Control de mini vehículo autónomo utilizando entrenamiento de aprendizaje por refuerzo.

Proyecto Terminal

Por

Ing. Carlos Enrique Coazozón Echevarria

En cumplimiento a los requerimientos para obtener
la Especialidad de Tecnólogo en Mecatrónica

Revisor académico: Dr. Leonardo Barriga Rodríguez

Santiago de Querétaro, Qro., México, Febrero 2020

Índice de contenido

Índice de ilustración

Introducción

Los sistemas automatizados, en este caso los robots móviles o de otro tipo, comúnmente son programados mediante métodos de control automático, dando como resultado, sistemas predispuestos a realizar ciertas tareas en específico, dentro de entornos en los cuales están destinados a operar desde un principio, sin embargo, este tipo de controles van cambiando mediante la inteligencia artificial avanza.

La inteligencia artificial, puede ser utilizada para realizar controles, ya que esta puede definirse como una ciencia para construir máquinas que realicen actividades que, si un humano las llevara a cabo, requeriría de una inteligencia, el objetivo concreto de esta ciencia, es a brindar a las máquinas la capacidad para aprender datos y utilizar lo aprendido en toma de decisiones ante diferentes situaciones tal como lo hace un humano.

Dentro de la inteligencia artificial existe un subcampo llamado aprendizaje automático el cual tiene como fin desarrollar técnicas mediante diferentes algoritmos para que las máquinas puedan aprender, si nos enfocamos en específico el área de la robótica, uno de estos algoritmos se conoce como aprendizaje por refuerzo el cual comúnmente se implementa para el control de robots.

El presente reporte de proyecto, muestra el desarrollo e implementación del control de un mini vehículo utilizando aprendizaje por refuerzo, el cual conlleva el entrenamiento mediante simulaciones creadas en el motor gráfico de Unity, así mismo también requiere la recopilación de datos de su entorno mediante un sensor láser LIDAR 2D de 360° de libertad, todo esto con el fin de implementar controles, donde no sea necesario tener un dominio amplio sobre los métodos de control automático.

Planteamiento del problema

Los sistemas automatizados como brazos robóticos, robots móviles, entro otros, que su control ha sido programado mediante métodos de control automático, operan exclusiva y únicamente para lo que han sido diseñados desde un principio, teniendo como fin desempeñar tareas repetitivas y monótonas dentro de una cierta cantidad de entornos predispuestos.

Destacando lo anteriormente mencionado, programar este tipo de controles sugiere una problemática, debido que realizar la tarea por la cual fue programado, en entornos no predispuestos se torna complejo. En consecuencia, es necesario programar controles muy específicos y tediosos, además cabe mencionar que en muchos casos es necesario probar la eficiencia de dichos controles de manera física en el sistema, limitando la rápida implementación de los mismos.

Como se sabe, los controles programados mediante método de control automático y la manera en cómo se implementan, tienen ciertas limitaciones, por este motivo es necesario indagar en el desarrollo de nuevos controles, en este proyecto se pretende implementar el control de un robot móvil, mediante una alternativa enfocada en el control de robots de diferentes tipos, la cual va de la mano con la inteligencia artificial, esta alternativa se conoce como aprendizaje por refuerzo.

Objetivo

Objetivo general:

En el motor gráfico de Unity entrenar mediante el aprendizaje por refuerzo a un agente simulado, con el fin de implementar en un mini vehículo físico un control autónomo con el aprendizaje obtenido del entrenamiento.

Objetivos específicos:

- En el motor gráfico de Unity diseñar un entorno para simular el mini vehículo.
- Realizar el entrenamiento del mini vehículo simulado mediante aprendizaje por refuerzo.
- Adquirir variables mediante los datos que proporciona el sensor láser en diferentes grados de posición.
- Controlar físicamente el mini vehículo con el aprendizaje obtenido del entrenamiento simulado.

Justificación

En el área de energía de CIDESI se trabaja con diferentes tipos vehículos robóticos, por lo cual surge la necesidad de implementar nuevos controles inteligentes para tener opciones alternativas ante diferentes necesidades, y estos mismos puedan ser simulados previamente, con la finalidad de poder analizar y observar cómo operan en los entornos en los que deseamos utilizarlos.

Antecedentes

Existe una gran cantidad de trabajos implementados mediante Unity con el complemento ML-Agents, por lo cual la mayoría de estos trabajas son relativamente similares, puesto que el desarrollo de estos implementa el aprendizaje por refuerzo con el fin de controlar las decisiones del agente dentro de un entorno simulado en Unity, sin embargo, cabe mencionar algunos trabajos que tiene una mayor aportación a nuestro proyecto planteado.

Uno de esos trabajos se describe en [12], se busca dar inteligencia a los agentes u objetos mediante Machine Learning con técnicas de aprendizaje profundo, en este trabajo se plantearon tres experimentos diferentes, los cuales dos de ellos tienen una relación al proyecto planteado debido que de igual manera, diseñan un entorno virtual en el cual se busca controlar, por medio de aprendizaje por refuerzo, el desplazamiento de un agente dentro de una pista delimitada, así mismo se muestra diferentes métodos de cómo lograrlo.

De igual manera el trabajo [13], explica el entrenamiento de un agente para que aprenda a desplazarse dentro de una pista cerrada, sin chocar con ninguna de las paredes que limitan la pista, este trabajo tiene una aportación importante para nuestro proyecto debido a que el entrenamiento lo prueba de manera físicamente en un vehículo real con sensores ultrasónicos, para obtener las variables de nuestro entorno real, este trabajo es similar al propuesto sin embargo el material que implementaron y la manera en que realizaron, son diferentes.

Los trabajos mencionados no contienen información detallada sobre su desarrollo, aun así, son de gran aportación para nuestro proyecto, puesto que nos da una ideología de cómo estructurar una nuestra propia metodología a seguir, cabe mencionar que estos proyectos no tienen la misma finalidad o no implementan la misma infraestructura que el trabajo que nosotros buscamos desarrollar.

Marco Teorico

Python.

Python es un lenguaje de programación de alto nivel, interpretado y multipropósito, actualmente es uno de los lenguajes de programación más empleados para el desarrollo de software, además tiene la flexibilidad de ser utilizado en diversas plataformas y sistemas operativos, de igual manera con este tipo de lenguaje podemos desarrollar software para aplicaciones científicas, para comunicaciones de red, para aplicaciones con interfaz gráficas y aplicaciones web.

Figura 1.- Logo de Python.

Este lenguaje es potente, flexible, con un sintaxis clara y concisa, además no requiere dedicar tiempo a su compilación debido a que es interpretado, otra característica importante es que Python es "Open Source", es decir cualquiera puede contribuir a su desarrollo y divulgación, además no es necesario pagar ninguna licencia para distribuir software desarrollados con este lenguaje [1].

Anaconda.

Anaconda es una distribución libre y abierta de los lenguajes Python y R, que se utilizan en ciencia de datos y machine learning, se emplea principalmente para procesamiento de grandes volúmenes de información, análisis predictivo y cómputos científicos. Las diferentes versiones de los paquetes se administran mediante el sistema de administración del paquete Conda, cual facilita implementar software de ciencia de datos y machine learning, tales como Scikit-team, TensorFlow y SciPy [2].

Figure 2.- Logo de Anaconda.

Unity.

Unity es un motor gráfico para desarrollo de videojuegos, una suite de herramientas diseñadas para facilitar el trabajo en los distintos aspectos de un videojuego, tales como los gráficos, las variables físicas, las animaciones o la inteligencia artificial. Este motor gráfico dispone de licencia gratuita, totalmente funcional para desarrolladores pequeños, esto hace que este software sea accesible para los usuarios [3].

Una de las principales características de Unity, es que permite crear juegos para sistemas de sobremesa como Windows o MAC OS, dispositivos móviles como iOS y Android, consolas como PlayStation, Xbox o Wii, debido a que permite la compilación gratuita en gran parte de las plataformas, permite ahorrar costos en el desarrollo de videojuegos [3].

La funcionalidad de Unity no se encuentra totalmente limitada, debido que pueden ser incrementada mediante plugins o complementos, de este modo Unity puede ser utilizado para desarrollar diferentes proyectos que tengan como necesidad crear un entorno gráfico, tal es el caso cuando se desea implementar la inteligencia artificial en videojuegos.

El motor gráfico de Unity cuenta con una interfaz como se observa en la figura 3, este se encuentra dividido de la siguiente manera:

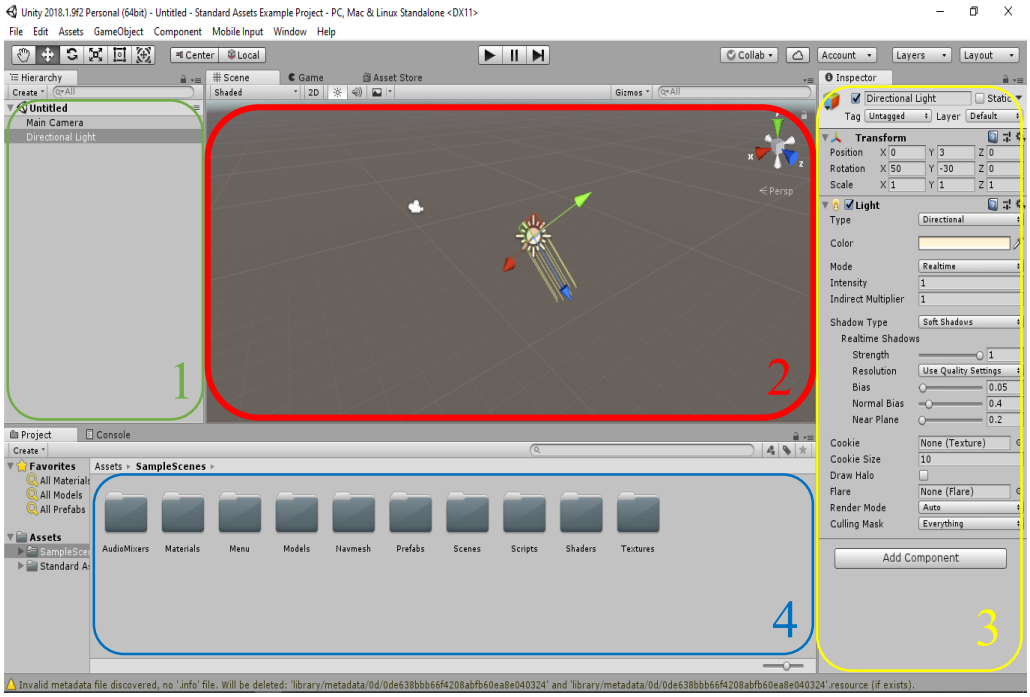

Figura 4.- Interfaz de programa Unity.

1. **Vista de "Jerarquías"**. Aquí veremos cada objeto nuevo de nuestro videojuego que esté en pantalla, por ejemplo, tendremos un listado de enemigos, los elementos de escenario con los que podemos chocar, el personaje, el fondo de la pantalla, entre otros.
2. **Vista de "Escena".** Consta de dos partes, "Escena" y "Juego". En la primera podremos colocar, mover, re dimensionar y rotar todos los objetos. La segunda veremos cómo quedará "en la realidad" nuestro juego. Podemos decir que la vista "Escena" es lo que ve el programador, y la vista Juego lo que verá el usuario cuando juegue.
3. **Vista de "Inspector".** Esta vista inspecciona cada objeto que seleccionemos, mediante ella podremos saber su tamaño, qué texturas y efectos le hemos aplicado, hacer que reaccione a las leyes de la física, entre otros.
4. **Vista de "Proyecto".** Aquí tendremos toda la colección de objetos, texturas, sonidos, vídeos y el material que utilizaremos para crear nuestro videojuego.

Aprendizaje automático.

Como se muestra en la figura 4, el aprendizaje automático es una rama de la inteligencia artificial, se trata de un aspecto de la informática en el que los ordenadores o las maquinas obtengan la capacidad de aprender sin necesidad de estar programadas específicamente para ello, y como resultado obtener predicciones ante una situación en particular. Para lograr eso el aprendizaje automático se apoya de algoritmos para aprender, estos algoritmos son el aprendizaje supervisado, aprendizaje no supervisado, aprendizaje de refuerzo [4].

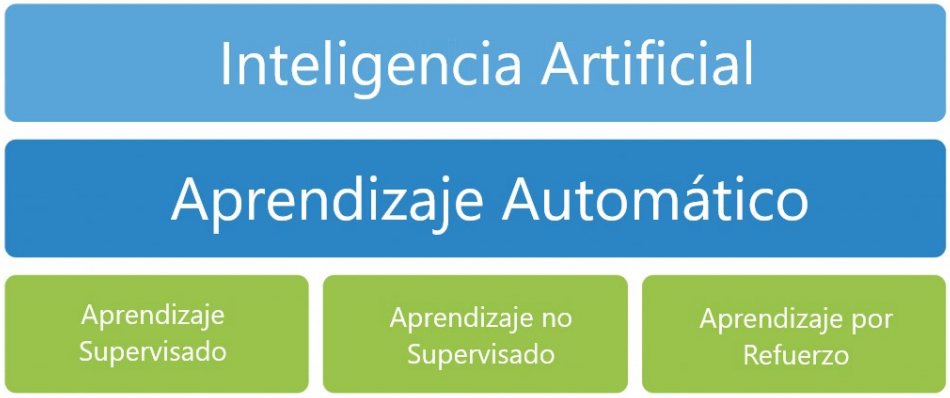

Figura 5.- Diagrama de inteligencia artificial.

Aprendizaje por refuerzo.

El objetivo específico del aprendizaje por refuerzo es que un agente aprenda un comportamiento dentro de un entorno, mediante observaciones y acciones, es decir que los datos que recolecta de su entorno de forma sensorial, darán como resultado una acción a la cual se le asignara un estímulo o recompensa, esta será positiva si su acción fue acertada, o negativa si su acción no fue conveniente para el comportamiento que se desea [5].

Citando un ejemplo, considere que el agente es un robot que tendrá como comportamiento la tarea extinguir incendios dentro de un entorno el cual podría ser una casa o un campo abierto, para poder lograr esta tarea, el robot necesita encontrar el incendio mediante sensores de temperatura, una vez ubicado el incendio el robot deberá tomar acciones para extinguirlo, si esta acción fue efectiva y lo está extinguiendo correctamente se le dará una recompensa positiva, en caso contrario y no lo está extinguiendo de manera efectiva, se le dará una recompensa negativa.

Este tipo de aprendizaje, está basado en un algoritmo que aprende basándose en la experiencia, por ese motivo para que un agente aprenda un comportamiento es necesario realizar una cantidad de iteraciones de las acciones, hasta obtener el comportamiento deseado, para ellos es de mucha ayuda recurrir a simulaciones.

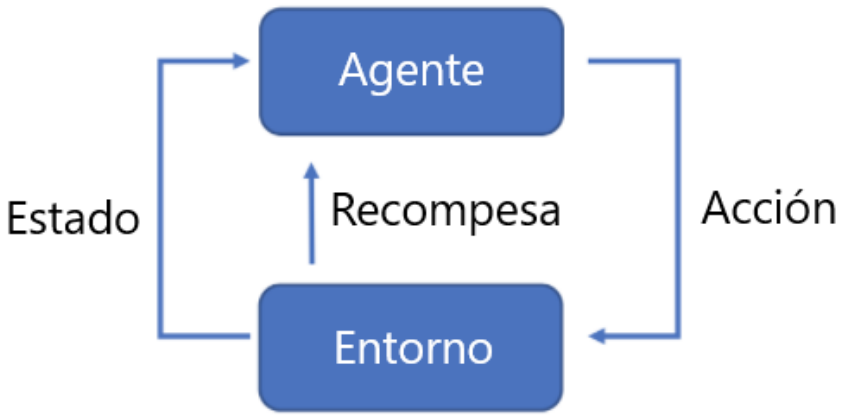

Figura 6.- Diagrama de aprendizaje por refuerzo.

En resumen, como se muestra en la figura 5 el aprendizaje por refuerzo trabaja dentro de un ciclo en el cual un agente genera una acción, la cual tiene una repercusión dentro de un entorno, dando como consecuencia una recompensa positiva o negativa, al mismo tiempo modificando el estado del agente.

Microsoft Visual Studio.

Visual Studio Code es un editor de código fuente ligero pero potente que se ejecuta en su escritorio y está disponible para Windows, macOS y Linux. Viene con soporte incorporado para JavaScript, TypeScript y Node.js y tiene un rico ecosistema de extensiones para otros lenguajes (como C++, C #, Java, Python, PHP, Go) y tiempos de ejecución (como .NET y Unity).

Permite a los desarrolladores crear sitios y aplicaciones web, así como servicios web en cualquier entorno compatible con la plataforma .NET, así, se pueden crear aplicaciones que se comuniquen entre estaciones de trabajo, páginas web, dispositivos móviles, dispositivos embebidos y videoconsolas, entre otros [6].

TensorFlow.

TensorFlow es una biblioteca de código abierto para realizar cálculos utilizando gráficos de flujo de datos, la representación subyacente de los modelos de aprendizaje profundo. Facilita la capacitación en una computadora de escritorio, servidor o dispositivo móvil. Dentro del kit de herramientas de ML-Agents, cuando entrena el comportamiento de un agente, la salida es un archivo de modelo TensorFlow que luego puede incrustar dentro de un cerebro [8].

Figura 7.- Logo de TensorFlow.

Kit de herramientas ML-Agents.

Es un complemento de Unity de código abierto que permite que los juegos y las simulaciones sirvan como entornos para la formación de agentes inteligentes, los agentes pueden recibir capacitación mediante el aprendizaje por refuerzo, el aprendizaje por imitación, la neuroevolución u otros métodos de aprendizaje automático a través de una API Python, también proporciona implementaciones (basadas en TensorFlow) de algoritmos de vanguardia para permitir a los desarrolladores de juegos y aficionados entrenar fácilmente agentes inteligentes para juegos 2D, 3D y VR / AR [8].

En la figura 7 se describe, la arquitectura interna de kit de herramientas ML-Agents para que un agente pueda aprender se puede describir mediante un diagrama de flujo:

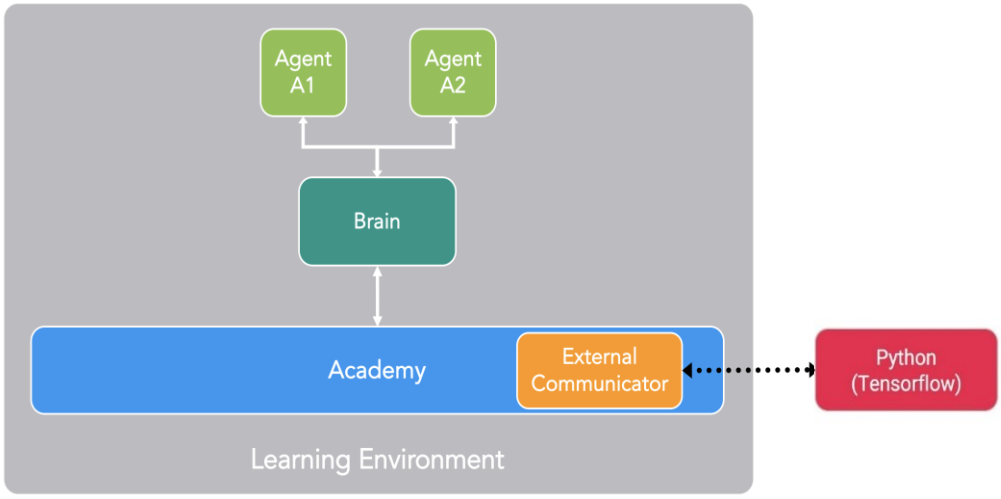

Figura 8.- Diagrama de la arquitectura de ML-Agents.

- **Learning Environment:** Contiene la escena con todos los personajes o agentes.
- **Agent:** Es cualquier personaje dentro de una escena y se encarga de generar sus observaciones, realizar las acciones que recibe y asignar una recompensa cuando sea apropiado, cada agente está vinculado a exactamente un cerebro.
- **Brain:** Encapsula la lógica para tomar decisiones, en esencia es lo que mantiene el comportamiento para cada agente y determina qué acciones debe tomar el en cada caso, en específico recibe las observaciones y recompensas del agente y genera una acción.
    - **External:** Las decisiones se toman usando el algoritmo que proporciona la API de TensorFlow.
    - **Internal:** Las decisiones se toman en función a la política generada por la API de TensorFlow, es decir, se usa un modelo ya entrenado denominado PPO.
    - **Player:** Las decisiones las toma un jugador por medio de un controlador.
- **Academy:** Organiza el proceso de observación y toma de decisiones, dentro de Academy se pueden especificar varios parámetros de todo el entorno, como la calidad de representación y la velocidad a la que se ejecuta el entorno, el comunicador externo se encuentra dentro.
- **External Communicator:** Conecta el entorno de aprendizaje con la API de Python(TensorFlow).

- **API Python(TensorFlow):** Contiene todos los algoritmos de aprendizaje automático que se utilizan para la capacitación, la API de Python no es parte de Unity, sino que es externo y se comunica con Unity a través del comunicador externo.

Sensor RP-Lidar.

El RPLidar A2 es un escáner láser 2D de 360 grados, comúnmente usado en aplicaciones de mapeos, localización y modelado de objetos y entornos, este sensor puede tomar hasta 4000 muestras por segundo con una alta velocidad de rotación dentro de un rango de escaneo de seis metros.

La forma en la cual opera este sensor, se basa en el principio de triangulación láser y trabaja con un sistema de adquisición y procesamiento de alta velocidad, la operación del sistema oscila entre 4000 veces por segundo, durante cada proceso de medición, el sensor emite una señal láser infrarroja modulada y la señal láser es reflejada por el objeto a detectar.

Una vez que regresa la señal enviada, esta es procesada por el sistema de adquisición de visión y procesamiento, una vez procesado los datos de muestreo es posible obtener datos como el ángulo y la distancia que existe entre objeto y el sensor, esto se puede visualizar a través de la interfaz de comunicación [9].

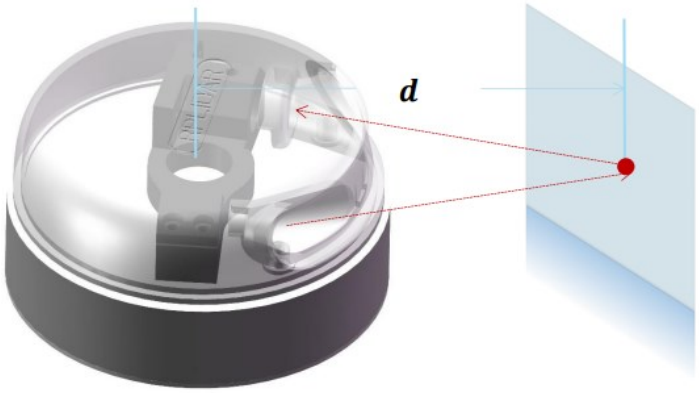

Figura 9.- Funcionamiento del sensor RP-Lidar.

La frecuencia de exploración típica del RPLidar A2 es de 10 Hz (600 rpm), bajo esta condición, la resolución de este sensor será 0.9°, la frecuencia de exploración puede ser libremente ajustado dentro del rango de 5-15hz según la aplicación que el usuario requiera de este sensor.

La arquitectura del sensor RPLidar consta de un escáner y un mecanismo que hace que el núcleo gire a gran velocidad, cuando este se encuentra funcionando normalmente, el escáner rotará en sentido horario, todos los usuarios pueden controlar el inicio, parada y velocidad de rotación del motor de rotación a través de PWM [9].

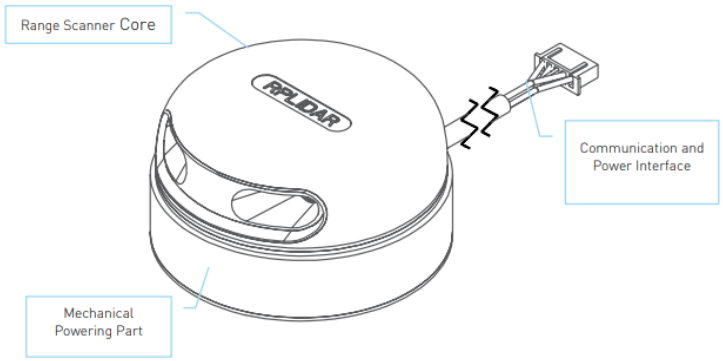

Figura 10..- Partes del sensor RP-Lidar.

Para poder posicionar al sensor RP-Lidar dentro de un entorno, es necesario implementar un sistema coordenado, en donde el eje x del sistema de coordenado se encuentra delante del sensor es decir el punto cero; así mismo el origen del sistema es el centro rotativo del escáner, y con respecto a algún eje x se puede deducir el ángulo donde se posiciona el punto cero, el cual aumenta a medida que esta gira en sentido horario [9].

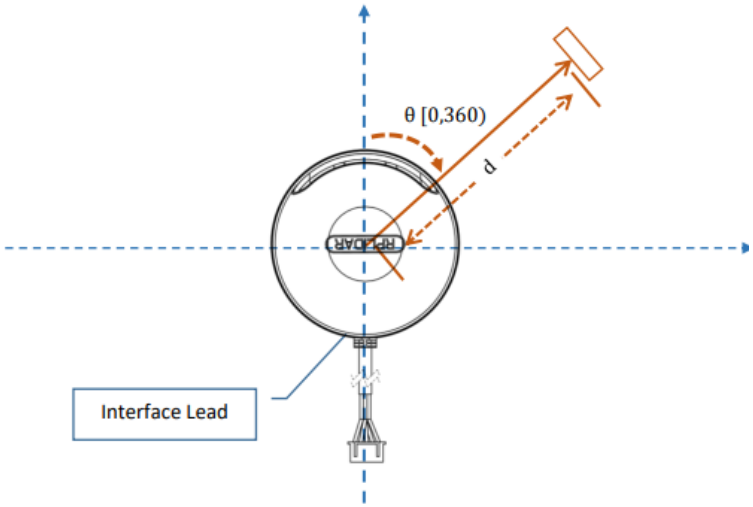

Figura 11.- Posicionamiento del sensor RP-Lidar.

El sensor RPLidar toma una alimentación externa para alimentar el núcleo del escáner y el sistema motor que hace girar el núcleo, para que el sensor funcione normalmente, el sistema necesita garantizar la salida de la alimentación y cumplir con sus requisitos de la fuente de alimentación, además toma el puerto serie 3.3V-TTL (UART) como la comunicación interfaz.

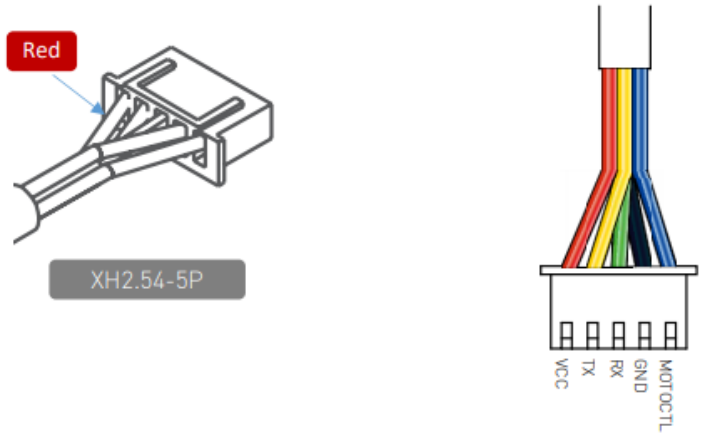

Figura 12.-Conexión del sensor RP-Lidar.

Arduino Mega 2560.

Es una placa electrónica basada en el microprocesador Atmega2560, tiene 54 pines digitales de entrada/salida, de los cuales 15 pueden utilizarse como salidas PWM, 16 entradas analógicas, cuenta con una memoria flash de 256 KB para el almacenamiento del código, así mismo dispone de cuatro conexiones para TTL de comunicación serie, este puede ser programado mediante el software de Arduino [10].

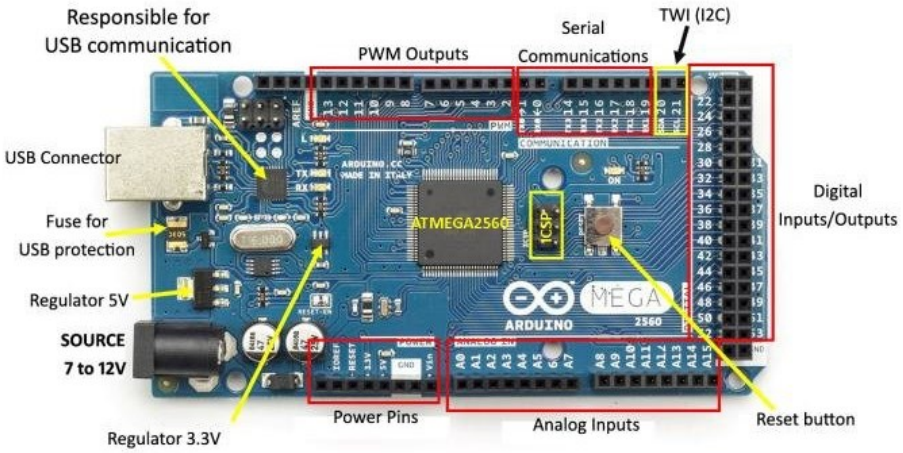

Figura 13.- Placa electrónica Arduino Mega 2560.

Con esta placa electrónica tiene como finalidad simplificar la interacción entre el hardware y software, puede servir para poder controlar elementos físicos, como también poder crear elementos autónomos, es decir automatizar.

Modulo Bluetooth HC-05.

El modulo bluetooth HC-05 es un pequeño modulo transmisor/receptor TTL fue diseñado para ser controlado a través de RS232. Permite transmitir como recibir datos a través de tecnología bluetooth sin necesidad de realizar conexiones físicas de comunicación, Este dispositivo se controla mediante comando AT mediante el puerto serie, es compatible con cualquier microcontrolador con comunicación UART [11].

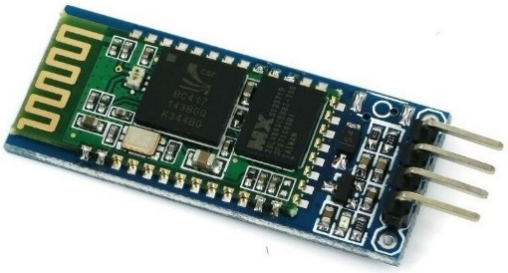

Figura 14.- Modulo Bluetooth HC-05.

Metodología

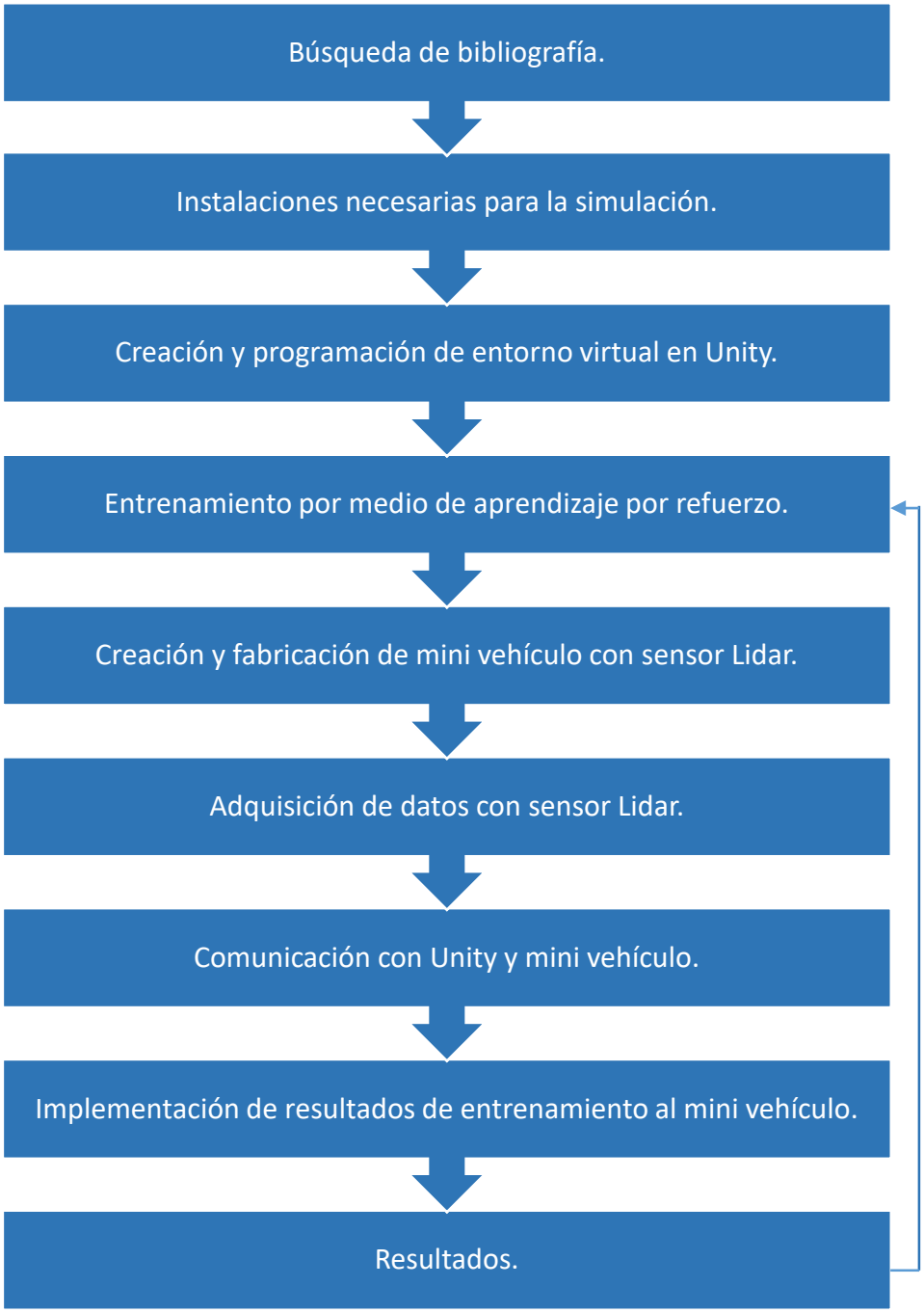

Figura 15.- Diagrama de flujo de la metodología del desarrollo de proyecto.

Instalación de programas y complementos.

Es necesario instalar una serie de programas, con el fin de poder diseñar nuestro entorno virtual, así mismo los complementos son necesarios debido que estos cuentan con los modelos para realizar nuestro entrenamiento por medio de aprendizaje por refuerzo.

- Unity. (versión 2018 superior).
- Anaconda. (versión 5.1).
- ML-Agents.
- TensorFlow. (versión 1.7.1).

Desarrollo de entorno virtual en Unity.

Dentro del motor gráfico de Unity podemos generar diferentes entornos virtuales dependiendo la necesidad de nuestro trabajo a fin, para este proyecto nosotros necesitamos crear un entorno el cual debe estar diseñado para simular un mini vehículo siendo este nuestro agente y con este mismo poder recorrer una pista delimitada por paredes frontales y laterales.

Dentro de la escena en la interfaz de Unity, logramos diseñar el mini vehículo mediante un cubo el cual se puede posicionar en cualquier eje de referencia, de la misma manera se agregan los demás objetos, como la pista en la cual se desplazará nuestro agente o mini vehiculó y las paredes las cuales delimitaran el área de la pista, para crear estos objetos se utilizan prismas de diferentes formas que contiene la interfaz de Unity.

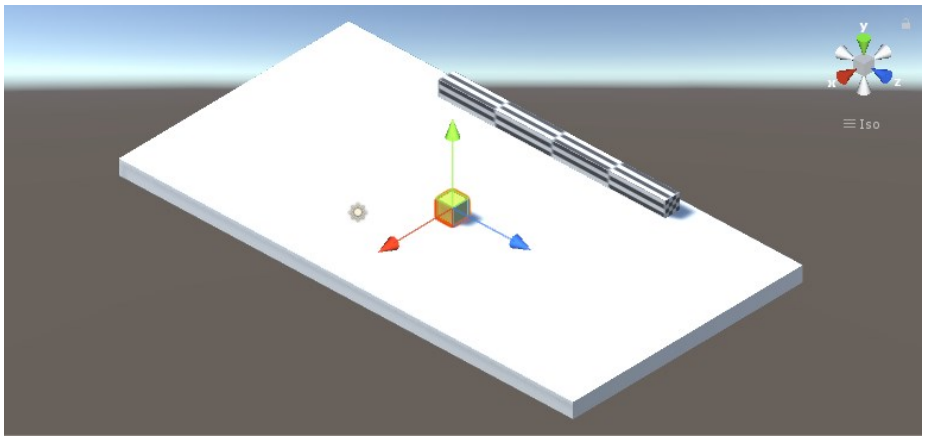

Figura 16.- Entorno virtual en desarrollo.

Cada objeto agregado a la escena tiene propiedades predeterminadas provocando que al momento de simular el entorno los objetos se comporten de maneras no deseadas, debido a esto cada uno de los objetos debe ser configurado con propiedades físicas como gravedad, peso, entro otros, con el fin de volverlos sólidos y su simulación sea relativamente realista.

Interacción con el entorno virtual.

Para poder ser capaces de obtener datos del entorno virtual que se diseñó, y poder utilizar dicha información recolectada en nuestro entrenamiento con aprendizaje por refuerzo, se debe recurrir a la programación, esto es posible gracias a que Unity permite realizar script los cuales sirven para que podamos interactuar con nuestro entorno virtual, para este proyecto nosotros debemos realizar scripts los cuales nos permitan medir distancias, conocer la ubicación de los objetos, controlar movimientos, entre otros.

El primer paso para lograr lo anteriormente mencionado, es crear tres scripts para correlacionarlas con los objetos que se encuentran en nuestra escena, para que posteriormente estos nos ayuden a entrenar nuestro agente, el primer script de se le denomina “Agent”, el segundo “Academy”, y el último “Brain”.

Variables del entorno.

El script denominado “Agent” se correlacionará con el objeto que utilizaremos para simular nuestro mini vehiculó o agente, es decir el prisma con forma de cubo, de misma manera este se deberá correlacionar directamente con el script “Brain” y por el último el script de “Academy” el cual comunicara los scripts con el compilador externo de Python.

```
public class Prueba4Agent : Agent
{
        public GameObject Pared1;
        public GameObject Pared2;
        public GameObject Pared3;
        public GameObject Movil;
                public float Speed;
                        float FinalDistance;
                        float ParedDistance;
        }
```

Figura 17.- Código para identificar las variables del entorno virtual.

Dentro del script “Agent” debemos indicar que objetos estarán dentro de nuestro escenario, asignándole una variable a cada uno de estos objetos, esto es necesario debido que necesitamos saber cómo nuestro agente esta interactuando con nuestro entorno, como se muestra en el fragmento de código de nuestro script “Agent” en la figura 16, cada objeto puede ser correlacionado mediante una variable, como por ejemplo “Pared1”, “Pared2”, “Movil”, es decir el prisma que simula el mini vehículo puede ser correlacionado con la variable “Movil”.

Declarar posición del agente.

De mismo modo en el script “Agent”, tenemos que determinar una posición en la cual el agente se encontrara dentro de nuestra escena, con el fin de que el agente siempre inicie en la misma ubicación cada vez que comience o se reinicie la simulación.

Debido que trabajamos dentro de un entorno tridimensional, proponemos la ubicación deseada mediante nuestros tres vectores los cuales son “X” igual a 0, “Y” igual a 0.15 y “Z” igual a 0, siendo este el punto de inicio cada vez que comience o reinicie nuestra simulación, esto se puede visualizar en el siguiente código de la figura 17.

```
public override void AgentReset (){
gameObject.transform.localPosition = new Vector3(0f, 0.15f, 0f);
        }
```

Figura 18.- Código para posicionar el agente.

Obtener distancias del entorno.

Una vez conociendo todos los objetos con los cuales interactúa nuestro agente y proponiendo su punto de partida, nosotros podemos obtener variables de nuestro entorno, como por ejemplo la distancia que existe entre el agente y los demás objetos que lo rodean, como se muestra en el código en la figura 18, se puede realizar mediante un comando llamado “RayCast”.

```
RaycastHit hit1;
Ray downray1 = new Ray(transform.TransformDirection(Vector3.forward));
Debug.DrawRay(transform.TransformDirection(Vector3.forward)*35, Color.red);
```

Figura 19.- Código para obtener las distancias del entorno.

Este comando nos permite posicionar una especie de rayo en una determinada dirección partiendo de nuestro agente, con esto podemos obtener la distancia que existe entre el objeto que interrumpa la trayectoria del rayo y el agente, para lograr esto implementamos el siguiente código, determinando la dirección, el tamaño y su color, como se aprecia en la figura 19.

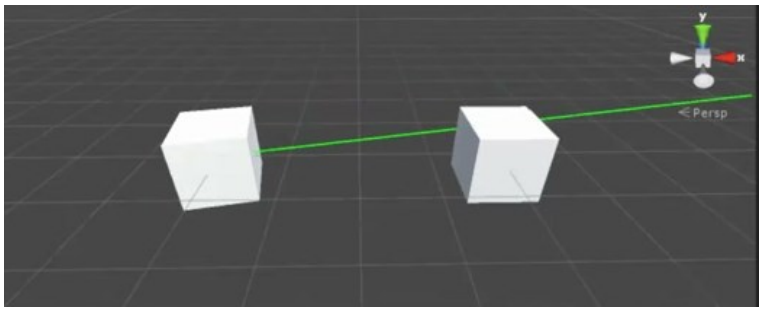

Figura 20.- Ejemplo de medición de distancia con RayCast.

Desplazamiento dentro del entorno.

Controlar el desplazamiento o la posición en la cual se encuentra nuestro agente, es una de las partes más importantes dentro de nuestra simulación, debido que sin ella no podemos realizar el entrenamiento, esto se puede implementar mediante el incremento o decremento de unidades sobre los vectores donde se encuentre ubicado nuestro agente, esto provoca su desplazamiento sobre alguno de sus ejes.

```
float newZ = transform.localPosition.z + (vectorAction[0] * Speed * Time.deltaTime);
      newZ = Mathf.Clamp(newZ, 0f, 22f);

float newX = transform.localPosition.x + (vectorAction[1] * Speed * Time.deltaTime);
      newX = Mathf.Clamp(newX, -9f, 9f);

      transform.localPosition = new Vector3(newX, 0, newZ);
```

Figura 21.- Código para controlar el desplazamiento del agente.

Para explicar en el código en la figura 20, el incremento y decremento se realiza mediante alguna entrada, la cual puede ser una tecla de teclado físico del portátil, y se puede seleccionar desde las opciones de la interfaz que brinda el script “Brain”, así mismo esta entrada está directamente relacionada con la variable denominada “vectorAction” dentro del script “Agent”, en el mismo fragmento de código podemos modificar la velocidad de desplazamiento, así como el vector al cual queremos que corresponda la variable de entrada, esto se puede probar simulando el entorno virtual y configurando el script “Brain” desde la interfaz de Unity como “Player”.

Recompensas y condiciones a seguir.

Para poder ser capaces de realizar el entrenamiento, se deben considerar las condiciones que deseamos que nuestro agente respete, como por ejemplo la distancia en que se encuentra el agente con respecto a otro, la ubicación a la cual el agente debe desplazarse, entre otros, del mismo modo debemos asignarle una recompensa positiva o negativa si el agente cumple o no con la condición que le asignamos.

```
if ((FDistance >= 4f && FDistance<=5f) && (PDistance>=3f && PDistance <=4f))
   {
            AddReward(10f);
          }
   else {
            AddReward(-0.3f);
               }
```

Figura 22.- Código para proponer las condiciones para el entrenamiento.

Las recompensas se asignan mediante el comando “AddReward” el cual se observa en la figura 21, dando un valor negativo si el agente no cumple con la condición que se le asigna, no existe un método estándar para saber cuánto debe ser el valor a asignar a cada condición, por este motivo se asignan de manera empírica simulando el entrenamiento mediante el compilador externo de Python y se modifica el valor de la recompensa hasta obtener el mejor resultado en el entrenamiento y nuestro agente obedezca nuestras condiciones de manera correcta.

Reinicio dentro del entorno.

Otra parte importante dentro de nuestro script “Agent” es realizar un fragmento de código en la figura 22, con el cual podemos ser capaces de resetear la simulación, esto se refiere a que el agente cada vez que choque con algún objeto, vuelva a la ubicación que se le asigno cada vez que reiniciara la simulación.

```
private void OnTriggerEnter(Collider other)
 {
    if (other.CompareTag("Muro1"))
     {
           AddReward(-10f);
           Done();
     }
 }
```

Figura 23.- Código para reiniciar el punto de partida del agente.

Entrenamiento.

Generar modelo virtual.

Antes de poder realizar el entrenamiento, se debe generar un modelo externo de nuestro entorno virtual desde Unity, para generar dicho modelo externo es necesario configurar como “External” el script “Brain” desde la interfaz de nuestro motor gráfico, esto tiene como fin de que sea la API de Python que tome las decisiones de las acciones que realizara el agente hasta cumplir con las condiciones que se le indicaron.

API de Python Tensorflow.

Una vez generado el modelo externo se puede realizar el entrenamiento, ejecutando la consola del programa Anaconda la cual contiene la API de TensorFlow, posteriormente activando los complementes de ML-Agents y por ultimo escogiendo nuestro modelo externo anteriormente generado.

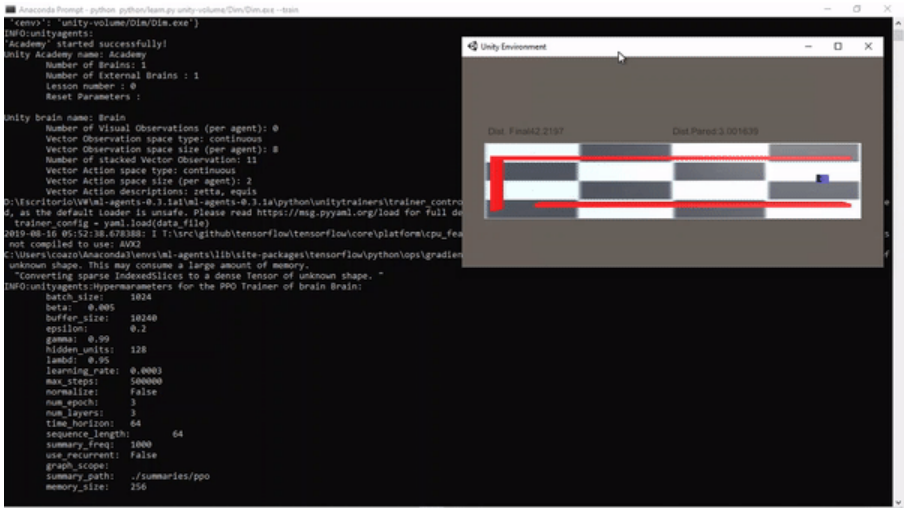

Figura 24.- Entrenamiento del modelo externo por prompt de Anaconda.

Como se muestra en la figura 23, la API de Python TensorFlow carga el modelo externo para entrenarlo por medio de una especie de simulación, donde la API de Python aplica los algoritmos de aprendizaje por refuerzo, realizando una serie de acciones en el agente dentro del entorno y asignándole recompensas si esas acciones fueron o no favorables, así hasta llegar a las condiciones que se le establecieron al agente.

Generar el modelo entrenado.

Al finalizar con el entrenamiento, la API de Python genera un modelo ya entrenado denominado PPO (Proximal Policy Optimization), debido que este es uno de los algoritmos en los que se apoya la API de Python para lograr el entrenamiento por aprendizaje por refuerzo, este modelo siempre será generado al finalizar sin importar que los resultados del enteramiento sean positivos o negativos, por ese motivo hay que modificar el peso de las recompensas.

Cargar el modelo entrenado.

Obteniendo el modelo entrenado PPO, puede ser cargado directamente a Unity cambiando el script "Brain" a tipo "Internal" y seleccionando el modelo entrenado, con esto se puede observar los resultados del entrenamiento directamente en la interfaz de Unity, simulando el entorno virtual.

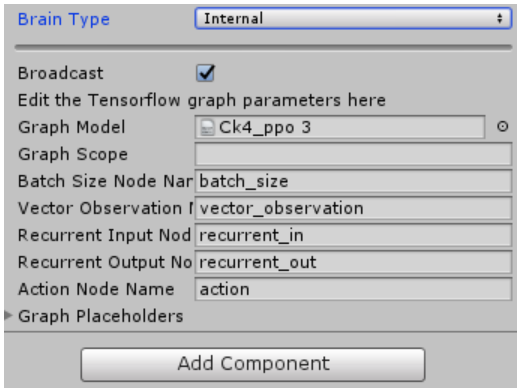

Figura 25.- Cargar el modelo PPO.

El modelo generado no es más que un archivo con terminación ".ppo", este modelo se puede generar con diferentes pesos sobre las recompensas, además dentro de la API de Python es posible configurar el número de iteraciones que debe realizar en la simulación, el tiempo que dura cada iteración, entre otras configuraciones que nos ayudaran a obtener mejores resultados con la misma simulación, solo es necesario variar el nombre del archivo a generar.

Diseño del mini vehículo.

Circuito electrónico del minivehiculo.

Mediante la tarjeta Arduino Mega se puede realizar el control del mini vehiculó en conjunto de un circuito integrado L293D el cual sirve para controlar los motores dando como facilidad aislar el control de la potencia y no afecta a nuestra tarjeta Arduino Mega, para facilitar la movilidad de nuestro mini vehículo de manera inalámbrica, se implementó un módulo Bluetooth HC-06 para transmitir los datos obtenidos del sensor RP-Lidar y también para recibir los datos necesarios para controlar las orientación de los motores dándole dirección a nuestro mini vehiculó.

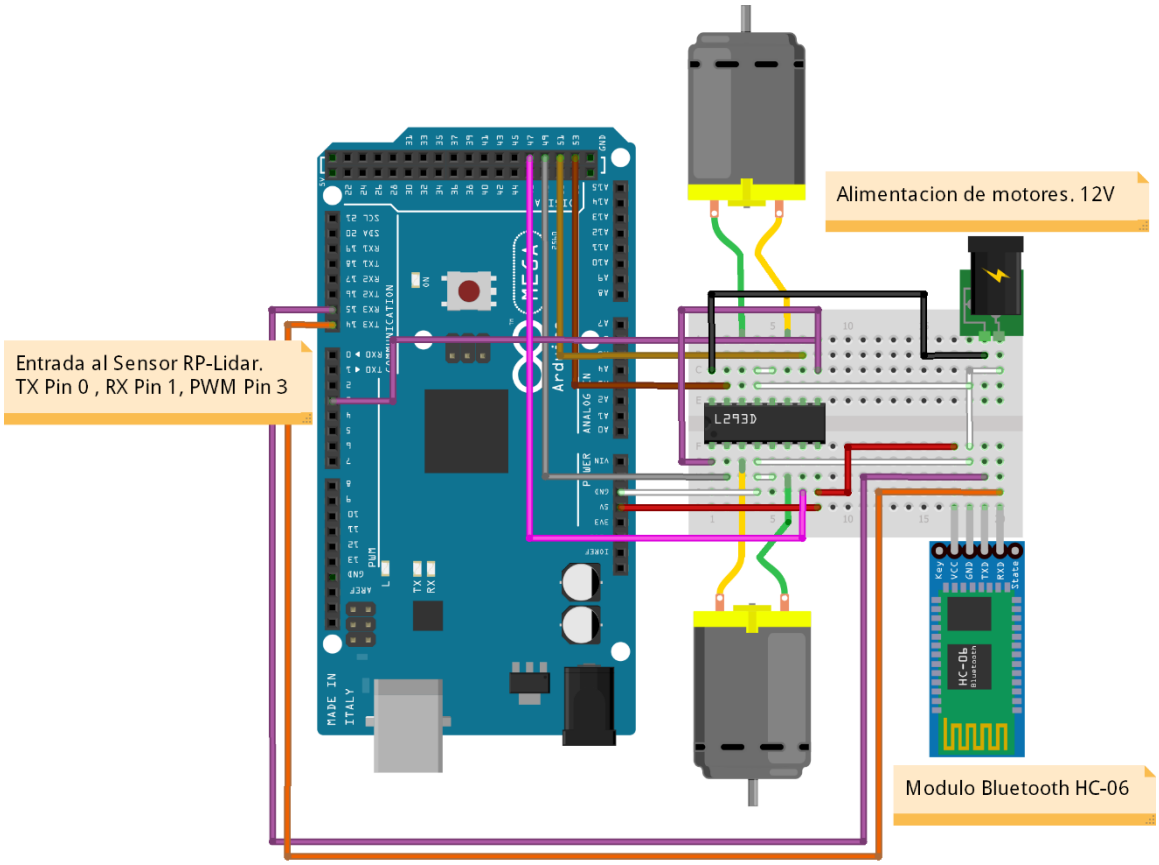

Figura 26.- Esquemático del circuito electrónico del mini vehículo.

Adquisición de datos del sensor RP-Lidar.

Algoritmo para adquirir datos de sensor RP-Lidar.

El sensor RP-Lidar trabaja mediante comunicación serial para enviar y recibir información, por ese motivo, mediante la tarjeta Arduino se debe establecer esa comunicación, con las librerías necesarias que fueron desarrolladas en el entorno Arduino podemos recopilar toda esa información.

```
void loop () {
if (IS_OK(lidar.waitPoint())) {
float distance = lidar.getCurrentPoint().distance; //distance value in mm unit
float angle    = lidar.getCurrentPoint().angle; //anglue value in degree
bool  startBit = lidar.getCurrentPoint().startBit; //whether this point is belong to a new scan
byte  quality  = lidar.getCurrentPoint().quality; //quality of the current measurement
}
```

Figura 27.- Código para obtener la información del sensor RP-Lidar.

La información que podemos obtener del sensor RP-Lidar son diferentes, como la distancia, el ángulo donde se tomó la lectura, entre otros, entonces mediante los comandos que facilita la librería del sensor fácilmente podemos obtener esa información para posteriormente utilizarla.

Comunicación entre el mini vehículo y Unity.

Transmisión de datos de Arduino a Unity.

De toda la información que proporciona el sensor RP-Lidar, solo nos interesan conocer la distancia que existe únicamente de dos ángulos, la primera distancia es la parte frontal del mini vehículo este tiene un ángulo de 0 grados a 2 grados, la segundo es el lado derecho el cual dentro de los 360 grados se encuentra ubicado entre los 90 grados.

```
if(angle <= 2.0 && angle > 1.0 ) {
    Serial1.print("A");
    Serial1.println(distance/100);
}
if(angle <= 91.0 && angle > 90.0 ) {
    Serial1.print("B");
    Serial1.println(distance/100);
}
```

Figura 28.- Código para obtener y transmitir la distancia en los angulos de interes.

Como se muestra en el código, cada que el sensor se encuentre en esa posición en grados, guarda la información en una la variable y la envía por el puerto serial en formato de texto siendo recibido y posteriormente tratada la información en Unity, cabe mencionar que toda la información transmitirá y recibida entre Unity al Arduino es mediante la transmisión serial por medio de bluetooth, entre el módulo HC-06 y el computador.

Recepción de datos de Arduino en Unity.

Para recibir en Unity los datos de la distancia de nuestras dos ubicaciones deseadas las cuales fueron transmitidas desde Arduino, estos tienen que ser tratados para que podamos utilizarlas dentro de nuestro entorno virtual.

```
string value = serialPort.ReadLine();
if (value.Contains("A")){
ungrado = float.Parse(vec6[1]);
TexZFinal.text = ("1_Grado" + ungrado);
}
if (value.Contains("B")){
novgrado = float.Parse(vec6[1]);
TexB.text = ("90_Grado" + vec6[1]);
}
```

Figura 29.- Código para recibir los datos transmitidos por Arduino.

En este caso la manera de tratar la información es separando en dos lo que se encuentra en todo momento en el puerto serial, esto se debe a que Arduino envía las dos distancias con diferentes etiquetas para no confundir una distancia con respecto a la otra, luego separamos esas dos distancias y las almacenamos en diferentes variables, para poder utilizar la información como sea necesario.

Transmisión de control de movimiento desde Unity.

Debido que los scripts “Brain” de Unity son los encargados de tomar las decisiones, es necesario llevar esas decisiones a nuestro mini vehículo por lo cual Unity debe transmitir información hasta el Arduino con el fin de controlar el movimiento de los motores del mini vehiculó.

Como se muestra en el código, nosotros transmitiremos el valor que se encuentre dentro del comando “vectorAction” donde si es -1 entonces el mini vehículo se desplazará en reversa, en el caso que el valor sea 1 entonces se desplazará hacia enfrente, cada valor se envía por puerto serial con una letra las cuales corresponden a cada valor.

```
if (estado == -1){
serialPort.Write("A"); }
if (estado == 0){
serialPort.Write("B"); }
if (estado == 1){
serialPort.Write("C"); }
```

Figura 30.- Código para transmitir datos de control a Arduino.

Recepción de control de movimiento desde Unity.

De igual forma nosotros recibimos en Arduino los datos transmitidos de Unity, como se muestra en el código leemos cual fue la letra recibida y asignamos una acción para cada letra, debido a que estamos trabajando con un driver para controlar los motores tenemos que cambiar el estado de las salidas digitales para generar el movimiento que corresponda a cada letra.

```
char letra=Serial1.read();
if(letra == A ){
digitalWrite (dirmotorA1,HIGH);// gira motor A derecha
digitalWrite (dirmotorA2,LOW);
analogWrite (PWMA, velocidad);
digitalWrite (dirmotorB1,HIGH);// gira motor B derecha
digitalWrite (dirmotorB2,LOW);
analogWrite (PWMA, velocidad);
delay(100);
}
```

Figura 31.- Código para recepción de los datos para el control del mini vehículo.

Resultados

Diseño del entorno virtual.

El entorno virtual que se consideró implementar consta de dos paredes laterales y una frontal, así mismo se decidió dejar un acceso libre para poder extender la pista para pruebas posteriores, también se le agregaron los aspectos físicos a las paredes para que sean sólidas y se encuentren fijas, esto también se implementó en el prisma en forma de cubo que simula el mini vehículo.

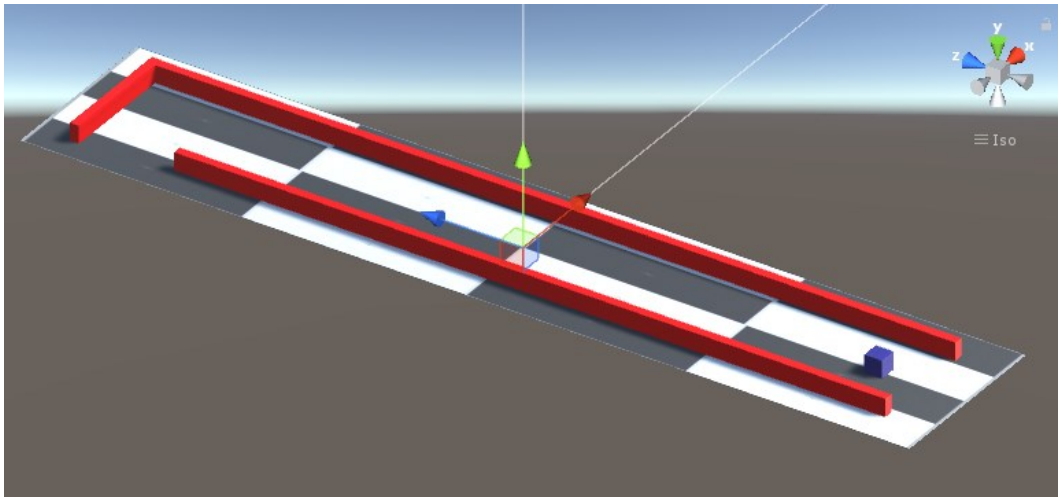

Figura 32.- Diseño final del entorno virtual.

Entorno virtual programado.

Al entorno virtual, se le programó el control necesario para su desplazamiento, en este caso solo se tomaron dos movimientos en particular, los cuales son para adelante y el otro para atrás, esto debido a que es el movimiento que buscamos que realice nuestro mini vehículo o agente, por ese motivo dentro de las propiedades físicas hay que restringir el movimiento sobre los otros vectores.

Otra cosa que se programó en nuestro entorno virtual, fue poder obtener información de nuestro entorno, en ese caso la distancia de nuestro agente con respecto a otro, solo ocupamos dos distancias, la del agente con respecto a la pared frontal, y la otra distancia con respecto a la pared lateral derecha, esto debido a que la condición que deseamos que nuestro agente cumpla solo necesitamos estos dos datos.

Podemos observar en la figura 32, desde una vista area de nuestra pista simulada, podemos visualizar a nuestro agente midiendo las dos distancias que existen con respecto a nuestro agente, otra cosa que se agregó dentro de la programación de nuestro entorno virtual es poder observar los datos obtenidos de la distancia, en el mismo entorno virtual facilitando la rápida lectura de nuestra variable.

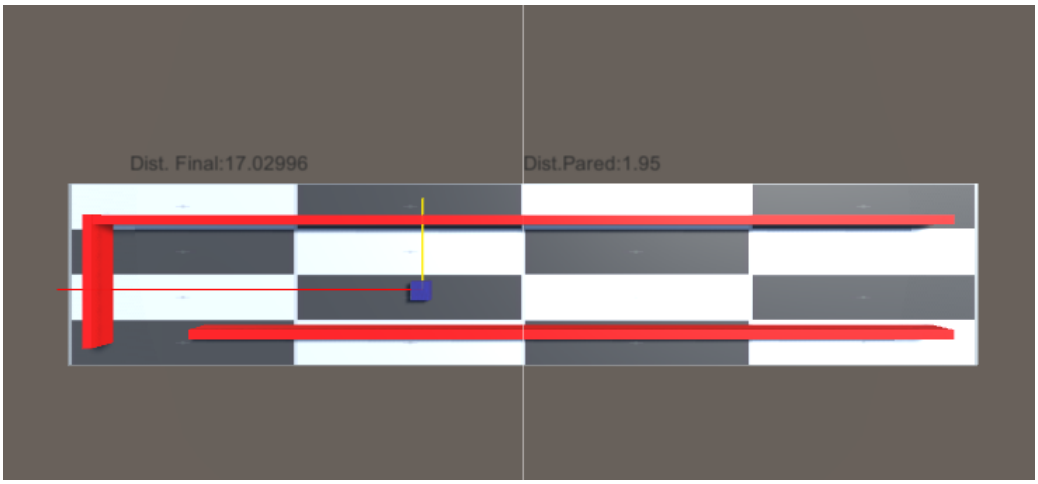

Figura 33.- Entorno virtual programado.

La programación también incluye los pesos de cada una de nuestras recompensas con sus respectivas condiciones, aunque aún no son los pesos definitivos, puesto que dependiendo de los resultados del primer entrenamiento sabremos si es conveniente modificarlas, además, es importante mencionar que para realizar el entrenamiento es necesario ignorar la comunicación del Arduino con respecto a Unity, por lo cual lo excluiremos de nuestro código mientras se realiza la parte del entrenamiento, solo se incluirá cuando ya tengamos un modelo PPO entrenado de manera correcta y los podamos probar en nuestro mini vehículo.

Entrenamiento del entorno virtual.

Teniendo el entorno virtual con su respectiva programación necesaria, podemos realizar el entrenamiento de nuestro entorno virtual, este consta de un número determinado de iteraciones cada una con diferentes acciones haciendo cambiar el valor de la recompensa, hasta lograr que la acción de nuestro agente sea la correcta.

Se puede observar en la figura 33, el número de iteración en la cual se encuentra el entrenamiento, así mismo se visualiza los pesos de las recompensas que está siendo modificadas con las respectivas acciones que realiza nuestro agente las cueles se pueden observar en la simulación del modelo externo.

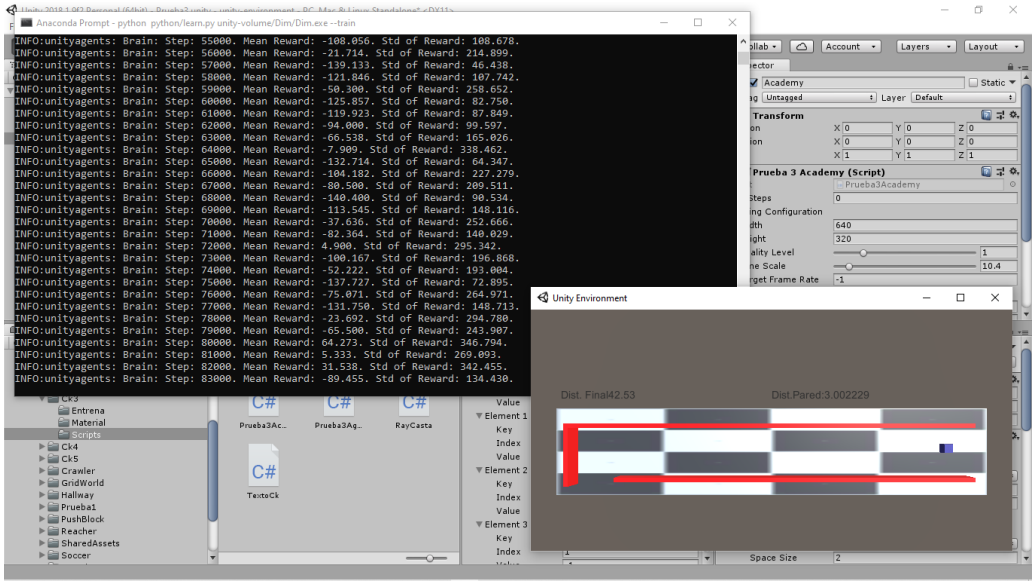

Figura 34.- Entrenamiento del entorno virtual externo.

Desarrollo del mini vehículo.

El mini vehículo se desarrolló de la misma manera que el esquema electrónico, posicionando el sensor RP-Lidar en la parte frontal para tener una lectura de las distancias sin objetos que bloqueen los ángulos de interés.

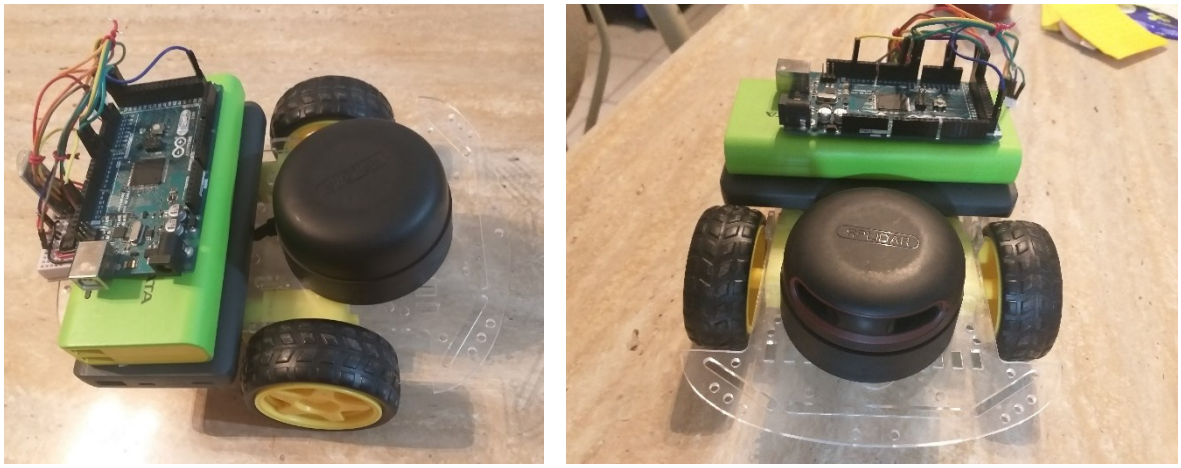

Figura 35.- Diseño final del mini vehículo.

Adquisición de datos con el Arduino del sensor RP-Lidar.

Como se puede visualizar en la figura 35, la adquisición de datos mediante el sensor RP-Lidar con Arduino, se realiza mediante una condicionen donde se escogen los grados donde nos interesa medir la distancia, para diferenciar la distancia de un ángulo con el otro, etiquetamos cada uno con diferentes letras, esto se realiza así debido en este caso el sensor envía lo datos de manera aleatoria y una forma de distinguirlo es etiquetarlos y posteriormente tratarlos en Unity para separarlos.

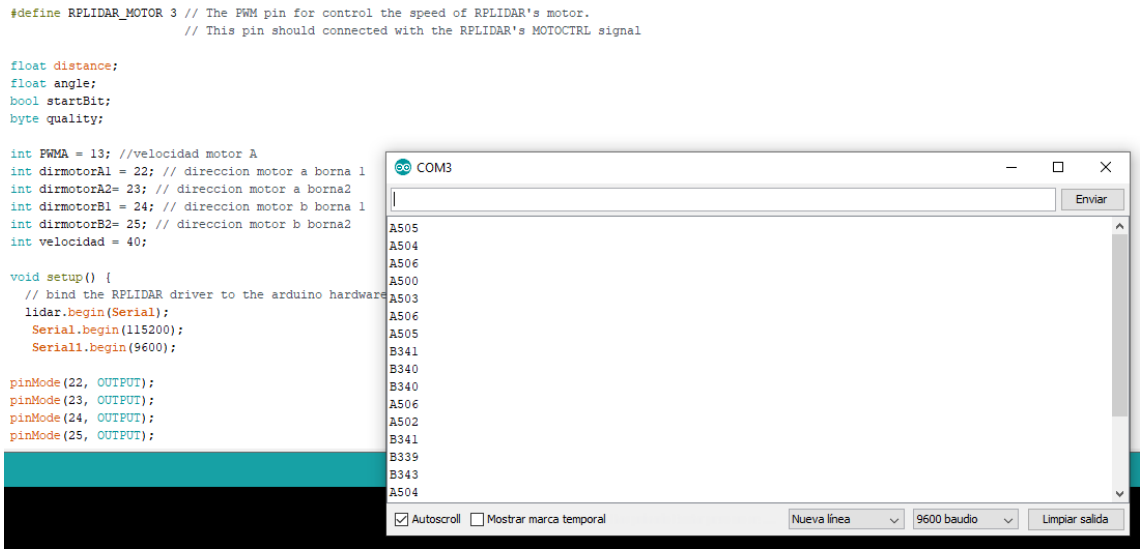

Figura 36.- Datos obtenidos del sensor RP-Lidar.

Comunicación e implementación del entrenamiento.

Una vez realizado el entrenamiento del entorno virtual de Unity y generado el modelo entrenado PPO, es posible establecer la comunicación entre el mini vehículo y Unity, para lograr esto debemos incluir los códigos con los cuales se establece esa comunicación, para probar si la recepción de datos es correcta, como se menciona solo se establecerá comunicación sin embargo ningún control, por lo cual aún no incluiremos.

Las distancias medidas de los dos ángulos de intereses, que fueron obtenidas por nuestro sensor RP-Lidar, se pueden visualizar en nuestro entorno virtual con el fin de probar la correcta comunicación entre Unity y el mini vehículo.

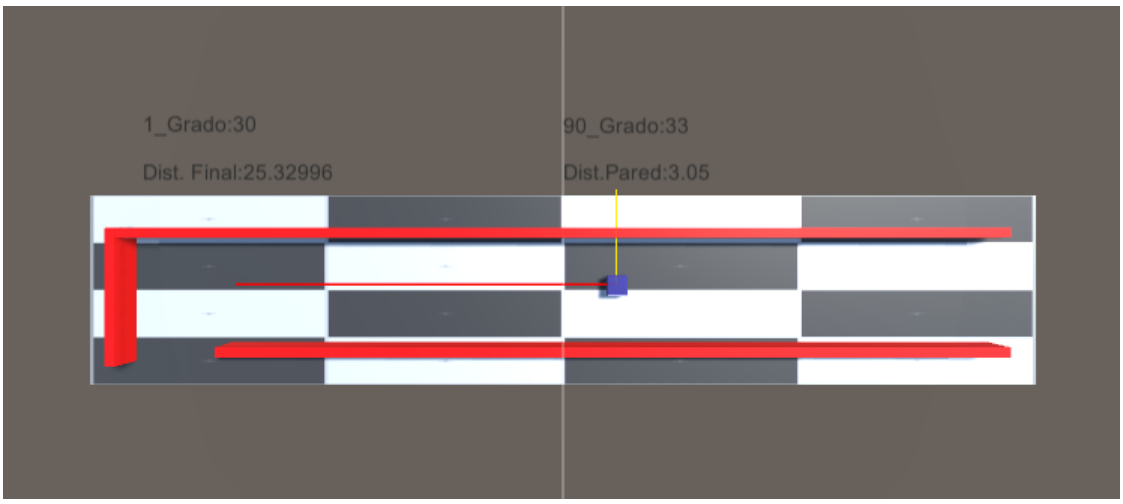

Figura 37.- Entorno virtual recibiendo los datos del sensor RP-Lidar.

Comprobando que exista una correcta comunicación, nosotros podemos probar físicamente el modelo PPO entrenado, para esto debemos incluir ahora el código necesario para implementar el control de nuestro mini vehículo, en este punto todo trabajara en conjunto y para tener un punto de referencia el agente y el mini vehículo físico, se deben comportar de manera similar, como se muestra en la figura 37, se implementó una pista similar a nuestro entorno virtual.

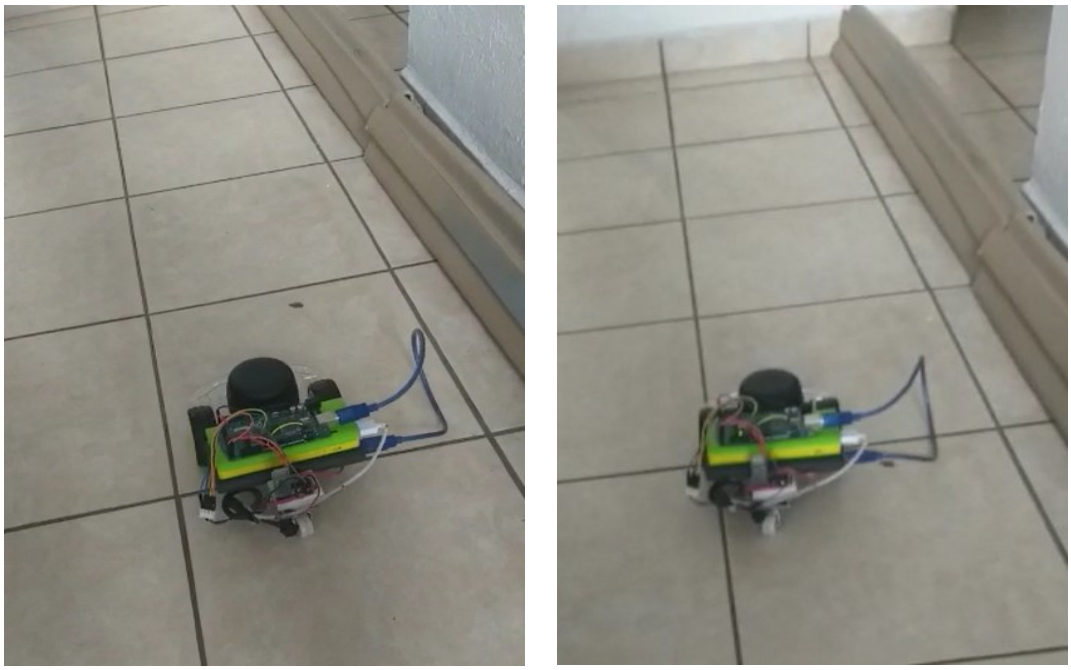

Figura 38.- Control físico del mini vehículo.

Conclusiones

Observando los resultados obtenidos del proyecto, podemos concluir que la implementación de un control por medio de entrenamiento con aprendizaje por refuerzo es viable, puesto que implementar la metodología para lograr este tipo de control tienes muchas ventajas, una de ellas es la facilidad de realizar el entrenamiento por media de una simulación y posteriormente probarla de manera física.

Sin embargo, es necesario hacer una experimentación amplia debido que es importante cubrir todas las condiciones relevantes para el control, de igual forma se debe diseñar un entorno virtual el cual tenga una similitud relativa con nuestro entorno físico, para que al momento de probar de manera física en control exista una mejor interacción del agente con el entorno.

Este tipo de proyecto tiene aportación muy amplia en diferentes áreas, una de ellas es la robótica puesto que se pueden diseñar entornos virtuales donde se puedan simular diferentes tipos de robots con los ambientes donde se utilizan, dando la facilidad de probar el control antes de implementarlo, y al mismo tiempo poder adaptarlos con mayor facilidad si el ambiente donde operan desea ser modificado.

Bibliografía

[1] A. F. Montoro, Python 3 al descubierto, Mexico: Alfaomega grupo editor S.A de C.V., 2013.

[2] C. Guagliano, Programación en Python I: Entorno de Programación – Sintaxis – Estructuras de Control, Buenos Aires, Argentina: Six Ediciones, 2019.

[3] A. D. Díaz, Unity 2017.X Curso Práctico, España: RA-MA Editorial, 2017

[4] L. Rouhiainen, Inteligencia artificial: 101 cosas que debes saber hoy sobre nuestro futuro, Alienta Editorial, 2018.

[5] "Unity-Technologies/ml-agents", GitHub, 2019. [Online]. Available: https://github.com/Unity-Technologies/ml-agents/blob/master/docs/Background-Machine-Learning.md. [Accessed: 22-Dec-2019].

[6] V. Code, "Documentation for Visual Studio Code", Code.visualstudio.com, 2019. [Online]. Available: https://code.visualstudio.com/docs. [Accessed: 22- Dec- 2019].

[7] "Unity-Technologies/ml-agents", GitHub, 2019. [Online]. Available: https://github.com/Unity-Technologies/ml-agents/blob/master/docs/Background-TensorFlow.md. [Accessed: 22- Dec-2019].

[8] "Unity-Technologies/ml-agents", GitHub, 2019. [Online]. Available: https://github.com/Unity-Technologies/ml-agents/blob/master/docs/ML-Agents-Overview.md. [Accessed: 22- Dec-2019].

[9] Bucket.download.slamtec.com, 2019. [Online]. Available: http://bucket.download.slamtec.com/a3fca807bdf41e15905f873373873994b0cb950d/LM310_SLAMTEC_rplidarkit_usermanual_A3M1_v1.2_en.pdf. [Accessed: 22- Dec- 2019].

[10] D. A. M. Cruz, Módulo con controladores lógicos programables para la enseñanza de aprendizaje de electrónica, Alzamora, Alicante: Área de innovación y desarrollo, S.L, 2019.

[11] R. Estrada, "Modulo bluetooth hc-05", HeTPro, 2019. [Online]. Available: https://hetpro-store.com/modulo-bluetooth-hc-05/. [Accessed: 22- Dec- 2019].

[12] A. A. Velasco, «Aplicación de técnicas de aprendizaje por refuerzo,» 2018.

[13] G. Sepúlvera Cervantes, E. Vega-Alvarado and E. Portilla-Floresa, "Machine Learning para Robots, del Entrenamiento Virtual a la Tarea Real", Pädi Boletín Científico de Ciencias Básicas e Ingenierías del ICBI, vol. 7, no., pp. 14-18, 2019. Available: 10.29057/icbi.v7iespecial.4785.