Plataforma Virtual e Interactiva para la Conducción de Vehículo Basado en Modelo 3D de Unreal Engine Mediante Transmisión de Datos TCP/IP

1st Erick Gómez López
Facultad de Ciencias de la Electrónica.
Benemérita Universidad Autónoma de Puebla
Puebla, México
erick.gomezlo@alumno.buap.mx

2nd Roberto Carlos Ambrosio Lázaro
Facultad de Ciencias de la Electrónica.
Benemérita Universidad Autónoma de Puebla
Puebla, México
roberto.ambrosio@correo.buap.mx

3th Ivan Cañedo Farfan
Facultad de Ciencias de la Electrónica.
Benemérita Universidad Autónoma de Puebla
Puebla, México
ivan.canedo@alumno.buap.mx

***Abstract*—En el siguiente trabajo se presenta el desarrollo de una plataforma virtual e interactiva para la conducción de un vehículo. Se establece una comunicación de datos entre LabVIEW y Unreal Engine 4, para la adquisición y envió de información al modelo virtual 3D de un vehículo dentro de un escenario similar a un circuito de conducción. Esta manera de transmitir los datos entre ambos software descarta el uso de archivos dll(Dynamic Link Library), en su lugar se utiliza hardware para lograr un transmisión mediante el protocolo de red TCP/IP, y una conexión serial. El hardware utilizado se basa en el dispositivo SoC(System on Chip) ESP32. El vehículo y el escenario son diseñados y visualizados a través del motor de videojuegos, mientras que el mando de velocidad y dirección del vehículo es gestionado desde el software de automatización y control.**

***Index Terms*—Conducción de vehículo, virtual, Protocolo de red TCP/IP, ESP32, Unreal Engine 4, LabVIEW, Transmisión de datos.**

I. Introducción

Un simulador de manejo es una herramienta que recrea, en un contexto artificial, la situación de conducción de un vehículo[1]; los simuladores de manejo son herramientas fundamentales para estudios regionales, nacionales e internacionales que buscan probar la eficiencia de programas de entrenamiento para conductores, aptitudes para manejar en pacientes con distintos trastornos, efectos en la capacidad de conducción de los individuos que se encuentran bajo los efectos de algún medicamento, impacto de señalamientos en el comportamiento del conductor, analizar estrategias con el fin de disminuir los accidentes, y validar ventajas y desventajas de nuevas tecnologías introducidas al vehículo [2][3]. En el campo de la investigación, los trabajos más comunes sobre simuladores de manejo son principalmente dirigidos hacia el software, varias universidades han realizado sus propias versiones basadas en motores gráficos como Unity, Unreal Engine, etc. En su desarrollo han dejado de lado el diseño de los estimuladores y la comunicación vía remota [4][5][6][7][8].

Por otro lado, un software con enormes capacidades de desarrollo en áreas de control, automatización e instrumentación electrónica entre muchas otras, puede en ocasiones encontrarse muy limitado al realizar animaciones en 3D, puesto que no esta enfocado y especializado en estas tareas. Si se logra una colaboración entre este tipo de software y otro especializado en animaciones 3D, el universo de posibilidades se incrementa. De manera general, cuando pensamos en utilizar dos software distintos que trabajen en conjunto, lo primero en mente es utilizar algún complemento que permita conectar de manera sencilla a ambos. Para lograr la colaboración, la manera mas fundamental sería desarrollar una gestión de archivos dll(Dynamic Link Library) para vínculos dinámicos entre ambos software, lo cual implicaría entre otras cosas, la necesidad de conocimientos avanzados en este tipo de archivos, un amplio panorama de la arquitectura de los software con los que se desea trabajar y un mayor tiempo de desarrollo[9]. Por otro lado existe la posibilidad de adaptar herramientas de software y complementos disponibles para otras finalidades, aprovechando sus características y logrando que trabajen de la manera en que lo necesitamos, lo cual brinda una posibilidad mas rápida para resolver la necesidad, y representa un nivel de desarrollo de software menos exigente.

En este trabajo establecemos una comunicación de datos entre LabVIEW y Unreal Engine 4, logrando que ambos funcionen de manera colaborativa para conseguir conducir un vehículo dentro de un entorno virtual. Tanto la dirección de giro, como el avance frontal, son adquiridos desde LabVIEW, por medio de un mando de Xbox One compatible con windows, posteriormente son transmitidos a la red a través del protocolo TCP/IP. LabVIEW es una plataforma de programación gráfica desarrollada por National Instruments para el diseño de sistemas de pruebas y mediciones, el cual debido a su naturaleza gráfica hace que tareas como la vinculación y transferencia de datos con hardware externo a la PC sean mas sencillas[10].

La animación en 3D del vehículo como de su entorno son diseñados y renderizados desde Unreal Engine 4(UE4),

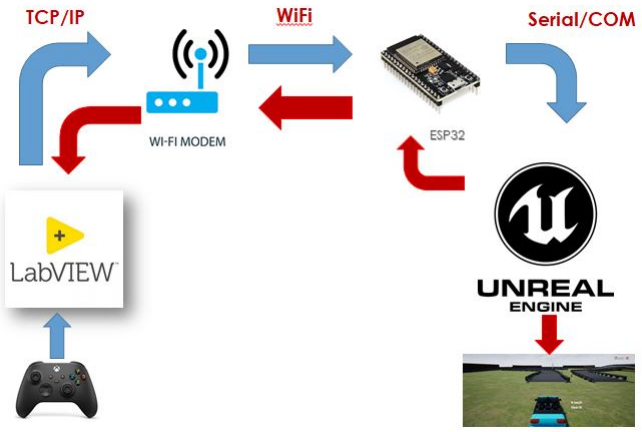

Fig. 1. Flujo de transmisión y recepción de datos a través del hardware para la comunicación entre los dos software.

un motor y editor de videojuegos gratuito desarrollado por la empresa Epic Games, utilizado para la producción de distintos tipos de videojuegos y aplicaciones, el cual tiene la posibilidad de desarrollar proyectos basados en C++, o como es el caso de este trabajo, proyectos basados en blueprints los cuales utilizan un ambiente de scripts visuales, en los que se programan funciones de una manera similar a la programación por bloques[11]. Parte de la comunicación de datos esta apoyada en el uso de UE4Duino un complemento para UE4, el cual permite establecer una conexión serial con un puerto COM. Cuando los datos son recibidos, UE4 se encarga de ejecutar las acciones necesarias para convertirlos en información y así renderizar la animación que obedece la nueva posición del vehículo.

Con el aumento de las necesidades de realizar actividades de manera remota, una transmisión de datos a través de la red utilizando un protocolo TCP/IP se vuelve bastante conveniente. Este tipo de transmisiones de datos se vuelven cada vez mas sencillas de implementar gracias a los avances que han sucedido en dispositivos pensados para proyectos de IoT, como lo es el desarrollo del ESP32, un SoC que involucra un procesador de doble núcleo, altas prestaciones en sus interfaces periféricas, tecnologías de comunicación WiFi y Bluetooth de modo dual, bajo consumo de energía y un bajo costo[12]. Dentro del trabajo que aquí se presenta, el ESP32 tiene la función de recibir mediante su conexión WiFi los datos provenientes de labVIEW que fueron enviados a través de la red, posteriormente los transmite hacia UE4 por medio de una comunicación serial en un puerto COM a través de un cable USB. El flujo de la transmisión y recepción de datos entre los diferentes elementos del sistema puede visualizarse en la figura 1.

II. Desarrollo

A. Adquisición de datos del mando

La adquisición de datos se realiza con base en los estados que se muestra en la figura 2, para comenzar se conecta el mando de Xbox One a la PC, posteriormente dentro de un proyecto se inserta un ciclo while que permita una ejecución continua, fuera de este ciclo iniciamos el reconocimiento del mando insertando la función *Open* que pertenece al grupo de herramientas *Maker Hub interface for Xbox One controller* y permitimos que permanezca en Auto Detect, dentro del ciclo while se anida un *eventstructur* para reconocer cuando sucede un evento en en mando de Xbox, dentro de esta estructura se agrega la función *Read* y por ultimo, fuera del ciclo infinito se cierra la conexión con el mando insertando la función *Close*, esta disposición de los bloques de funciones se muestra en la figura 3.

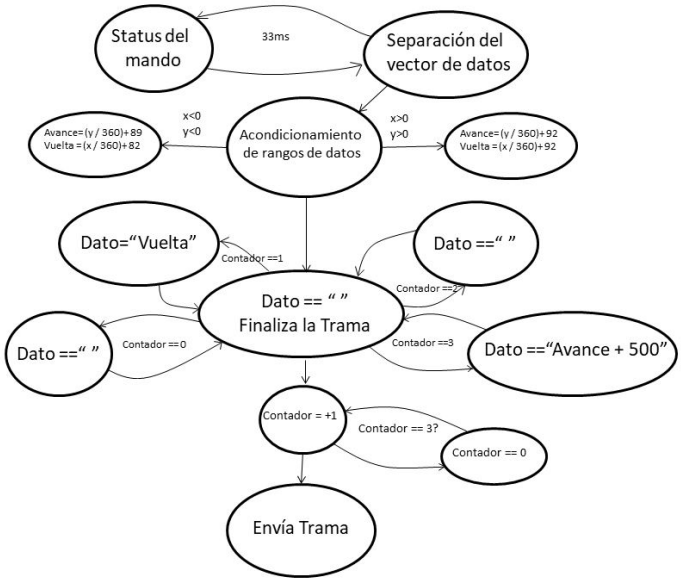

Fig. 2. Diagrama de estados para la adquisición, acondicionamiento y envió de datos.

Las palancas del mando tienen una resolución de 16 bits para cada eje, por lo que los datos de lectura en labVIEW, pueden tomar valores desde -32768 hasta 32768 en cada uno de los ejes y son entregados por medio de un vector de datos que incluye todos los valores de posición, de las palancas de mando, gatillos y D-pad. Se deben separar los datos del vector y deben ser convertidos en información útil. Así que tal y como se observa en la figura 3, al vector de datos se le aplica un primer bloque de función *Unbundle*, lo que separa los datos de la palanca de mando izquierda, un segundo *Unbundle* entrega los datos separados del eje x y el eje y, los cuales brindaran el giro y la aceleración respectivamente. Para obtener la información que será enviada, convertimos los datos del rango de 16 bits a un rango simple de 1 a 180, esto lo logramos con la ayuda de un bloque de función *Case structure*, el cual separa los numero negativos de los

numero positivos, los cuales dividimos entre 360, sumamos la cantidad de 89 para los datos negativos y 92 para los datos positivos, de esta manera y repitiéndolo para cada eje, obtenemos los valores que se envían. Cada eje obtendrá un rango de valores diferentes en labVIEW y y serán utilizados para ser transmitidos hacia UE4(figura 4).

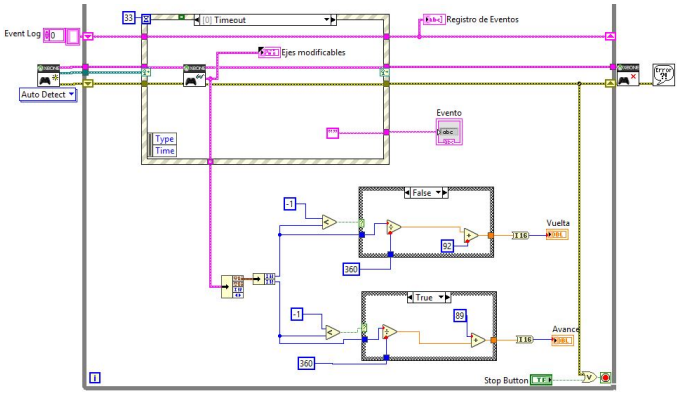

Fig. 3. Disposición de bloques de función para el inicio y fin de la conexión con el mando de Xbox one.

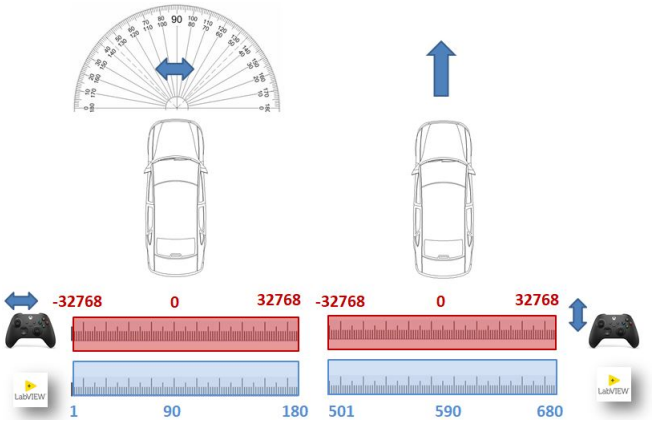

Fig. 4. El eje x de la palanca de mando izquierda brindara la dirección del vehículo, el eje y la aceleración, los datos del mando son convertidos a un rango útil y diferente cada uno, para ser transmitidos.

B. Transmisión de datos en red mediante protocolo TCP/IP

Para transmitir los datos se utiliza el protocolo TCP/IP, para lo cual desde el proyecto en labVIEW se inicia configurando fuera del ciclo while, un bloque de función *TCP Open connection*, en el cual se coloca la dirección IP (192.168.0.13) asignada por el módem de conexión al dispositivo ESP32, un numero de puerto de servicio (8888) que coincide con el que se elige de manera deliberada en el programa que se carga en el dispositivo ESP32, además un valor de 60000 ms para el timeout. Posterior a esto, se coloca un bloque de función *TCP Write* dentro del ciclo while del proyecto, y se cierra la conexión colocando fuera del ciclo el bloque de función *TCP Close connection*, en la figura 5 se puede observar esta configuración utilizada.

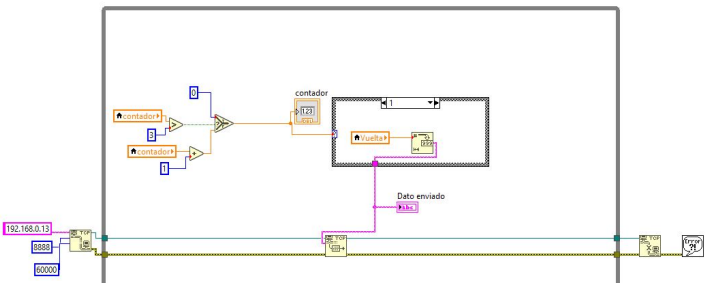

Fig. 5. Disposición de bloques de función para el inicio y fin de la transmisión de datos por protocolo TCP/IP.

La transmisión de datos se realiza cuando Labview da por finalizada una trama, para poder finalizar la trama de información que sera transmitida, es necesario enviar un carácter de espacio vació. Para poder realizar esta tarea se implemento un contador ascendente que va de 0 a 3, a su vez el valor de este contador es conectado a una estructura de control *case structure* de 4 con cuatro casos posibles, de manera que cada ciclo de programa el contador ira variando los casos a ejecutar de una manera secuencial, en los casos 0 y 2, el dato que se envía será un carácter de espacio vació que indicara la finalización de la trama, el caso 1 el dato que se enviara será el valor de la variable local *Vuelta*, en el caso 3 el dato que se envía será el valor de la variable local *Avance* sumando la cantidad de 500, logrando un rango distinto que pueda servir a identificar que datos pertenecen a cada eje. Todo lo anterior se puede apreciar en el diagrama de estados que se muestra en la figura2, la codificación de dicho diagrama corresponde a la disposición de los bloques en la parte superior de la figura 5.

C. Programación de ESP32

Para programar el dispositivo desde el IDE de arduino, es necesario, instalar la biblioteca WiFi.h y el gestor de tarjetas ESP32, una vez realizando lo anterior, procedemos a codificar el algoritmo que se describe en el diagrama de flujo de la figura 6. En la codificación del programa se comienza incluyendo la biblioteca WiFi.h con la que se trabaja, definimos las variables que usaran como también las constantes para el SSID y el Password del módem al que se conectará, de igual manera definimos un puerto de servicio de manera arbitraria que en este caso será 8888, dentro del void setup se configura la conexión serial a 9600 baudios y se inicializa la conexión WiFi. Una vez que el ESP32 se conecta al módem, se le asigna una dirección IP al dispositivo, la cual es necesaria para vincular con LabVIEW, para poder visualizar esta dirección es necesario imprimirla en el monitor serial. Posterior a la configuración inicial el void loop que se estará ejecutando continuamente realiza la tarea de búsqueda del cliente con el que se conectara, una vez

conectado consultara si existe un dato disponible a la entrada, los datos que se reciben se almacenan en una variable, en seguida el valor contenido en esa variable es transmitido por la conexión serial, se detiene la conexión con el cliente y el proceso se vuelve a repetir.

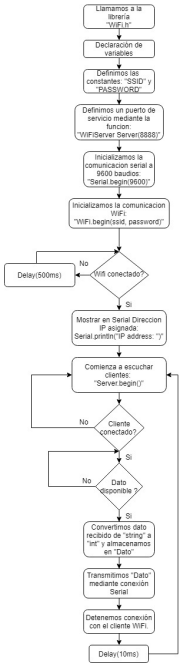

Fig. 6. Disposición de bloques de función para el inicio y fin de la transmisión de datos por protocolo TCP/IP.

D. Comunicación Serial desde UE4

Para realizar la comunicación serial se trabaja con el complemento de distribución libre llamado UE4Duino, el cual permite establecer comunicaciones seriales a través de los puertos COM. Para la instalación de este complemento, se crea una carpeta en el directorio del proyecto la cual albergue los archivos del complemento y se edita el archivo necesario.

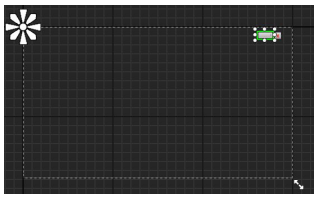

Fig. 7. Diseño del Widget Blueprint que permite al usuario seleccionar el puerto COM.

Una vez instalado el complemento, se crea un *Widget Blueprint* (figura 7) que permita seleccionar el puerto COM por el cual se establecerá la comunicación de los datos, este widget se trata de una ventana flotada sobre el escenario que permite una interfaz con el usuario desde la cual se puede realizar una selección desde un combo box desplegable. La programación de este blueprint se muestra en la figura 8, al existir un evento *On selection*, inicia una rutina para inicializar el puerto COM del numero seleccionado, por otro lado si existe un evento *On Clicked* sobre el boton con la X, la comunicación con el puerto COM debe finalizar y se cierra el juego.

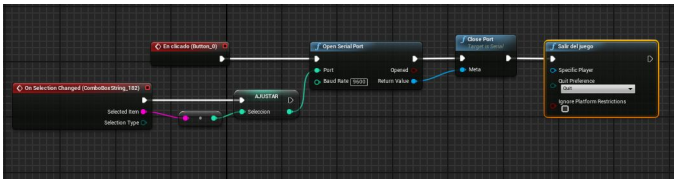

Fig. 8. Programación del Widget Blueprint que permite al usuario seleccionar el puerto COM.

Es necesario programar adecuadamente las tareas que se ejecutaran una vez que se inicia el juego y en que momento se ejecutaran dichas tareas, para esto es importante editar adecuadamente el *Blueprint del nivel* desde el cual se comienza por inicializar todas las variables en un valor de 0, posterior a esto se establece que el widget desde el cual se selecciona el numero de puerto COM se añada al primer plano de la ventana gráfica, además de especificar también que aparezca el cursor del mouse. Lo siguiente es extraer el dato entero seleccionado en el widget para poder inicializar la conexión con el puerto COM adecuado, este dato entero se almacena en una variable llamada *numero puerto*. Ajustado el valor de la variable, se procede a inicializar la conexión con el puerto serial, hacia el cual una vez iniciada la comunicación se envía el valor contenido en la variable salida, después de un breve retardo de 20 ms, se procede a leer los valores que estén siendo recibidos para comenzar a trabajar con ellos. La programación de este Blueprint se muestra en la figura 9.

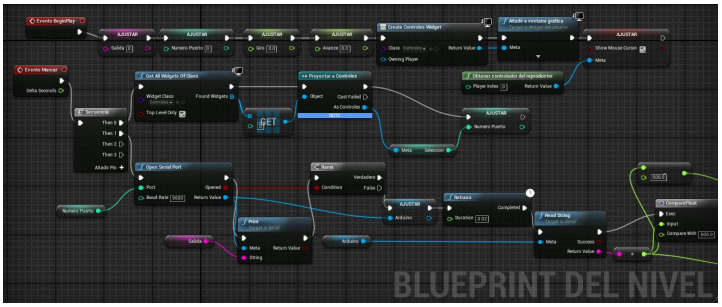

Fig. 9. Programación del Blueprint del nivel.

Una vez que es posible recibir los datos, estos deben ajustarse a la manera en que se trabajara con ellos, lo primero es convertir la cadena en valores de tipo flotante, después de que estos son convertidos se tendrá entonces valores que pertenecerán a dos diferentes rangos, el primero entre 1 y 180 , el segundo entre 501 y 680, entonces se realiza una comparación del valor para distinguir el eje de la palanca de

mando del cual proviene, en el caso del segundo rango, una vez distinguido es posible restar las 500 unidades que se le agregaron para su distinción, hecho esto, se almacenan los datos en sus variables correspondientes y se repite el ciclo. Esto se realiza de la manera que se muestra en la figura 10.

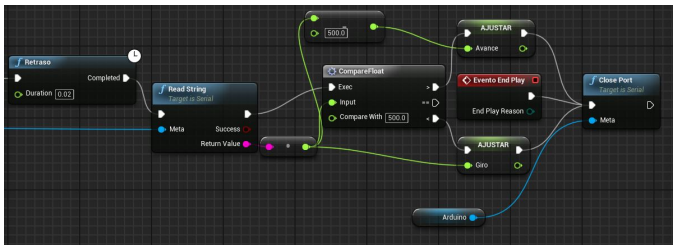

Fig. 10. Distinción de los datos de rangos distintos y su almacenamiento en las variables correspondientes a cada uno.

Los valores de las variables que ya almacenan los datos provenientes desde LabVIEW son utilizadas para dar la dirección al vehículo (figura 11) y la aceleración (figura 12), estas funciones para la entrada del volante y el avance son editadas desde el blueprint del vehículo sedan del proyecto.

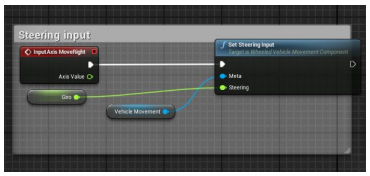

Fig. 11. Entrada para la dirección del volante desde la variable Giro.

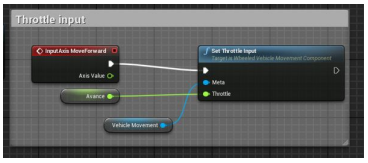

Fig. 12. Entrada para la aceleración del vehículo desde la variable Avance.

E. Diseño de escenario de conducción

El escenario de conducción es estructurado a partir del uso de dos mallas prediseñadas, diferentes texturas y un algoritmo de creación de SPline. Las mallas prediseñadas son las formas básicas para la construcción de una carretera dentro de UE4, tanto la cinta asfáltica como las barras de contención presentan texturas repetitivas que pueden ser replicadas múltiples veces. Con la ayuda de un algoritmo para la creación de un SPline es posible insertar de manera mas fácil estructuras repetitivas como puentes, tuberías y en nuestro caso una carretera, facilitando en gran media el diseño de nuestro entorno de conducción. Las texturas precargadas por el mismo UE4, nos ayudan a dar un entorno mas realista, en este trabajo se agregaron texturas de césped en el suelo y piedra en las paredes que rodean el escenario(figura 13).

Fig. 13. Perspectiva de edición en el diseño para el escenario de conducción.

III. Resultados

Se realizo la adquisición de los datos del mando de Xbox one desde el software de LabVIEW, se acondicionaron los valores de la resolución de 16 bits en rangos de 1 a 180 para el eje X, de 501 a 680 para el y, ambos ejes pertenecientes a la palanca de mando izquierda. La estrategia de utilizar diferentes rangos de números para distinguir entre diferentes ejes de una misma palanca de mando fue adecuada para el envió de la información hacia el software Unreal Engine 4, en la figura 14 es posible observar los datos obtenidos desde el palanca de mando.

Fig. 14. Resultado obtenido al colocar la palanca de mando izquierda situada al extremo derecho en el eje X y en el medio de la altura en el eje Y.

La comunicación de datos a través de protocolo TCP/IP presenta un buen desempeño en la transmisión de información entre ambos software, no se presento un retardo importante entre un movimiento en el mando y la respuesta de la animación en el motor de videojuegos. Además de lograr establecer la comunicación sin el uso de archivos dll, fue posible conducir el vehículo virtual de manera satisfactoria (figuras 15, 16).

Conclusiones

Un trabajo colaborativo entre LabVIEW y Unreal Engine 4, sin el uso de archivos dll y sin complementos específicos para la simulación virtual de la conducción de un vehículo, es completamente posible de realizar, funciona de manera adecuada y es estable, lo cual incrementa en gran medida el universo de posibilidades en animaciones 3D que pueden realizarse en desarrollos de control, automatización entre otros. Parte de los