneficios para el proyecto, el primero es que tanto el servidor web como el Bot de RPA están en el mismo lenguaje y esto genera menos integraciones dentro de la solución, lo segundo y lo más importante, es que su automatización es compatible con el sistema operativo Ubuntu 14.04, este es el único sistema operativo con el que se comunica el CoBot con cualquier ordenador.

En el caso de la ruta de “programaciones nuevas”, primero se deben guardar los datos provenientes de la interfaz, para esto se utiliza un archivo en formato .txt, este cumple las funciones de almacenamiento temporal de los datos, luego se ejecutan las programaciones realizadas, dependiendo si se quieren mover los brazos, tomar un objeto, guardar las rutinas programadas o ejecutar las programaciones guardadas.

Para la ejecución de los programas nuevos se crearon tres *scripts*, estos atienden a las necesidades planteadas en el proyecto como lo son tomar objetos y realizar movimientos con cualquier parte del robot. Además, como valor agregado se creó un programa que guarda las posiciones del robot, es decir, mientras se van ejecutando órdenes y se van realizando movimientos, el asistente virtual Baxter Assistant cuenta la cantidad de veces que se configura el robot, esto para que el programa guarde esas posiciones y los concatena con los anteriores, al final, cuando el usuario oprima el botón “guardar” se guarda un archivo con todas las órdenes dadas por el usuario hasta ese punto.

En los anexos del proyecto se encuentran los programas mencionados en esta sección, el anexo 2 se refiere al programa donde se encuentran las rutas del servidor web, este se llama “Servidor”, el anexo 3, anexo 4 y anexo 5 corresponden a los programas de tomar objetos, mover partes del robot y ejecutar programas guardados, respectivamente.

3.4.4 Verificación de resultados

Para realizar la verificación del proyecto se realizó una prueba a veinte personas en la universidad EIA, a estas se les entregó un reto, llevar al CoBot a una posición deseada, mostrada en la figura 11. Las personas fueron seleccionadas con el objetivo de tener tres muestras diferentes, personas que alguna vez hayan programado el CoBot, otras que sepan programar pero que nunca hayan programado al CoBot y, por último, personas que nunca hayan programado.

Figura 11 Posición de testeo

Luego de que las personas terminaran con el reto, se les pidió diligenciar una encuesta en la que se mediría, en una escala de 1 a 5, la facilidad que representó para ellos haber programado el CoBot, además, se preguntó acerca de la utilidad del agente virtual (Baxter Assistant) dentro del proceso de programación. Las respuestas obtenidas se muestran en la figura 12, la cual recopila las cinco preguntas realizadas a las personas.

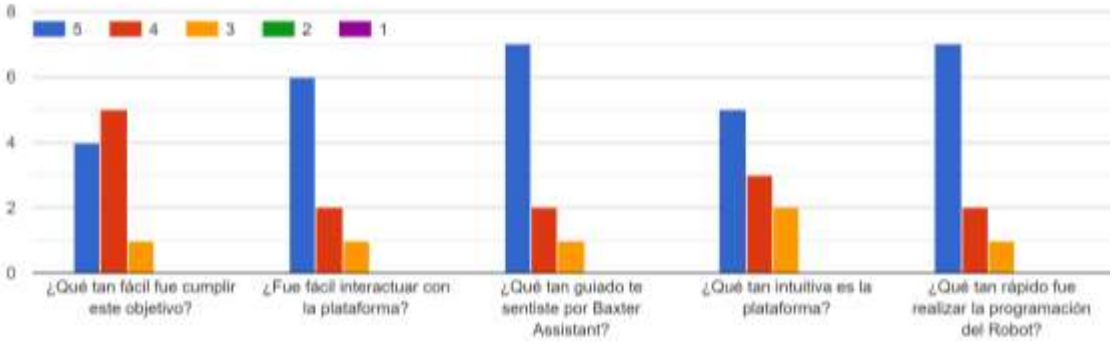

Figura 12 Resultados prueba de funcionamiento

Por otra parte, al final de la encuesta se les preguntó a qué público objetivo considerarían que impactaría el desarrollo de este trabajo de grado, las opciones se muestran en la figura 13. Esta pregunta se realizó con el objetivo de que los usuarios al interactuar con el desarrollo entendieran el alcance que posee y las aplicaciones que se le pueden dar.

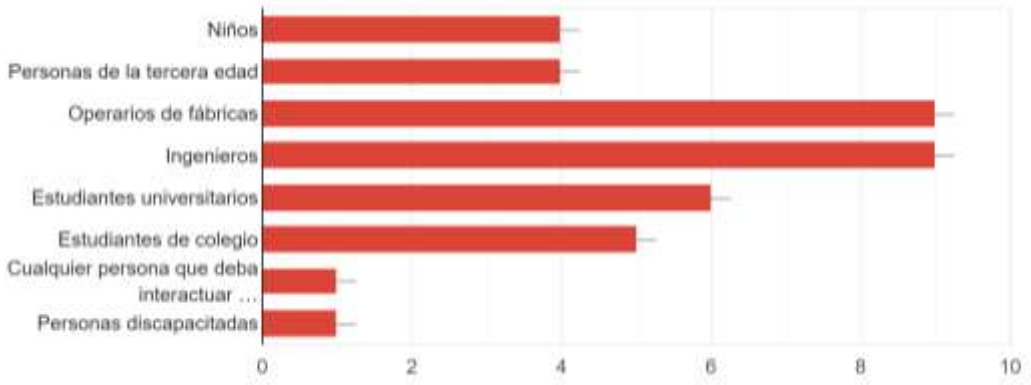

Figura 13 Preguntas abiertas sobre aplicación del desarrollo

Con el desarrolló del trabajo de grado se creó una interfaz capaz de ser un puente o traductor entre las personas y el robot, un puente que permite ahorrar el tiempo que se invertiría en el aprendizaje de un lenguaje de programación, gracias al resultado obtenido se tiene la capacidad de que cualquier persona, con el simple hecho de escribir las órdenes que quiere, en lenguaje natural, sea capaz de configurar un dispositivo tecnológico de punta, como es un CoBot.

Además, permite realizar esas configuraciones de forma remota y con la capacidad de brindar información en tiempo real de las articulaciones del robot. La unión de las tecnologías de punta que se utilizaron en este trabajo permitirá que personas que se sientan excluidas por la falta de conocimientos tecnológicos puedan comenzar a desenvolverse de una manera natural en el medio. En los objetivos planteados inicialmente se consideraba tener una interfaz que permitiera estructurar algunas programaciones en base a unas ya predeterminadas, sin embargo, el resultado demuestra que se cumplió con esto y se generó una interfaz que permite guiar a las personas, realizar cualquier movimiento articular, tomar objetos en posiciones predeterminada, guardar las programaciones realizadas por medio de la interfaz, conocer los ángulos en tiempo real y con la capacidad de comunicarse remotamente.

4. CONCLUSIONES Y CONSIDERACIONES FINALES

4.1 CONCLUSIONES

El enfoque de la mecatrónica siempre ha ido dirigido a la integración de conocimientos para generar desarrollos de valor, en este trabajo de grado se reúnen diferentes tecnologías que dan como resultado una aplicación con la capacidad de acercar a las personas a la tecnología de una manera intuitiva.

Los pilares fundamentales de este proyecto son los robots colaborativos, el procesamiento de lenguaje natural por medio de inteligencia artificial, la implementación de tecnologías Cloud, el desarrollo de interfaces web para diferentes dispositivos y la automatización robótica de procesos. Este conjunto de tecnologías hace posible que una persona sin ningún conocimiento previo de programación pueda configurar CoBots que se utilizan en la industria, lo que significa un avance en la implementación de estos en diferentes procesos.

La versatilidad en la creación de interfaces web que posee la herramienta Node-red es, sin lugar a duda, la razón principal para ser la herramienta apropiada a la hora de prototipar interfaces, a esto se le suma su amplia gama de nodos integrados a otras aplicaciones, como por ejemplo el paquete de IBM WATSON que fue usado en este proyecto.

El procesamiento de lenguaje natural para la creación de asistentes virtuales posee gran demanda debido a su amplio panorama de implementaciones, esto hace que existan diferentes herramientas de las mejores empresas de tecnología, como Google, IBM o Microsoft. Para este proyecto se implementó IBM WATSON debido a que posee una versión gratuita con gran demanda de solicitudes al API del Cloud y tiene una interfaz con diferentes funcionalidades que permiten crear un asistente virtual robusto. Además, se decidió implementar el asistente virtual y no simplemente un procesador de lenguaje, debido a que el segundo no podía generar una interacción amigable y guiar al usuario en el proceso de la creación del agente.

Por otra parte, en el desarrollo del proyecto se planteó inicialmente realizar la implementación completa de un motor lingüístico utilizando herramientas de Deep Learning, sin embargo, se consideró que utilizar las tecnologías Cloud, que prestan esta herramienta como servicio, esta implementación se consideró como valor diferenciador debido a que estos conocimientos con lo que los estudiantes se relacionan en su transcurso en la universidad y que en el sector laboral se vuelven

muy útiles.

El RPA es una tecnología muy útil en las empresas prestadoras de servicios, sin embargo, para este proyecto los softwares comerciales y robustos no son implementables debido a que solo son compatibles con un sistema operativo Windows, y para la comunicación con el CoBot se debe utilizar Ubuntu, la solución a este fue implementar librerías del lenguaje de programación Python, este permitió tener algunas funcionalidades de softwares como UiPath o Automation Anywhere y permitió llegar al objetivo del proyecto.

La forma encontrada para la comunicación de los diferentes sistemas fue la creación de servidores web, esto permite que los datos se transfieran rápidamente desde la interfaz hasta el computador que está conectado al CoBot. Estos conocimientos deberían ser abordados en alguna materia de la malla curricular de la carrera debido a su amplio campo de aplicación.

4.2 CONSIDERACIONES FINALES Y TRABAJO FUTURO

Al realizar desarrollos complementarios se generarían la oportunidad de que el trabajo de grado tenga campos de acción en diferentes áreas, debido a que se podría tener una interfaz de programación de robot colaborativos muy robusta e incluso aplicable al mercado. Las funcionalidades propuestas para las futuras etapas de proyecto son las siguientes:

4.2.1 Implementación de Hardware diferente.

Para la comunicación con el robot se usó un computador, el cual tenía el sistema operativo necesario para la conexión con el robot colaborativos, Ubuntu 14.04, se propone la búsqueda de implementar otro tipo de hardware que soporte el sistema operativo necesario y que tenga la capacidad de comunicarse con el Baxter.

4.1.1 Manejo de sesiones en la plataforma.

El objetivo de que CoBot pueda ser programada de forma remota se cumplió, sin embargo, para que esta implementación esté completa se necesita crear una capa de sesión en la plataforma que permita identificar la persona que está interactuando con él, esto con el fin de evitar colisiones en la información. Además, crear roles que permitan dar funcionalidades extra, como borrar programar, eliminar usuarios, entre otros.

4.1.2 Ingreso de ordenes por voz.

Al querer generar una experiencia más agradable para el usuario, se propone agregar una opción de enviar las ordenes provenientes del usuario a Baxter Asisstant por medio de

voz. Esto requiere conocimientos en desarrollo de páginas web, debido a que se debe acceder a los periféricos del dispositivo desde el navegador.

5. REFERENCIAS

(29 de Abril de 2008). Obtenido de BBC MUNDO: http://news.bbc.co.uk/hi/spanish/business/newsid_7374000/7374275.stm

(15 de Agosto de 2016). Obtenido de UTADEO: http://www.utadeo.edu.co/es/link/maestria-en-modelado-y-simulacion-mms/26106/layout-1/que-es-modelado-y-simulacion-ms

(15 de Agosto de 2016). Obtenido de sparkfun: http://cdn.sparkfun.com/datasheets/Sensors/Temp/ntcle100.pdf

(8 de Agosto de 2016). Obtenido de python: https://www.python.org/doc/essays/blurb/

(08 de 08 de 2016). Obtenido de OpenCV: http://opencv.org/

(15 de Agosto de 2016). Obtenido de Mecatronica: http://www.mecatronica.es/index.php?option=com_content&view=article&id=111:principales-fabricantes-de-componentes-electronicos&catid=1:latest-news&Itemid=1

(16 de Agosto de 2016). Obtenido de littelfuse: http://www.littelfuse.com/~/media/electronics/datasheets/varistors/littelfuse_varistor_la_datasheet.pdf.pdf

(15 de Agosto de 2016). Obtenido de Keyword: http://www.keyword-suggestions.com/MWsgcmVzaXN0b3IgZGF0YSBzaGVldA/

(08 de Agosto de 2016). Obtenido de Interacpedia: https://www.interacpedia.com/co-create/challenge

(15 de Agosto de 2016). Obtenido de ÉLOGISTCA: http://www.logisticamx.enfasis.com/notas/72941-modelos-simulacion-manufactura

(15 de Agosto de 2016). Obtenido de Electan: http://www.electan.com/datasheets/cebek/CE-C2795.pdf

Acharya, V. (17 de Abril de 2017). El análisis de los factores en la automatización industrial utilizando el proceso analítico jerárquico . *Computadoras e ingeniería electrica*.

Anderson, J. M., Kalra, N., Stanley, K. D., Sorensen, P., Samaras, C., & Oluwatola, O. A. (22 de March de 2016). *RAND CORPORATION.* Obtenido de https://www.rand.org/pubs/research_reports/RR443-2.html

Anguiano, J. (30 de Junio de 2014). *IBM*. Obtenido de https://www.ibm.com/developerworks/ssa/data/library/tipos_bases_de_datos/index.html

Arias, G. (15 de Marzo de 2010). *El Financiero.* Obtenido de http://wvw.elfinancierocr.com/ef_archivo/2012/agosto/26/economia3284455.html

Automation Anywhere. (s.f.). *Automation Anywhere*. Obtenido de https://www.automationanywhere.com

BBC Mundo. (29 de Abril de 2008). Obtenido de http://news.bbc.co.uk/hi/spanish/business/newsid_7374000/7374275.stm

Blue Prism . (s.f.). *Blue Prism* .

Cámara de Comercio de Medellín. (2016). *Herramientas empresariales Cámara de Comercio de Medellón*. Obtenido de http://herramientas.camaramedellin.com.co/Inicio/BuenaspracticasempresarialesBibliotecaProducci%C3%B3nyOperaciones/Automatizaciondelosprocesosindustriales.aspx

Cedillo Méndez, J. L., Rafael Esteban, F., & Ramírez Salas Linares, L. O. (08 de Mayo de 2012). Optimización de ancho de banda para sistemas GSM.

Cosourcing . (s.f.). *Cosourcing Partners*. Obtenido de https://cosourcingpartners.com/

Dinero. (2017). Sin marcha atrás: La automatización será una realidad en Colombia en 2020. *Dinero*.

Dolar Web . (2018). *Dolar wilkinson*. Obtenido de https://dolar.wilkinsonpc.com.co/dolar.php

El Financiero. (10 de Abril de 2018). ¿Qué son los robots colaborativos y por qué son idealespara las Pymes?

EY building a better working world. (junio de 2017). *Ey boletin* . Obtenido de http://www.eyboletin.com.mx/boletines/eventos/agradecimiento_Conferencia_Desayuno_EY-RPA.pdf

Foreman, C. R. (2000). *Estados unidos Patente nº Real-Time data acquisition.*

Fourier. (s.f.). *Fouriering*. Obtenido de http://www.fouriering.com/

Gandotra, R. (26 de Diciembre de 2017). *ETCIO*. Obtenido de https://cio.economictimes.indiatimes.com/news/strategy-and-management/ai-blockchain-iot-cloud-rpa-and-3d-printing-will-be-the-game-changers-in-2018/62247932

García, A. d. (11 de Junio de 2017). IoT: Dispositivos, tecnologías de transporte y aplicaciones. Cataluña, España.

Ghica, A. (2017). *Uipath* . Obtenido de https://www.uipath.com/press-room/uipath-rpa-leader-star-performer-everest-group-rpa-peak-matrix-2018

Google . (s.f.). *Google cloud* . Obtenido de https://cloud.google.com/natural-language/

Google . (s.f.). *Google Cloud* . Obtenido de https://cloud.google.com/natural-language/

Hager, g., & Padoy, N. (2011). *Patente nº US20130218340A1.*

Halvorsen, H.-P. (28 de Octubre de 2016). Data Acquisition in LabVIEW. Obtenido de http://home.hit.no/~hansha/documents/labview/training/Data%20Acquisition%20in%20LabVIEW/Data%20Acquisition%20in%20LabVIEW.pdf

HTML 5. (s.f.). *W3*. Obtenido de https://www.w3.org/html/logo/

IBM. (28 de 02 de 2019). *IBM Cloud*. Obtenido de https://cloud.ibm.com/docs/services/assistant?topic=assistant-intents

IBM. (s.f.). *IBM*. Obtenido de https://www.ibm.com/watson/

IBM. (s.f.). *IBM*. Obtenido de https://www.ibm.com/watson/

Intel software. (22 de julio de 2014). *Developer zone* . Obtenido de https://software.intel.com/es-es/android/articles/pros-and-cons-of-html5-cross-platform-android-mobile-app-development-tools-on-intel

Interoute . (s.f.). *interoute from the ground to the cloud* . Obtenido de https://www.interoute.es/what-paas

Java Script. (s.f.). *Java Script*. Obtenido de https://www.javascript.com/

Javier Moreno, D. R. (Junio de 2007). *Universidad de Alicante.* Obtenido de https://rua.ua.es/dspace/bitstream/10045/1109/7/Informe_ZigBee.pdf

Jose Garzón, D. G. (2017). Sistema de adquisición de datos de bajo costo aplicable a entornos industriales.

Jose Miguel Garzón Vargas, D. G. (2017). Sistema de adquisición de datos de bajo costo aplicable a entornos industriales. Medellín, Colombia.

Koukkari, T. (2 de Diciembre de 2016). Collaborative robotics . Seinäjoki, Finlandia.

Lefcovich, M. (s.f.). Recuperado el 21 de Julio de 2017, de https://www.gestiopolis.com/just-in-time-camino-hacia-la-excelencia/

Lefcovich, M. (s.f.). *Gestiopolis*. Recuperado el 21 de Julio de 2017, de https://www.gestiopolis.com/just-in-time-camino-hacia-la-excelencia/

Litoral, F. C.–U. (28 de Abril de 2017). Obtenido de http://beta.mba.americaeconomia.com/sites/mba.americaeconomia.com/files/potenciasemergentes_bric_corvalan_delbarco.pdf

Lora Alliance. (Noviembre de 2015). *TUV.* Obtenido de https://www.tuv.com/media/corporate/products_1/electronic_components_and_lasers/TUeV_Rheinland_Overview_LoRa_and_LoRaWANtmp.pdf

Mabie, S. (s.f.). *universal robots* .

Marin, F. (s.f.). Recuperado el 21 de Julio de 2017, de http://www.cge.es/portalcge/tecnologia/innovacion/4115sistemajust.aspx

Marius Ghercioiu, H. H. (2005). *Estados Unidos Patente nº US20050289274 A1.*

Martínez. (10 de Marzo de 2010). Obtenido de http://www.eumed.net/rev/china/10/dqm.htm

Martínez, D. Q. (10 de Marzo de 2009). Obtenido de http://www.eumed.net/rev/china/10/dqm.htm

Mataweb. (9 de Agosto de 2017). Obtenido de http://www.matweb.com/search/DataSheet.aspx?MatGUID=eeea822d8b87419f9f1ff8c950f2669c

Measurement Computing. (2012). *Data Acquisition.* Estados Unidos.

Medeiros, C. M. (2009). *Advanced geographic information systems .* Unesco.

Minutos, R. R. (21 de Enero de 2010). *20 Minutos.* Obtenido de http://www.20minutos.es/noticia/611279/0/economia/china/2009/

MIT. (s.f.). *My App Inventor*. Obtenido de http://appinventor.mit.edu/explore/

Molina, F. J. (2012). Diseño y construcción de un vehículo de tracción humana para la competencia UNIANDES 2011. Bucaramanga, Colombia: Universidad Pontifica Bolivariana.

Montoya, J. R. (s.f.). *Herramientas comerciales Cámara de Comercio Medellín*. Obtenido de

http://herramientas.camaramedellin.com.co/Inicio/BuenaspracticasempresarialesBibliotecaGerenciaEstrategica/Direccionamientoestrategico.aspx

National Instrument. (03 de Mayo de 2013). Obtenido de http://www.ni.com/white-paper/3536/en/

National instruments. (2012). *National instruments*. Obtenido de http://www.ni.com/data-acquisition/what-is/esa/

NICE Robotics process automation . (s.f.). *NICE*.

Node JS. (s.f.). *node js* . Obtenido de https://nodejs.org/es/

Node RED. (s.f.). *Node RED*. Obtenido de https://nodered.org/

Ogata, K. (s.f.). *Ingeniería de control modena.*

Omer, A. I. (2014). ARCHITECTURE OF INDUSTRIAL. *European Scientific Journal* , 274.

osa, E. d. (8 de 11 de 2011). *El País.* Obtenido de http://internacional.elpais.com/internacional/2011/10/17/actualidad/1318844584_652292.html

Papa, E., & Ferreira, A. (12 de January de 2018). Sustainable Accessibility and the Implementation of Automated Vehicles: Identifying Critical Decisions. *urban science*, pág. 14.

Perry C. Steger, G. W. (2009). *Estados Unidos Patente nº US7542867B2.*

Peshkin, M., Wannasuphoprasit, W., & Colgate, E. (1 de Diciembre de 1996). Cobots: robots for collaboration with human operators.

Plan Automation Technology . (2017). *Plan Automation Technology* . Obtenido de https://www.plantautomation-technology.com/articles/why-is-industrial-automation-important

prensa, R. i. (20 de Febrero de 2012). *El Economista.* Obtenido de http://eleconomista.com.mx/economia-global/2012/02/20/china-principal-socio-comercial-brasil-chile

Python. (s.f.). *Python*. Obtenido de https://www.python.org/

Quiroz, D., & Yarlequé, J. (2017). Aplicación web móvil con geolocalización para mejorar la experiencia de compra del consumidor de Trujillo en la busqueda de promociones en supermercados en el año 2016. Trujillo, Perú.

Redacción. (26 de Septiembre de 2012). *Voa Noticias.* Obtenido de http://www.voanoticias.com/a/venezuela-elecciones-hugo-chavez-financiacion-proyectos-obras-deuda-banco-china/1515010.html

RETANA, G. A. (20 de Agosto de 2012). *El financiero.* Obtenido de http://www.elfinancierocr.com/economia-y-politica/invierte-China-region_0_140986046.html

Rethink robotics. (s.f.). *Rethink robotics*. Obtenido de https://www.rethinkrobotics.com/baxter/

Rethink robotics. (s.f.). *Rethink robotics* . Obtenido de https://www.rethinkrobotics.com/smart-collaborative-difference/

Rivas, J. M. (s.f.). *National Instrument.* Obtenido de ftp://ftp.ni.com/pub/branches/spain/eventos/multimedia/fundamentos_tecnologia_adquisicion_datos.pdf

Romero, S. (17 de Agosto de 2010). *LA NACIÓN .* Obtenido de http://www.lanacion.com.ar/1295346-la-otra-cara-de-las-inversiones-de-pekin-en-america-latina

Rouse, M. (s.f.). *Techtraget* . Obtenido de Search Data Center: https://searchdatacenter.techtarget.com/es/definicion/Biometria

S.A. (S.F). *Coche Español*. Obtenido de http://www.automotriz.mobi/coches/cars-trucks-autos/other-autos/112723.html

Siega. (17 de Junio de 2010). *Rebelión.* Obtenido de http://rebelion.org/noticia.php?id=107992

Siega, V. d. (17 de Junio de 2010). *Rebelión.* Obtenido de http://rebelion.org/noticia.php?id=107992

Sigfox technical overview. (Mayo de 2017). *Disk91.* Obtenido de https://www.disk91.com/wp-content/uploads/2017/05/4967675830228422064.pdf

Tabuenca, D. (2017). Implementación de robots colaborativos en linea de producción. Valladolid, España.

Talleres y Repuestos. (9 de Agosto de 2017). Obtenido de http://talleresyrepuestos.com/documentacion-tecnica/inyeccion-electronica-de-combustible/12-mantenimiento-del-tanque-de-gasolina

Tartera, P. (s.f.). Recuperado el 21 de Julio de 2017, de https://www.youtube.com/watch?v=czVxWjlrG30

Udai, A. D. (Octubre de 2013). Make your own USB data acquisition system. India.

UiPath. (s.f.). *UiPath*. Obtenido de https://www.uipath.com/

Ulrich, K., & Eppinger, S. (2008). *Diseño y desarrollo de productos.* Mc Graw Hill.

Universal Robots. (Abril de 2017). El futuro es colaborativo.

Universal Robots. (s.f.). *Universal Robots*. Obtenido de https://www.universal-robots.com/

Universidad de Chile. (2012). Vigilando las fronteras tecnologicas . Santiago, Chile.

Universidad Externado de Colombia . (27 de junio de 2016). *Revista u externado*. Obtenido de http://revistas.uexternado.edu.co/index.php/propin/article/view/4602/5519

ANEXOS

Anexos 1 Vigilancia tecnologíca

Sistematic Literature Review.xlsx

Anexos 2 Servidor web

```
#------------------Importar librerias----------------------------------
import pyautogui
from flask import Flask, render_template
from flask import Flask, request, jsonify
import time,json
import os

#------------------Inicializar variales ---------------------------------
pyautogui.FAILSAFE = True
pyautogui.PAUSE = 1
def borrar_registros():
    wave1 = {str("right")+'_s0': 0, str("right")+'_s1': 0, str("right")+'_e0': 0, str("right")+'_e1': 0,
    str("right")+'_w0': 0, str("right")+'_w1': 0, str("right")+'_w2': 0}
    with open("movimientosright.txt",'w'): pass
    text_file = open("movimientosright.txt", "w")
    text_file.write(str(wave1))
    text_file.close()
    with open("movimientosleft.txt",'w'): pass
    text_file = open("movimientosleft.txt", "w")
    text_file.write(str(wave1))
    text_file.close()

#------------------definicion funciones---------------------------------
def abrirterminal():
    pyautogui.click(x=100, y=0, clicks=1, button='left')
    pyautogui.hotkey('Ctrl','Alt','t')

def escribirterminal(comando):
    pyautogui.typewrite(comando, interval=0)
    pyautogui.hotkey('ENTER')


#------------------config rutas----------------------------------
app = Flask(__name__)

@app.route('/nuevas', methods=['POST'])
def prueba_datos():
    datos=request.json['parte']
    if (datos['confirmacion']==True):
        with open('datos.txt','w'): pass
        text_file = open("datos.txt", "w")
        text_file.write(str(datos))
        text_file.close()
        if(datos['accion']=="tomar"):
            escribirterminal("python tomarfoto.py")
```

```
            time.sleep(1)
            escribirterminal("python mov_arms.py")
        if(datos['accion']=="mover"):
            escribirterminal("python pose.py")
        if(datos['accion']=="guardar"):
            os.rename("movimientosleft.txt",
"Programas/"+str(datos['nombre_programa'])+"l.txt")
            os.rename("movimientosright.txt",
"Programas/"+str(datos['nombre_programa'])+"r.txt")
        if(datos['accion']=="ejecutar"):
            escribirterminal("python programas_guardadas.py")

        time.sleep(10)
        escribirterminal("python obtener_pos.py")
        datos = open("pos_actual.txt", "r")
        valores=eval(datos.read())
        return(jsonify(datos['system']['dialog_stack'][0]['dialog_node']))
    else:
        escribirterminal("python obtener_pos.py")
        datos = open("pos_actual.txt", "r")
        #valores=eval(datos.read())
        return (datos.read())


#Ruta de carga de informacion
@app.route("/grabadas", methods=["POST"])
def home():
    payload = request.json
    if (payload["accion"]=='iniciar'):
        abrirterminal()
        escribirterminal("cd ros_ws")
        escribirterminal(". baxter.sh")
        escribirterminal("rosrun baxter_tools enable_robot.py -e")
        time.sleep(10)
        escribirterminal("python obtener_pos.py")
    if (payload["accion"]=='set'):
        escribirterminal("rosrun baxter_tools tuck_arms.py -u")
        borrar_registros()
        time.sleep(5)
        escribirterminal("python obtener_pos.py")
    if(payload["accion"]=="apagar"):
        escribirterminal("rosrun baxter_tools tuck_arms.py -t")
        escribirterminal("exit")
        escribirterminal("exit")
        borrar_registros()
    print(payload["accion"])
    datos = open("pos_actual.txt", "r")
```

```
    #valores=eval(datos.read())
    return (datos.read())

#Inicializacion del servidor
if __name__ == "__main__":
    app.run(debug=True)
```

Anexos 3 Tomar foto y tomar objetos

```
import rospy
import baxter_interface
from baxter_interface import CHECK_VERSION
from sensor_msgs.msg import Image
from cv_bridge import CvBridge, CvBridgeError
import cv2
from posiciones import der_ini
from posiciones import izq_ini
bridge = CvBridge()

def config_inicial():
    global bd
    rospy.init_node('image_listener', anonymous=True)
    print("Obteniendo el estado del robot... ")
    baxter_interface.RobotEnable(CHECK_VERSION).state().enabled
    print("Habilitando motores... ")
    baxter_interface.RobotEnable(CHECK_VERSION).enable()
    bd = baxter_interface.Limb('right')
    bd.set_joint_position_speed(1.0)

def ubi_cam():
    bd.move_to_joint_positions(der_ini[2])
    '''
    for i in range(0,3):
        bd.move_to_joint_positions(der_ini[i])
    '''
def image_callback(msg):
    print("Recibe una imagen")
    global img
    try:
        img = bridge.imgmsg_to_cv2(msg, "bgr8")
    except CvBridgeError, e:
        print(e)
    else:
        cv2.imwrite('/home/juan/ros_ws/c1.jpg', img)

def main():
    config_inicial()
    ubi_cam()
    image_topic = "/cameras/right_hand_camera/image"
    rospy.Subscriber(image_topic, Image, image_callback)
    rospy.sleep(0.1)
    rospy.signal_shutdown("fin del proceso")

if __name__ == '__main__':
```

```
    main()
import rospy
import baxter_interface
from baxter_interface import CHECK_VERSION

import cv2
import numpy as np
import math

##------Encontrar objeto-----------
xx=0
yy=0
Ruta="c1.jpg"
kernel = np.ones((3,3),np.uint8)
foto1= cv2.imread(Ruta)
foto = foto1[120:379, 137:543]
foto_hsv=cv2.cvtColor(foto,cv2.COLOR_BGR2HSV)
foto_gris=cv2.cvtColor(foto,cv2.COLOR_BGR2GRAY)
hh,w=foto.shape[:2]

#leer datos
datos = open("datos.txt", "r")
valores=eval(datos.read())

if (valores['objeto']=='engranaje'):
    ## ------GRIS------
    lower=np.array([0,0,0])
    upper=np.array([255,110,43])
    foto_gris=cv2.inRange(foto_hsv,lower,upper)


if (valores['objeto']=='pieza'):
    ##------Azul------
    lower=np.array([0,118,0])
    upper=np.array([255,255,51])
    foto_gris=cv2.inRange(foto_hsv,lower,upper)

opening = cv2.morphologyEx(foto_gris, cv2.MORPH_OPEN, kernel)

##------Encontrar lado
der=[]
izq=[]
hh1,w1=opening.shape[:2]
for i in range(0,hh1):
    for j in range(0,w1):
        x = opening[i,j]
```

```
        if(x==255):
            if(j>203):
                der.append(j)
            else:
                izq.append(j)
        else:
             foto_gris[i,j]=0
if(len(der)>len(izq)):
    lado="Derecha"
else:
    lado="Izquierda"
#cv2.imshow("Opening", opening)
#cv2.waitKey(0)
#cv2.destroyAllWindows()
##---------Tomar objeto------------
rospy.init_node('cinematica_inversa')

baxter_interface.RobotEnable(CHECK_VERSION).state().enabled
baxter_interface.RobotEnable(CHECK_VERSION).enable()

gripD= baxter_interface.Gripper('right',CHECK_VERSION)
gripL= baxter_interface.Gripper('left',CHECK_VERSION)
limbi = baxter_interface.Limb('left')
limbd = baxter_interface.Limb('right')

'''
anglesi = limbi.joint_angles()
anglesd = limbd.joint_angles()

print "izq = ", anglesi
print "der = ", anglesd
'''


homeDer =  {'right_s0': 0.07554855380335662, 'right_s1': -0.985199161019407,
'right_w0': -0.6611457195786133, 'right_w1': 1.0607477148227635, 'right_w2':
0.5253884198507542, 'right_e0': 1.1554710284746879, 'right_e1': 1.8756750083868896}
der =  {'right_s0': 0.650024358866444, 'right_s1': -0.3267379078195931, 'right_w0': -
0.7528010716547668, 'right_w1': 0.9779127522769513, 'right_w2': 0.5165680303204131,
'right_e0': 0.5905826033358843, 'right_e1': 1.1412817061867477}
der1 =  {'right_s0': -0.0003834951969713534, 'right_s1': -0.6711165946998685, 'right_w0':
-0.44753889486556947, 'right_w1': 1.7142235304619498, 'right_w2':
1.0726360659288756, 'right_e0': 0.4425534573049419, 'right_e1': 0.5806117282146291}
homeizq =  {'left_w0': 0.6676651379271263, 'left_w1': 1.0281506230801987, 'left_w2': -
0.5019952128355016, 'left_e0': -1.1876846250202815, 'left_e1': 1.9424031726599051,
'left_s0': -0.08168447695489828, 'left_s1': -1.0013059592922038}
```

izq = {'left_w0': 0.6128253247602228, 'left_w1': 0.8356360342005792, 'left_w2': -0.4816699673960199, 'left_e0': -0.4801359866081345, 'left_e1': 1.279339977096435, 'left_s0': -0.6527088252452435, 'left_s1': -0.39883500485020756}

```
if(lado=="Derecha"):
    limbd.move_to_joint_positions(homeDer)
    gripD.calibrate()
    limbd.move_to_joint_positions(der)
    gripD.close()
    rospy.sleep(1)
    limbd.move_to_joint_positions(der1)
    gripD.open()
    rospy.sleep(1)
    limbd.move_to_joint_positions(homeDer)
if(lado=="Izquierda"):
    limbi.move_to_joint_positions(homeizq)
    gripL.calibrate()
    limbi.move_to_joint_positions(izq)
    gripL.close()
    rospy.sleep(1)
    limbi.move_to_joint_positions(homeizq)
    gripL.open()
    limbd.move_to_joint_positions(homeDer)
```

Anexos 4 Mover articulaciones del robot

```
#coding=utf-8

import rospy
import baxter_interface
from baxter_interface import CHECK_VERSION
from threading import Thread
import math

rospy.init_node('simultaneo')

baxter_interface.RobotEnable(CHECK_VERSION).state().enabled
baxter_interface.RobotEnable(CHECK_VERSION).enable()

cabecita=baxter_interface.Head()
gripI= baxter_interface.Gripper('left',CHECK_VERSION)
gripD= baxter_interface.Gripper('right',CHECK_VERSION)
limbi = baxter_interface.Limb('left')
limbd = baxter_interface.Limb('right')
#leer datos
datos = open("datos.txt", "r")
valores=eval(datos.read())
#clase que configura los datos
class brazos:
    def __init__(self, brazo,arti,cantidad):
        if (brazo=="derecha"):
            arm="right"
        else:
            arm="left"
        self.brazo = arm
        self.arti=arti
        rad=cantidad*(math.pi/180)
        self.cantidad=rad

class cabeza:
    def __init__(self,direccion,cantidad):
        if (direccion=="derecha"):
            arm="-"
        else:
            arm=""
        self.direccion = arm
        self.cantidad= cantidad/90.0

def config_data(x,contador):
    if(contador==1):
```

```
        #Crear archivo de izq
        wave2 = {str("left")+'_s0': 0, str("left")+'_s1': 0, str("left")+'_e0': 0, str("left")+'_e1': 0,
        str("left")+'_w0': 0, str("left")+'_w1': 0, str("left")+'_w2': 0}
        with open("movimientosleft.txt",'w'): pass
        text_file = open("movimientosleft.txt", "w")
        text_file.write(str(wave2))
        text_file.close()
        #crear archivo de Derecha
        wave1 = {str("right")+'_s0': 0, str("right")+'_s1': 0, str("right")+'_e0': 0, str("right")+'_e1':
0,
        str("right")+'_w0': 0, str("right")+'_w1': 0, str("right")+'_w2': 0}
        with open("movimientosright.txt",'w'): pass
        text_file = open("movimientosright.txt", "w")
        text_file.write(str(wave1))
        text_file.close()
        if x.brazo=="right":
            wave1.update({str(x.brazo)+"_"+str(x.arti): x.cantidad})
            with open("movimientosright.txt",'w'): pass
            text_file = open("movimientosright.txt", "w")
            text_file.write(str(wave1))
            text_file.close()
            wave=wave1
        else:
            wave2.update({str(x.brazo)+"_"+str(x.arti): x.cantidad})
            with open("movimientosleft.txt",'w'): pass
            text_file = open("movimientosleft.txt", "w")
            text_file.write(str(wave2))
            text_file.close()
            wave=wave2
    else:
        datos = open("movimientos{}.txt".format(x.brazo), "r")
        wave=eval(datos.read())
        wave.update({str(x.brazo)+"_"+str(x.arti): x.cantidad})
        with open("movimientos{}.txt".format(x.brazo),'w'): pass
        text_file = open("movimientos{}.txt".format(x.brazo), "w")
        text_file.write(str(wave))
        text_file.close()

    return(wave)

def mover_brazos(posicion, brazo):
        if brazo =='left':
                limbi.move_to_joint_positions(posicion)
        if brazo =='right':
                limbd.move_to_joint_positions(posicion)
```

```
limbi.set_joint_position_speed(0.5)
limbd.set_joint_position_speed(0.5)

izq1 =  {'left_w0': 0, 'left_w1': 0, 'left_w2': 0, 'left_e0': 0, 'left_e1': 0, 'left_s0': 0, 'left_s1': 0}
der1 = {'right_s0': 0, 'right_s1': 0, 'right_w0': 0, 'right_w1': 0, 'right_w2': 0, 'right_e0': 0,
'right_e1': 0}

if(valores['partes']=='brazo'):
    x=brazos(valores["direccion"],valores["articulaciones"].lower(),valores["grados"])

    if(valores['contador']==1):
        print "comienza rutina inicial"
        if (x.brazo=='left'):
            t1=Thread(target=mover_brazos, args=(izq1,'left',))
            t1.start()
            t1.join()

        else:
            t1=Thread(target=mover_brazos, args=(der1,'right',))
            t1.start()
            t1.join()

    print "comienza rutina programada"
    t2=Thread(target=mover_brazos, args=(config_data(x,valores['contador']),x.brazo,))
    t2.start()
    t2.join()
elif(valores['partes']=='cabeza'):
    y=cabeza(valores["direccion"],valores["grados"])
    pos=str(y.direccion)+str(y.cantidad)
    cabecita.set_pan(float(pos))
elif(valores['partes']=='garra'):
    if(valores['direccion']=='derecha'):
        gripD.calibrate()
        if(valores['accion_garra']=='abrir'):
            gripD.open()
        else:
            gripD.close()
    if(valores['direccion']=='Izquierda'):
        gripI.calibrate()
        if(valores['accion_garra']=='abrir'):
            gripD.open()
        else:
            gripD.close()
```

Anexos 5 Ejecutar programaciones guardadas

```
import rospy
import baxter_interface
from baxter_interface import CHECK_VERSION
from threading import Thread
import math

rospy.init_node('simultaneo')

baxter_interface.RobotEnable(CHECK_VERSION).state().enabled
baxter_interface.RobotEnable(CHECK_VERSION).enable()

cabecita=baxter_interface.Head()
gripI= baxter_interface.Gripper('left',CHECK_VERSION)
gripD= baxter_interface.Gripper('right',CHECK_VERSION)
limbi = baxter_interface.Limb('left')
limbd = baxter_interface.Limb('right')
#leer datos
datos = open("datos.txt", "r")
valores=eval(datos.read())
left =
open("Programas/{}.txt".format(str(valores['nombre_programa'].split("ejecutar")[1][1:][:-
1])+"l"), "r")
izq=eval(left.read())

right =
open("Programas/{}.txt".format(str(valores['nombre_programa'].split("ejecutar")[1][1:][:-
1])+"r"), "r")
der=eval(right.read())
#clase que configura los datos
class brazos:
    def __init__(self, brazo,arti,cantidad):
        if (brazo=="derecha"):
            arm="right"
        else:
            arm="left"
        self.brazo = arm
        self.arti=arti
        rad=cantidad*(math.pi/180)
        self.cantidad=rad

class cabeza:
    def __init__(self,direccion,cantidad):
        if (direccion=="derecha"):
            arm="-"
        else:
```

```
            arm=""
        self.direccion = arm
        self.cantidad= cantidad/90.0

def mover_brazos(posicion, brazo):
            if brazo =='left':
                        limbi.move_to_joint_positions(posicion)
            if brazo =='right':
                        limbd.move_to_joint_positions(posicion)

limbi.set_joint_position_speed(0.5)
limbd.set_joint_position_speed(0.5)

t1=Thread(target=mover_brazos, args=(izq,'left',))
t1.start()
t1=Thread(target=mover_brazos, args=(der,'right',))
t1.start()
t1.join()
```