Diseño de una pseudo-mano para tocar piano con el robot UR5

Arroyo N., Belen M., Palma D., Solorzano R.
Universidad de Ingeniería y Tecnología – UTEC, Lima, Perú

***Resumen*—En este paper se presenta el diseño y la implementación de una pseudo-mano robórica de cinco dedos, cada uno con un grado de libertad, para lograr que el robot UR5 sea capaz de tocar melodías de complejidad media en un teclado. Se presenta la metodología que siguió el proyecto, la cual se divide en cinco etapas. En la primera etapa se muestra el diseño de la pseudo-mano, en el que se simulan los movimientos y dimensiones de la misma. En la segunda etapa se explica el procedimiento y la lógica por los cuales se interpreta una canción guardada en un archivo MIDI. En la tercera etapa se explica la comunicación entre la Computadora, la Raspberry Pi y el controlador de servomotores a través de ROS. En la cuarta etapa se muestra la lógica que se usa para mover cada dedo de la mano utilizando servomotores. Finalmente, en la quinta etapa, se muestra el desarrollo de una simulación del movimiento del UR5 en el programa Gazebo. De estas cinco etapas de desarrollo se obtiene como resultado la sincronización del movimiento de la simulación del UR5 junto con el movimiento de los servomotores al mandarles la misma data por medio de los tópicos de ROS. Sin embargo, para una implementación real se requiere de la impresión en 3D, el UR5 real, un teclado físico, algunos cambios mínimos en los códigos de comunicación por ROS y ajustes en los parámetros de los servomotores.**

***Palabras clave*— pseudo-mano, UR5, piano, ROS, MIDI**

I. Introduction

Con el desarrollo de la ciencia y la tecnología, los robots son cada vez más utilizados en la vida moderna, pues son capaces de mejorar nuestra calidad de vida. Además de los robots industriales, los robots domésticos, o robots de servicio y entretenimiento, han sido muy apreciados por la mayoría de las agencias de investigación de mercado y robótica [1]. Un ejemplo de ello son los robots que pueden tocar instrumentos, debido a que representan un avance significativo en el campo de la robótica y una gran atracción para el público. En los últimos años, se han propuesto diversos diseños de robots musicales, los cuales pueden tocar instrumentos como el violín, guitarra, flauta, tambor, trompeta y piano. Entre ellos, el robot que toca piano (Fig. 1) es uno de los que causa más interés, pues es uno de los instrumentos más populares y complejos [2].

En ese sentido, el presente proyecto pretende diseñar y simular una pseudo-mano con el objetivo de aumentar la capacidad del robot UR5 para que pueda tocar piano. Para ello, la pseudo-mano actúa como efector final del robot y está compuesta por dedos que poseen un grado de libertad, los cuales se encuentran ubicados de forma paralela y uno al costado del otro. Por ello es que se le denomina pseudo-mano, ya que no cuenta con una forma antropomorfa, ni con todos los grados de libertad de una mano real.

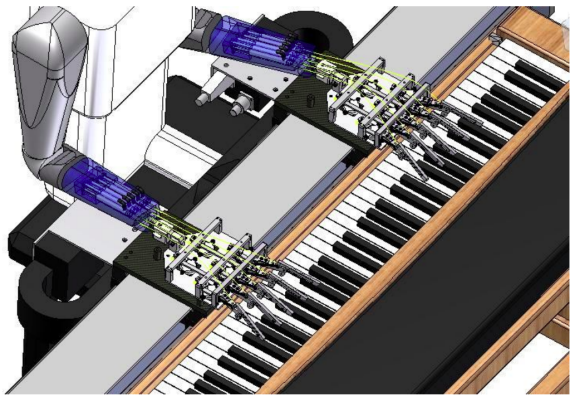


Fig. 1: Robot tocando piano. [2]

En base a esta configuración, cada dedo es capaz de presionar independientemente una tecla de un teclado musical de tamaño real, generando polifonía al activar varios dedos a la vez. Con respecto al movimiento de los dedos, estos son provocados por servomotores, los cuales son controlados por el módulo PCA9685. Además de ello, se utiliza una Raspberry Pi para la comunicación entre el UR5 y los servomotores, y para procesar la canción que se desee interpretar.

II. Metodología

Para realizar una mejor explicación de cómo se desarrolló el proyecto, se realizó un diagrama de bloques (Fig. 2) que representa de modo general los dispositivos que se emplean y cómo será la comunicación entre estos para lograr que el robot UR5 toque el piano. A partir de ello, se dividió el desarrollo del proyecto en cinco etapas: Diseño mecánico de la pseudo-mano, procesamiento de canciones, comunicación entre dispositivos, movimiento de los servomotores y la simulación del robot UR5. A continuación, se presenta y explica detalladamente en qué consiste cada una de estas etapas:

A. Diseño mecánico de la pseudo-mano

La aplicación más común de las manos mecánicas es la de agarrar objetos de manera antropomórfica. Sin embargo, para el desarrollo del presente proyecto, esta característica no es

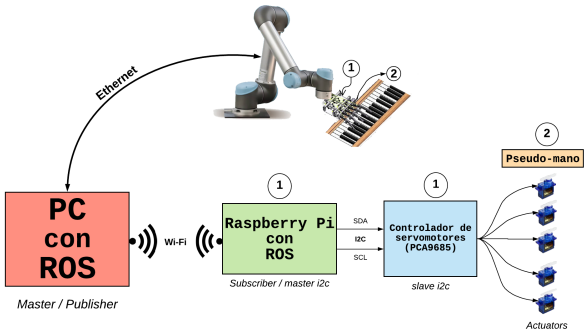


Fig. 2: Diagrama de bloques del proyecto

estrictamente necesaria. Únicamente se requiere que la mano diseñada pueda flexionar los dedos con un cierto grado de libertad de forma que logre presionar las teclas del piano. Este movimiento va a estar limitado por los componentes utilizados como cables, servo-motores y el microcontrolador. En esta sección se explicará cómo fue el proceso de diseño, simulación y modelado, mediante el software Autodesk Inventor, de la mano mecánica.

En internet existen muchos modelos y diseños para la mano mecánica; sin embargo, estos suelen ser muy complejos debido a que buscan imitar todas las posibles funciones de una mano real. Como se mencionó anteriormente, nuestro diseño no debe ser muy complejo ya que la mano mecánica solo debe poder realizar dos acciones: bajar los dedos y subirlos, individualmente o en conjunto. En primer lugar se realizó el diseño de los dedos y se determinó que este movimiento sería realizado mediante servomotores, así que se usó como base la medida de estos (Fig. 3).

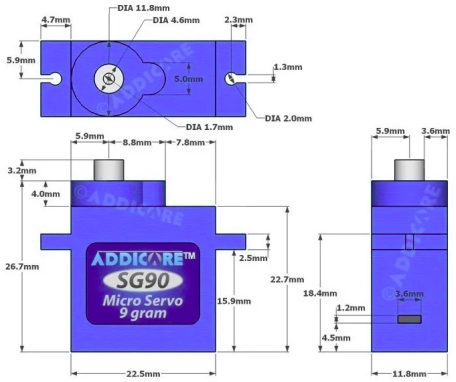
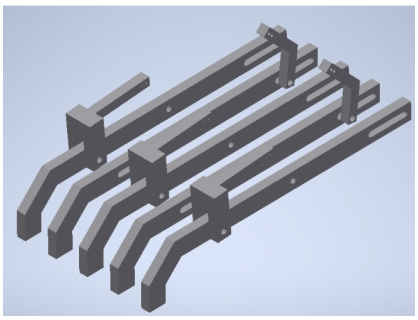


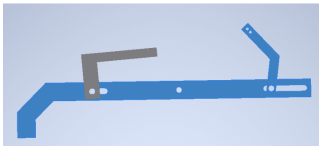
Fig. 3: Medidas de los servomoteres. [3]

También, se tuvo en cuenta la distancia entre las teclas del piano, ya que esta tendría que ser la distancia entre los dedos. No existe una medida estándar para la distancia entre las teclas de un piano, pero las medidas rondan entre los 20 y 23 mm. De esta forma se eligió la medida de 20 mm. No obstante, debido a las dimensiones del servomotor, no se podría colocar de forma contigua y dejar el espacio determinado para los dedos. Debido a esto se realizaron dos configuraciones (Fig. 4) para el movimiento de los dedos. En la primera el servomotor moverá la articulación hacia abajo para mover el dedo, y en la segunda el servomotor deberá mover la articulación hacia arriba para realizar el mismo movimiento. Intercalando ambas configuraciones se logró que los dedos tuvieran una separación de 20 mm sin interferir con el movimiento de los servomotores.



(a) Conjunto de 5 dedos con sus artuculaciones

(b) Configuración 1: dedo-articulación (izquierda)



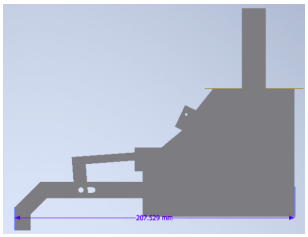
(c) Configuración 1 dedo-articulación (derecha)

Fig. 4: Configuraciones de la pseudo-mano

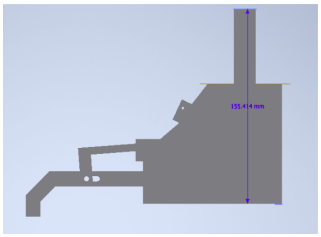
En segundo lugar, se diseñó el soporte de los dedos, el soporte de los servomotores y la unión con el brazo robótico UR5. Para ello, se utilizó un eje circular que pasaba a través de todos los dedos, este serviría como eje de giro para estos. Se utilizaron dos placas con soportes en las posiciones en las que debían ir los servomotores, las cuales se colocaron una sobre otra a cierta distancia para no interferir con el movimiento. La mano mecánica sería sujetada por el UR5 mediante un gripper, por lo que se colocó una parte sobresaliente para sujetarla. Asimismo, se determinó que el UR5 debía sujetar la mano desde la parte superior debido a que facilitaba su movimiento. En base a todo ello, las dimensiones finales del diseño de la pseudo-mano mecánica fueron de 144 mm de ancho, 155 mm de alto y 208 mm de largo. Estas dimensiones se muestran en la Fig. 5.

B. Procesamiento de canciones

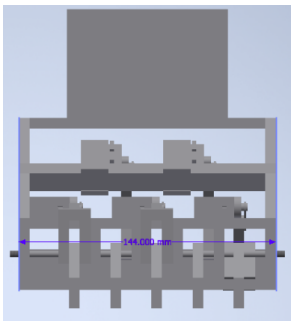
Para el presente proyecto se requiere que la mano mecánica junto al brazo robótico puedan tocar canciones y/o melodías relativamente complejas con el piano. Para esto, se tuvo que diseñar un código que permita a los dispositivos interpretar las notas musicales de archivos de música y transformarlas en instrucciones de movimiento para el robot y la pseudo-mano. Se utilizó el formato MIDI en los archivos de música para su procesamiento, los software de programación Matlab y Pycharm para escribir los código y la aplicación Synthesia para realizar simulaciones de piano. En esta sección explicaremos



(a) Largo



(b) Alto



(c) Ancho

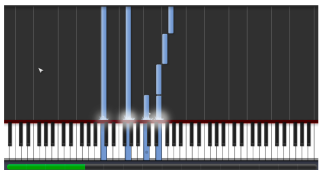
Fig. 5: Dimensiones del diseño de la pseudo-mano

el procedimiento de diseño y codificación del programa para realizar esta interpretación de los archivos musicales y de los software utilizados.

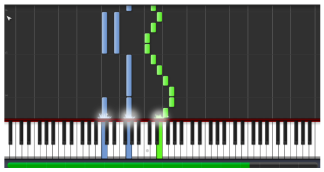
MIDI (Musical Instrument Digital Interface en inglés) es un formato de archivos musicales que contienen protocolos, interfaces digitales y conectores que permiten que varios instrumentos musicales electrónicos, ordenadores y otros dispositivos relacionados se conecten y comuniquen entre sí [4].La ventaja de los archivos MIDI es que se puede extraer las notas musicales utilizadas y los tiempos en las que son activadas para diferentes instrumentos, entre ellos el piano. Sin embargo, hay algunos archivos MIDI que no contienen toda su información, debido a que no fueron programados correctamente o fueron convertidos a partir de otro tipo de archivo. Debido a esto, utilizaremos el programa Synthesia para comprobar que el MIDI contenga la información necesaria para ser trabajada. Lo más importante que se busca, es diferenciar las notas de armonía y melodía, debido a que el robot sólo podrá interpretar una de estas dos. Los archivos MIDI que no contengan esta característica serán automáticamente descartados.Un ejemplo de cada uno de estos archivos se muestra en la Fig 6.

Aplicando el software Matlab procesamos la información de los archivos utilizando las funciones desarrolladas por [5]. Mediante esta se extraen 8 columnas que contienen la información de los archivos MIDI mostrada en la Tabla 1.

Las notas musicales de la melodía y la armonía son diferenciadas a partir del número de canal. Luego de filtrar la información del canal determinado, esta es almacenada en un arreglo que es trasladado a un archivo csv para poder ser trabajado en Pycharm.



(a) Archivo que **no** diferencia la melodía de la armonía



(b) Archivo que diferencia la melodía de la armonía

Fig. 6: Diferencia entre archivos MIDI

TABLE I: Información extraida de los archivos MIDI

Información	**Descripción**
Número de pista	Permite diferenciar conjuntos de eventos MIDI.
Número de canal	Permite diferenciar a qué dispositivo está dirigida la información de los eventos MIDI. Una simple conexión MIDI puede transmitir hasta dieciséis canales de informaciónque pueden ser conectados a diferentes dispositivos cada uno [4].
Número de nota	Las notas musicales se representan en MIDI mediante un número entero positivo comprendido entre 0 y 127 (requiere de 7 bits). Con este rango cubre 11 octavas. [6]
Velocidad	Representa numéricamente qué tan rápido se tocó o soltó una nota, sirve para representaciones gráficas. El valor predeterminado es 64.
Tiempo de inicio (s)	Segundo en el que se realiza la acción del evento MIDI.
Tiempo de finalización (s)	Segundo en que se deja de realizar la acción del evento MIDI.
Número de mensaje de note_on	Indica que el evento MIDI es la acción de tocar la nota
Número de mensaje de note_off	Indica que el evento MIDI es la acción de dejar de tocar la nota.

Para diseñar la lógica del programa se buscó recrear el "pianoroll" de Synthesia. Es decir, la representación gráfica del archivo MIDI. El primer problema era el cómo separar las notas, pues en conjunto no tienen la misma duración y a veces se cruzaban. Debido a esto, se optó por separar las notas por cada milisegundo. Asimismo, se determinó crear un arreglo con 128 columnas, las cuales representan el número de notas codificadas por MIDI, y un número de filas igual al número de milisegundos que dure la canción. Las teclas presionadas fueron representadas por un "1" y las teclas no presionadas fueron representadas por un "0". Con esta información se puede enviar instrucciones de movimiento al UR5 y a los servomotores.

A partir del arreglo de las teclas se realizan algunos filtros por las limitaciones de la mano mecánica. Primero, se limitarán las octavas que pueda tocar el piano (solo 8), debido a que la mayoría de los piano no toca más. Entonces se limitan las teclas del 12 al 119, quedando 108 columnas en el arreglo. Segundo, no se pueden tocar las teclas negras por lo que estas serán eliminadas del arreglo quedando solo 63 columnas. A

partir de este arreglo se agrupan 5 teclas consecutivas donde al menos hay una tecla presionada, con esto se busca la posición más óptima para el UR5 y la mano mecánica. La posición inicial será la primera tecla de la cuarta octava (columna 28) que es la nota 60. La posición se guiará con el dedo pulgar de la mano mecánica (representando la mano derecha).

Finalmente, la información se envió en un arreglo de 3 columnas con la posición, servos activados, y la duración de la nota. La posición será 0 cuando el pulgar de la mano mecánica se encuentre en la nota 60 y aumentará o disminuirá si el brazo se mueve a la derecha o a la izquierda respectivamente. Los servos activados son representados por un arreglo de 5 elementos donde "1" significa activo y "0" inactivo. La duración de la nota se encuentra en segundos e indicará cuánto tiempo se realizará la acción de la fila. En la Fig. 9 . muestra una representación gráfica de la interpretación del movimiento a partir del código.

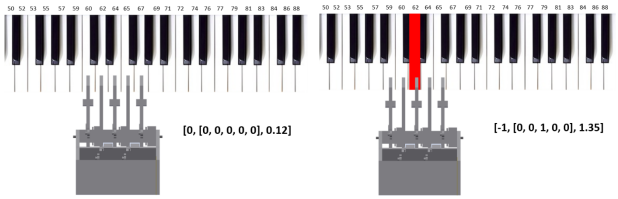
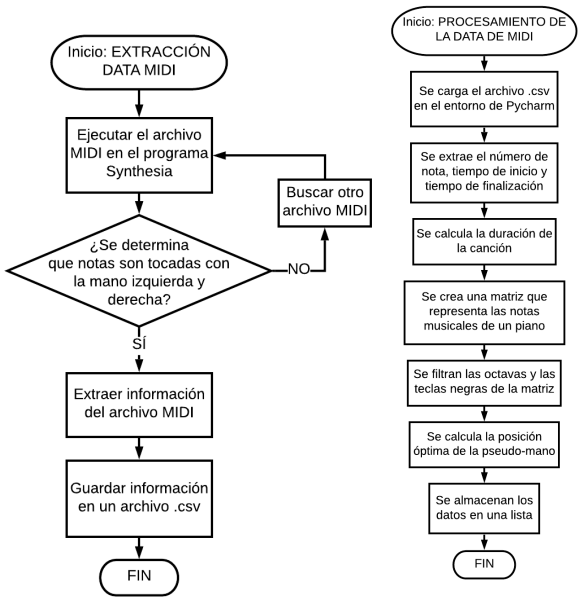


Fig. 7: Representación gráfica de la interpretación del movimiento a partir del código

Asimismo, en la Fig. 8 se muestra el diagrama de flujo para la extracción de la data del MIDI y su procesamiento.



(a) Extracción de data MIDI

(b) Procesamiento de la data MIDI

Fig. 8: diagrama de flujo para la extracción de la data del MIDI y su procesamiento

C. Comunicación entre dispositivos

Como se observa en la Fig. 2 hay tres dispositivos principales que se encargan de que la pseudo-mano pueda moverse y tocar el piano. Estos dispositivos son: Computadora (Master), la Raspberry Pi y el controlador de servomotores. El primero de ellos tiene como objetivo ejecutar el programa para realizar la simulación en Gazebo. Además, debe ejecutar el código principal que procesa las notas de la canción y enviar los datos generados al segundo dispositivo. Luego de recibir los datos, la Raspberry Pi tiene como finalidad procesarlos y ejecutar el código (en Python) que brinde los comandos necesarios al controlador para generar el movimiento de los servomotores. Sin embargo, para que cada uno de estos dispositivos cumpla con su función se debe establecer una comunicación entre ellos. Para ello, se trabajará en la plataforma ROS.

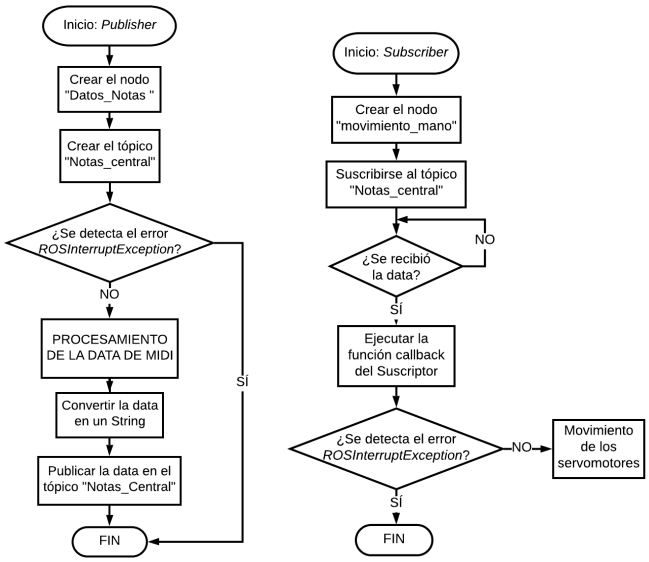
ROS, *Robot Operating System*, es una plataforma de desarrollo de aplicaciones en robótica que proporciona características como comunicación de mensajes, administración de paquetes, computación distribuida, entre otros [7]. En este proyecto se utiliza la característica de comunicación de mensajes para enviar los datos del Master a la Raspberry Pi. Asimismo, se enviarán los comandos para la simulación en Gazebo por medio de ROS. En esta sección nos centraremos en explicar cómo se realizó la comunicación entre el Master y la Raspberry Pi.

Para realizar la comunicación entre los dos dispositivos requeridos se emplearán los conceptos de nodes, topics, publisher y subscriber. En base a [7], se explican dichos conceptos:

- *Nodes*: Un *node* se refiere a la unidad más pequeña de procesador que funciona en ROS, el cual se puede interpretar como un programa ejecutable. ROS recomienda crear un solo nodo para cada propósito, con el fin de desarrollar una fácil reutilización. Cabe mencionar, que la base de la comunicación en ROS se da entre nodos.
- *Publisher*: El *publisher* se encarga de crear un nodo publicador que enviará mensajes (data) a un topic.
- *Subscriber*: El *subscriber* es quien recibe la data enviada por el publisher a través de un topic. Para ello, se debe crear un nodo suscriptor.
- *Topics*: Un *topic* se puede interpretar como un tema de conversación, pues en ROS un topic es el espacio donde un publisher y un subscriber intercambian data.

En base a estos conceptos, en el presente proyecto se creará un nodo publicador que se encargará de enviar la data relacionada a las notas que se deben tocar y los servomotores que deben activarse. Este nodo publicador se implementará en el código que procesa la data del archivo .csv y será ejecutado por el Master, el cual actuará como publisher. Asimismo, se creará un nodo suscriptor el cual recibirá la data. Este nodo será implementado en el archivo que enviará los comandos al controlador y será ejecutado por la Raspberry Pi, la cual actuará como un subscriber. Los diagramas de flujo de los códigos donde se implementan el publisher y subscriber se

muestran en la Fig. 9. De este modo, queda establecida la comunicación entre nuestros dispositivos



(a) Publisher (b) Suscriber

Fig. 9: Diagrama de flujo del publicador y suscriptor

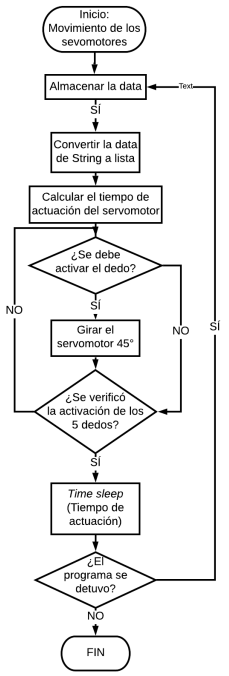


Fig. 10: Diagrama de flujo del código que genera el movimiento del servomotor

D. Movimiento de los servomotores

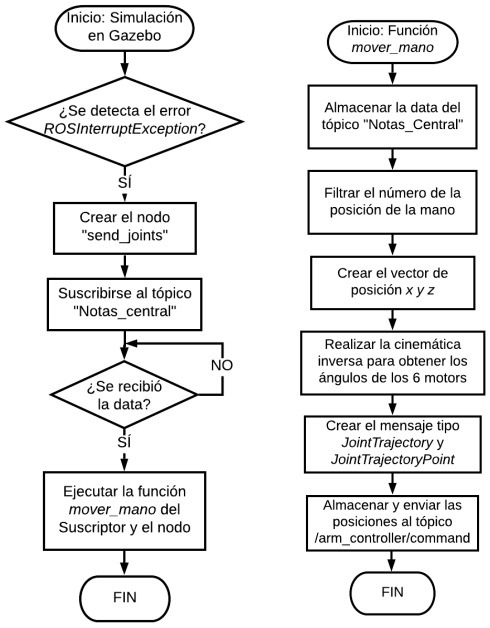
Como se ha mencionado anteriormente la pseudo-mano que se diseñó consta de cinco dedos y cada uno posee un grado de libertad. Por ello, se utilizaron cinco servomotores para realizar el movimiento de cada uno de los dedos. Asimismo, se empleó el módulo PCA9685 para controlar los servomotores, el cual recibe los comandos de movimiento por medio de la Raspberry Pi. En la Fig. 10 se muestra el diagrama de flujo que muestra cómo funciona el código que genera el movimiento del servomotor. Esto es relativamente sencillo, pues solo consiste en indicar que dedos deben activarse y girar el servomotor 45° en caso se activen.

E. Simulación del robot UR5

Para sincronizar el movimiento del UR5 junto con el movimiento de los servomotores se usó el programa Gazebo, el cual permite visualizar una simulación del robot real. A esta se le envían las posiciones x, y y z que se le enviarían al UR5 real y utilizando la cinemática inversa se determinan los ángulos de cada uno de los 6 motores del robot para lograr que el final llegue a la posición deseada. Se utilizó el repositorio de GitHub de Universal Robots [8] en el que se encuentran los archivos URDF y LAUNCH en el que se encuentran toda la información de los links del robot, como la inercia, dimensiones, etc. Asimismo también brinda los controladores necesarios para que cada motor se mueva de manera adecuada al momento de recibir los ángulos correspondientes. También se utilizó la función de cinemática inversa InvKine para el UR5 para Python que se extrajo de [9].

Para la comunicación entre los diferentes programas se utilizó los nodos y tópicos que facilita ROS. El código crea un nodo llamado “send_joints”, el cual será el encargado de mandar los ángulos a la simulación del UR5 para que se mueva en Gazebo. Este nodo se suscribe al tópico llamado /Notas_Central en el que se recibe el String que se generó en el código de Python de la interpretación de la canción. De este String se filtra el primer número, el cual indica la posición del pulgar en el teclado. Con este número se define la posición en el plano xyz al que tiene que ir la mano que sostiene el UR5. Las posiciones en el eje x y z se escogieron de manera arbitraria y son constantes. En cambio, la posición en el eje y cambia con respecto al número que se filtra. Esta toma valores que son múltiplos de 0.02 ya que cada tecla del teclado tiene una separación de 20 mm aproximadamente. Una vez definidas las posiciones se usa la función InvKine para obtener los ángulos de los motores. Cabe resaltar que esta función devuelve una matriz de 6x8 en el que cada columna son 6 ángulos posibles que se pueden usar para que el robot llegue a la posición deseada. Después de probar las 8 columnas se decidió que se usaría la segunda columna de ángulos. Se crean los tipos de mensaje JointTrajectory y JointTrajectoryPoint en el que se guarda la información de los ángulos. Finalmente, se publica el mensaje al tópico del controlador de la simulación llamado /arm_controller/command. Todo el proceso mencionado se resumen en el siguiente diagrama de flujo de la Fig. 11:

Los pasos para correr de manera correcta la simulación es la siguiente. Primero se debe correr el archivo LAUNCH



(a) Simulación en Gazebo (b) Función *mover_mano*

Fig. 11: Diagrama de flujo de la simulación en Gazebo y la *función mover_mano*

llamado ur5.launch. De esta manera se abre el programa de Gazebo con el UR5 listo para recibir posiciones. Después se debe inicializar el nodo "send_joints", el cual espera a recibir posiciones para mandarle los ángulos al UR5 de Gazebo. Por último se debe inicializar el nodo "Datos_notas", el cual genera el String de la posición de la mano y movimiento de los servomotores y lo publica en el tópico /Notas_central. De esta manera la información comienza a circular y la simulación del robot en Gazebo se comienza a mover.

III. Resultados

A partir de la metodología descrita anteriormente se realizó el diseño mecánico de la pseudo-mano, el código en MATLAB para obtener las notas y tiempos necesarios de la canción que tocará el robot UR5, el código que procesa la canción y envía la data al controlador y, finalmente, el código para realizar el movimiento de los servomotores.

En primer lugar, con respecto a la pseudo mano, el diseño final tiene las siguientes dimensiones: 144 mm de ancho, 155 mm de alto y 208 mm de largo. En la Fig. 12. se muestra el diseño 3D en el software Inventor. Debido a las limitaciones del proyecto, no fue posible realizar la impresión 3D, pero se espera que las medidas sean adecuadas y solo se requiera de pequeños ajustes manuales.

En segundo lugar, con respecto al procesamiento de las canciones, se logró realizar un arreglo que sigue la secuencia de las notas de acuerdo a la información almacenada en los archivos MIDI. En la Fig. 13. se muestra la secuencia de notas en el programa Synthesia de un archivo MIDI y el arreglo de

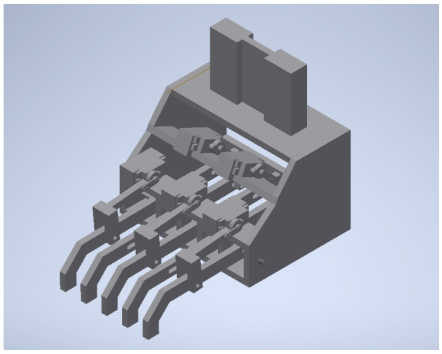


Fig. 12: Diseño 3D de la pseudo-mano

notas conseguido con Python. Se observa que el arreglo sigue las notas dejando una fila de cero. Esto se realizó para que el brazo tuviera tiempo de posicionarse si es que lo necesitara.

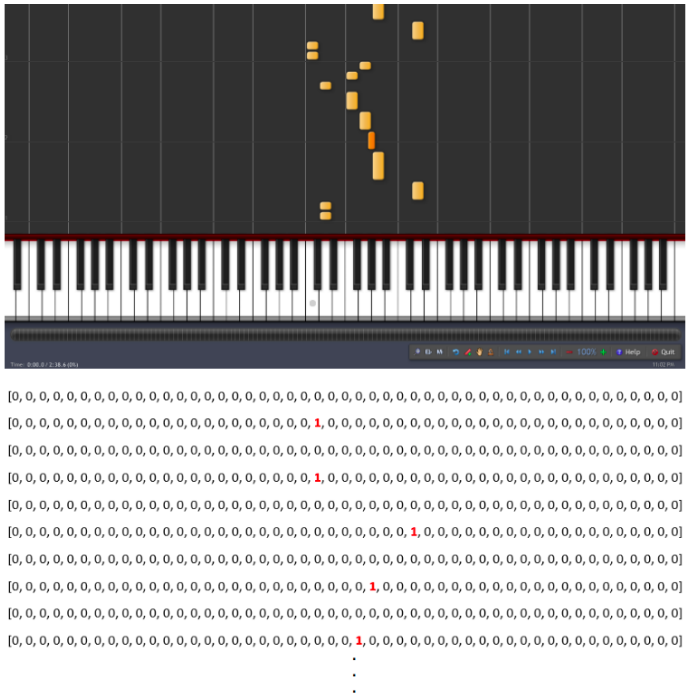


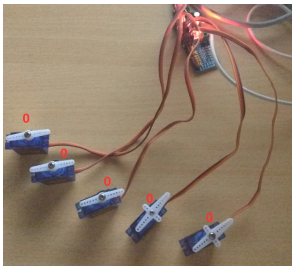
Fig. 13: Secuencia de notas en el programa Synthesia y su representación en un arreglo

En tercer lugar, con respecto a la comunicación entre los dispositivos, se logró comunicar la computadora (Master) y la Raspberry Pi exportando sus IP y declarando a la computadora como el dispositivo Master. Asimismo, se creó el nodo publicador y se envió la data al tópico correspondiente. Además, se creó el nodo suscriptor, el cual recibe la data, y los nodos correspondientes a la simulación en Gazebo. En la Fig. 16 se muestran los tópicos de color rojo, los nodos relacionados a gazebo de color azul, el nodo relacionado al movimiento de los servomotores de color a marillo y el nodo publicador de color verde. Esta gráfica fue obtenida por medio del comando rqt_graph. Esta gráfica nos asegura que la comunicación entre

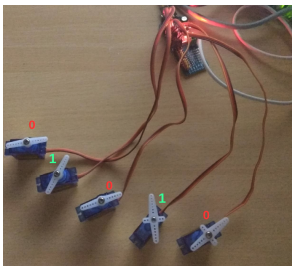
dispositivos se está realizando de manera correcta.

Fig. 14: Representación gráfica de nodos y tópicos

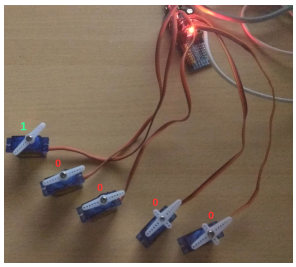
En cuarto lugar, se logró realizar el movimiento de los servomotores en base a la data recibida por el nodo publicador. En la Fig. 15 se muestra una secuencia en la que se activan algunos servomotores (giran). Un '0' indica que el servomotor no ha sido activado, por el contrario, un '1' indica que sí.



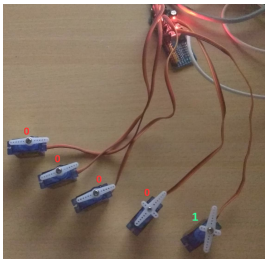
(a) Ningún servomotor activado



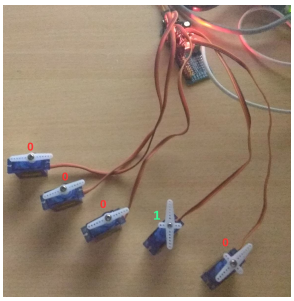
(b) Servomotores 2 y 4 activados



(c) Servomotor 1 activado



(d) Servomotor 5 activado



(e) Servomotor 4 activado

Fig. 15: Activación de servomotores (Del 1 al 5 de izquierda a derecha)

Finalmente, con respecto a la simulación en Gazebo, en la Fig X se muestra cómo el robot UR5 se desplaza desde el punto A al punto C, pasando por el punto B. Este es el movimiento que realiza el robot al trasladarse de una tecla a otra y corresponde a la canción Oda a la Alegría.

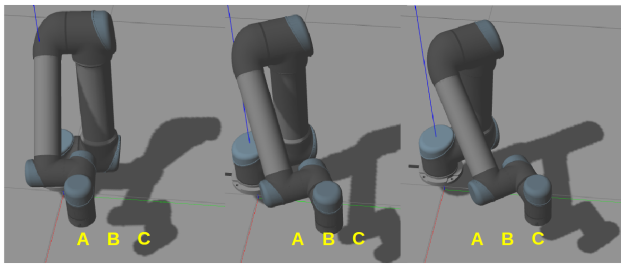


Fig. 16: Simulación del movimiento del robot UR5 desde el punto A al C

IV. Implementación Real

La implementación real del diseño requiere realizar actividades extra a las mencionadas anteriormente en la metodología. Esto con la finalidad de comprobar experimentalmente la sincronización de todo el sistema en conjunto, debido a que por las limitaciones del proyecto no se han testeado todas las partes del sistema. A pesar de que el diseño se ha verificado virtualmente, la implementación física requiere de ajustes experimentales, especialmente en la parte mecánica. A continuación, se describirán las etapas necesarias para la implementación real del proyecto.

En primer lugar, la impresión del diseño mecánico de la pseudo-mano implica la posibilidad de realizar ajustes, ya que el resultado de la impresión no es exacta al diseño. Además, es posible que los elementos que van conectados al diseño, como los servomotores, tarjetas y tornillos, no se acoplen perfectamente al diseño, por lo que se necesitaría realizar un diseño con nuevas medidas o simplemente de un ajuste manual. Asimismo, la comunicación entre el master, es decir, la computadora o laptop con el software ROS y el robot UR5 mediante un patch cord para Ethernet. Esta actividad implica la verificación de la comunicación bajo las mismas direcciones IP.

Por otro lado, los servomotores deben ser probados para asegurarse que los valores de ancho de pulso empleados sean los necesarios para los ángulos que deseamos. Si bien se considera para el diseño un ángulo de 45° para tocar cada tecla de piano, existe la posibilidad que de acuerdo a las pruebas en campo sea necesario modificar este valor y que sea diferente en cada dedo. Además de ello, existe la posibilidad de modificar en el código la velocidad y la fuerza de manera que no lleguen a ser muy lentas o rápidas para la canción que se requiera y, en consecuencia, afecten la melodía. Finalmente, la verificación y ajuste de los movimientos del UR5 en base al software Gazebo. En esta parte la concordancia entre la simulación del software Gazebo y la implementación real es más exacta, sin embargo, es posible que sea necesario probar y ajustar el movimiento del UR5 respecto a todos los demás componentes para obtener una mejor sincronización.

V. Conclusiones

Según lo abordado con anterioridad, se logró diseñar e implementar de manera virtual una pseudo-mano con 5 dedos con el objetivo de aumentar la capacidad del robot UR5 de tocar el teclado al tocar melodías de complejidad media. Se ha presentado la metodología llevada a cabo para el proyecto, la cual consiste en diferentes fases que se desarrollaron para cumplir con el objetivo planteado.

Por un lado, en la primera fase se enfocó en el diseño mecánico de la pseudo-mano en el software Autodesk Inventor, en donde se definió sus dimensiones físicas y las conexiones entre las distintas piezas, así como el movimiento de los dedos. En la segunda fase se explicó cómo se interpretan las canciones para poder brindar al robot la posición a la que debe ir y qué dedos mover. Esto se logró a través del uso de archivos MIDI y softwares de programación para procesar una melodía y convertirla en data para el movimiento del robot y de la mano. En la tercera fase del proyecto se explicó la comunicación entre los dispositivos utilizados, los cuales fueron una computadora, una Raspberry Pi y el controlador de servomotores. En esta sección se introdujo el concepto de ROS y cómo se usan los nodos y tópicos para realizar la transferencia de datos.

Por otro lado, en la cuarta fase se centró en la parte electrónica del proyecto en la que se usó el controlador de servomotores y la Raspberry Pi para mover los dedos con la data de la canción interpretada. En la quinta y última fase se introduce la posibilidad de simular en Gazebo el movimiento del robot UR5, se utilizó posiciones en un plano xyz y la cinemática inversa para definir los ángulos de rotación para cada uno de los 6 motores del robot. De esta manera, gracias a las posibilidades que ofrecen los tópicos de ROS, se consiguió sincronizar el movimiento de la simulación del UR5 con el movimiento de los servomotores. No obstante para concluir el proyecto por completo haría falta realizar la implementación real de toda la metodología en el robot real, la cual debido a las limitaciones del proyecto no pudo realizarse.

Para una futura implementación se recomienda realizar ajustes en la impresión 3D de la pseudo-mano para acoplar los componentes; ajustar los parámetros de la programación de los servomotores, así como su velocidad y ángulos de giro; redefinir las posiciones xyz del UR5 a las posiciones reales de las teclas del teclado e indicar correctamente los tópicos de ROS en los códigos de programación.

References

[1] L. Jen-Chang, H. Li, K. Huang and S. Lin, "Design of Piano -playing Robotic Hand", IAES International Journal of Robotics and Automation (IJRA), vol. 3, no. 2, 2014. Disponible en: https://www.researchgate.net/publication/270723392_Design_of_Piano_-playing_Robotic_Hand

[2] J. Lin, H. Huang, Y. Li, J. Tai and L. Liu, "Electronic piano playing robot", 2010 International Symposium on Computer, Communication, Control and Automation (3CA), 2010. Disponible en: https://ieeexplore.ieee.org/abstract/document/5533457

[3] "Mercado Libre Colombia - Sermovotor SG90", Articulo.mercadolibre.com.co, 2020. [En línea]. Disponible en: https://bit.ly/3a0aaKx

[4] "An Introduction to MIDI", Internet Archive Wayback Machine, 2012. [En Línea]. Disponible en: https://web.archive.org/web/20120830211425/http://www.doc.ic.ac.uk/~nd/surprise_97/journal/vol1/aps2/

[5] K. Schutte, "MATLAB and MIDI", Kenschutte.com, 2012. [En línea]. [En Línea]. Disponible en: https://kenschutte.com/midi#Files

[6] A. López, "Apéndice: Formato MIDI", Lpi.tel.uva.es. [En línea]. Disponible en: https://www.lpi.tel.uva.es/~nacho/docencia/ing_ond_1/trabajos_01_02/formatos_audio_digital/html/midiformat.htm

[7] Y. Pyo, H. Cho, L. Jung and D. Lim, ROS Robot Programming (English). ROBOTIS, 2017.

[8] "ros-industrial/universal_robot", GitHub, 2020. [En línea]. Disponible en: https://github.com/ros-industrial/universal_robot

[9] "mc-capolei/python-Universal-robot-kinematics", GitHub, 2020. [En línea]. Disponible en: https://github.com/mc-capolei/python-Universal-robot-kinematics