Universidad
Rey Juan Carlos

Motor de físicas mejorado para simulador robótico basado en tecnologías web

TRABAJO FIN DE GRADO

NATALIA MONFORTE RODRÍGUEZ

JOSÉ MARÍA CAÑAS PLAZA

OCTUBRE 2020

Índice

Introducción

Robótica

(a) Roomba

(b) Thermomix

(c) Tello

(d) Tesla

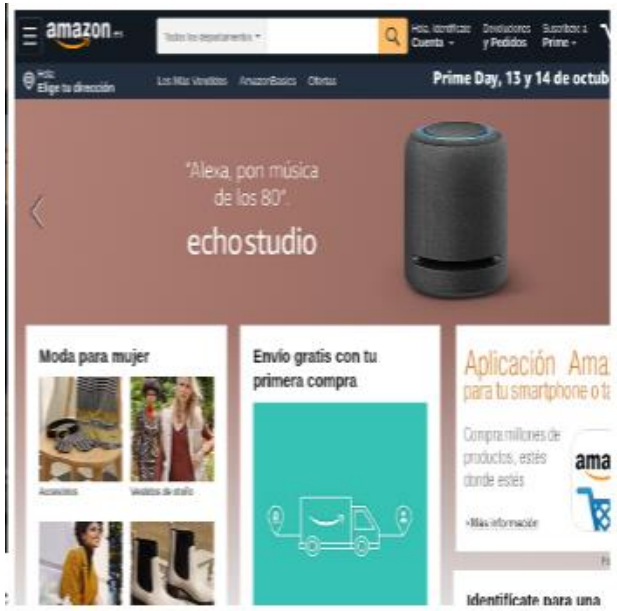
(e) Robot DaVinci

(f) Robots LEGO

- Creación de máquinas automatizadas útiles que recrean comportamientos humanos o animales.

- Partes de un robot:
  - Hardware
  - Software

Tecnologías web



- Aplicaciones web
- Modelo cliente – servidor -> Protocolo HTTP

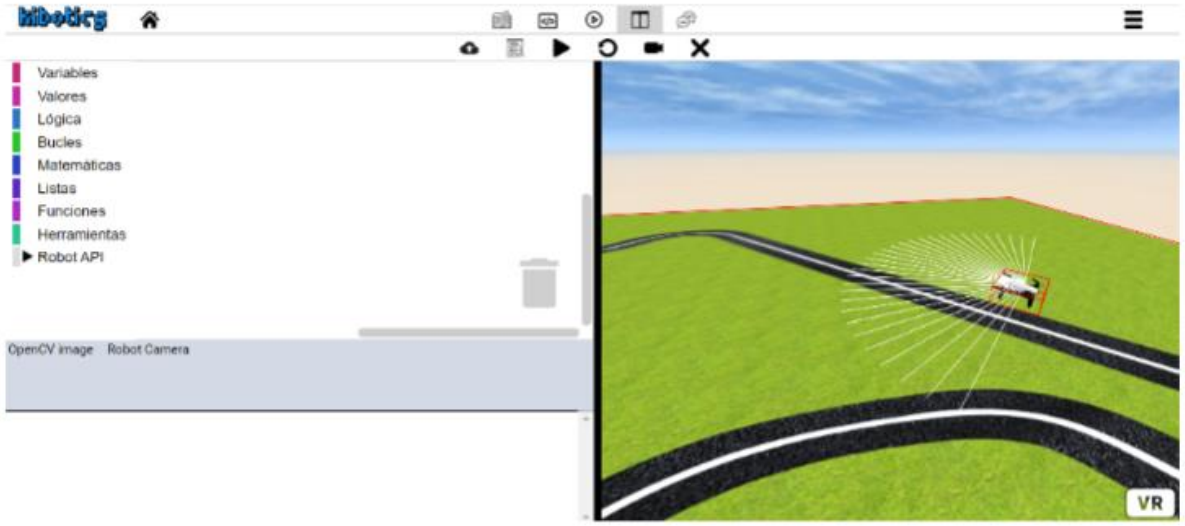
- Tecnologías de *frontend:*
  - *HTML5*
  - *CSS3*
  - *JavaScript*
- Tecnologías de *backend*

Docencia robótica

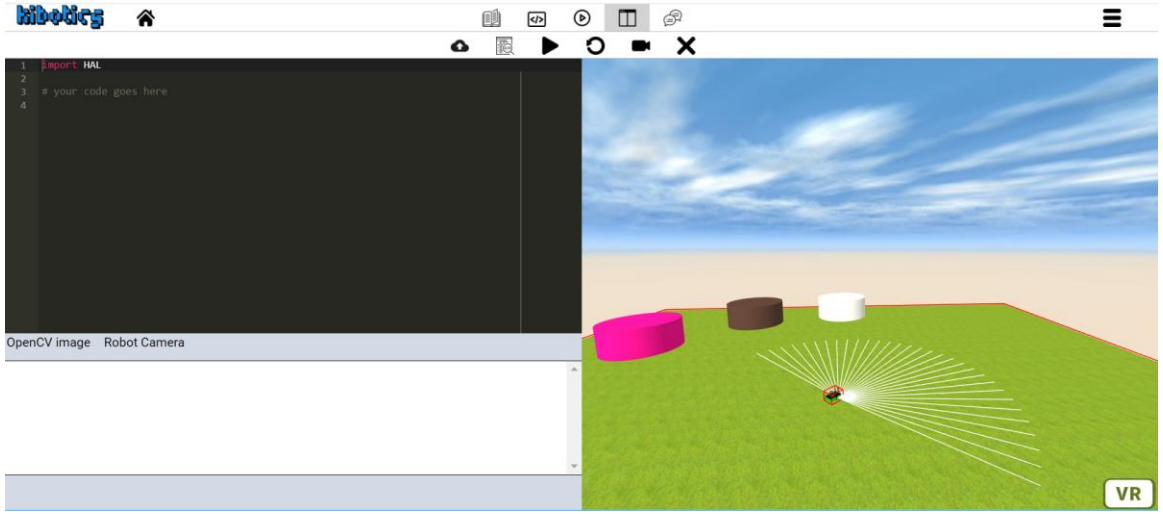
Robótica + tecnologías web ⟶ docencia robótica

Educación *STEM*

- *Lenobotics*
- *LEGO education*
- *OpenRoberta*
- *iRobot*
- *Scratch*
- *Kibotics* ⟶ soporta *Python* y *Scratch*



Interfaz de programación en *Kibotics* de un ejercicio en *Scratch*



Interfaz de programación en *Kibotics* de un ejercicio en *Python*

Objetivos

Objetivos

1

- Motor de físicas basado en *A-Frame* que replique de modo realista el movimiento autónomo de los robots.
- Complementario a *CANNON*.
- Robots de distinta masa.

2

- Novedosos ejercicios para la plataforma educativa *Kibotics* que aprovechen el nuevo motor de físicas.

Herramientas


kibotics


Javascript
JS


HTML
5

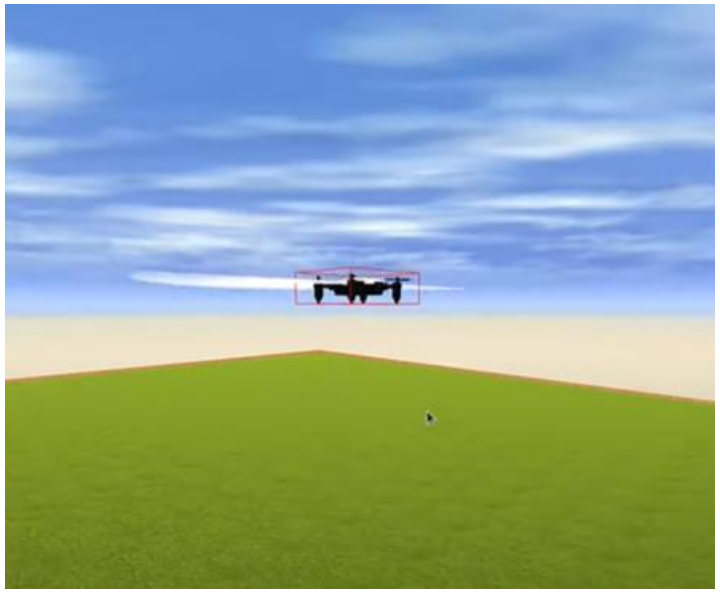

JSON

Mejora de las físicas en *WebSim*

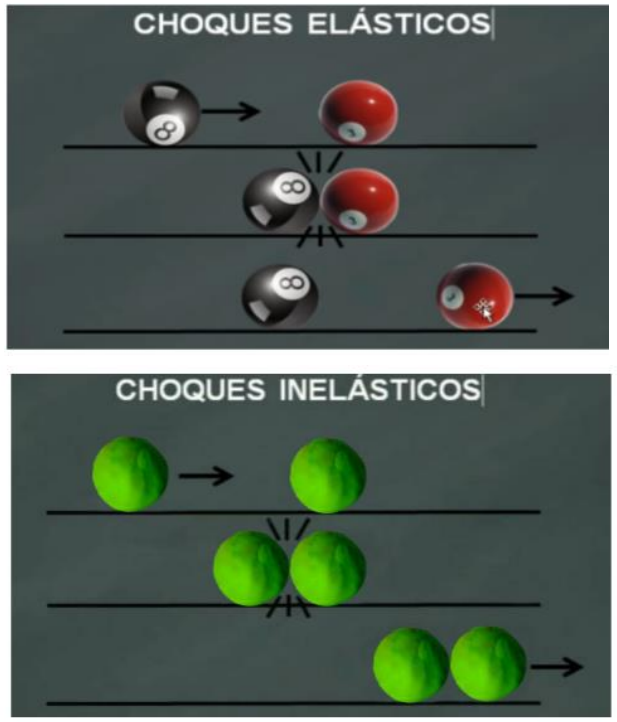
Estudios previos: motor de físicas por defecto en *A-Frame*

- *CANNON* es el motor por defecto de *A-Frame*

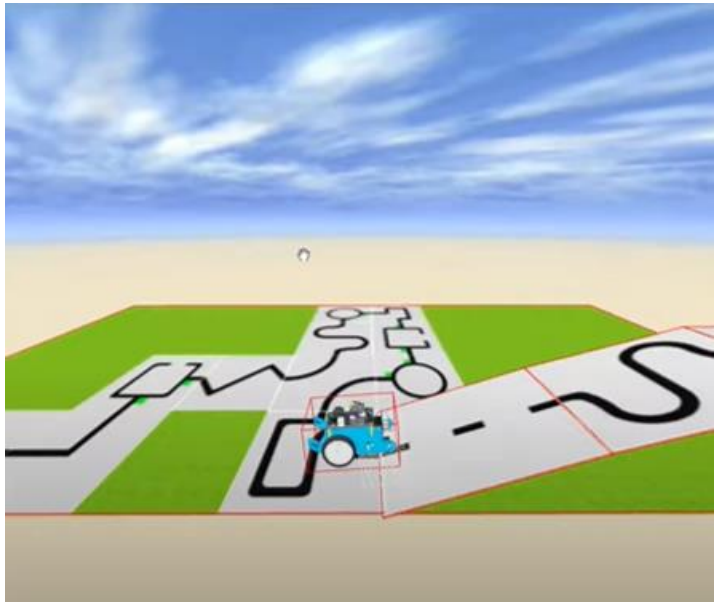
Gravedad



Colisiones



Fricción



Motor de físicas actual para robots en *WebSim*

- No recrea un movimiento realista.
- Modelo cinemático: los robots adquieren instantáneamente la velocidad ideal.
- Aceleración infinita.

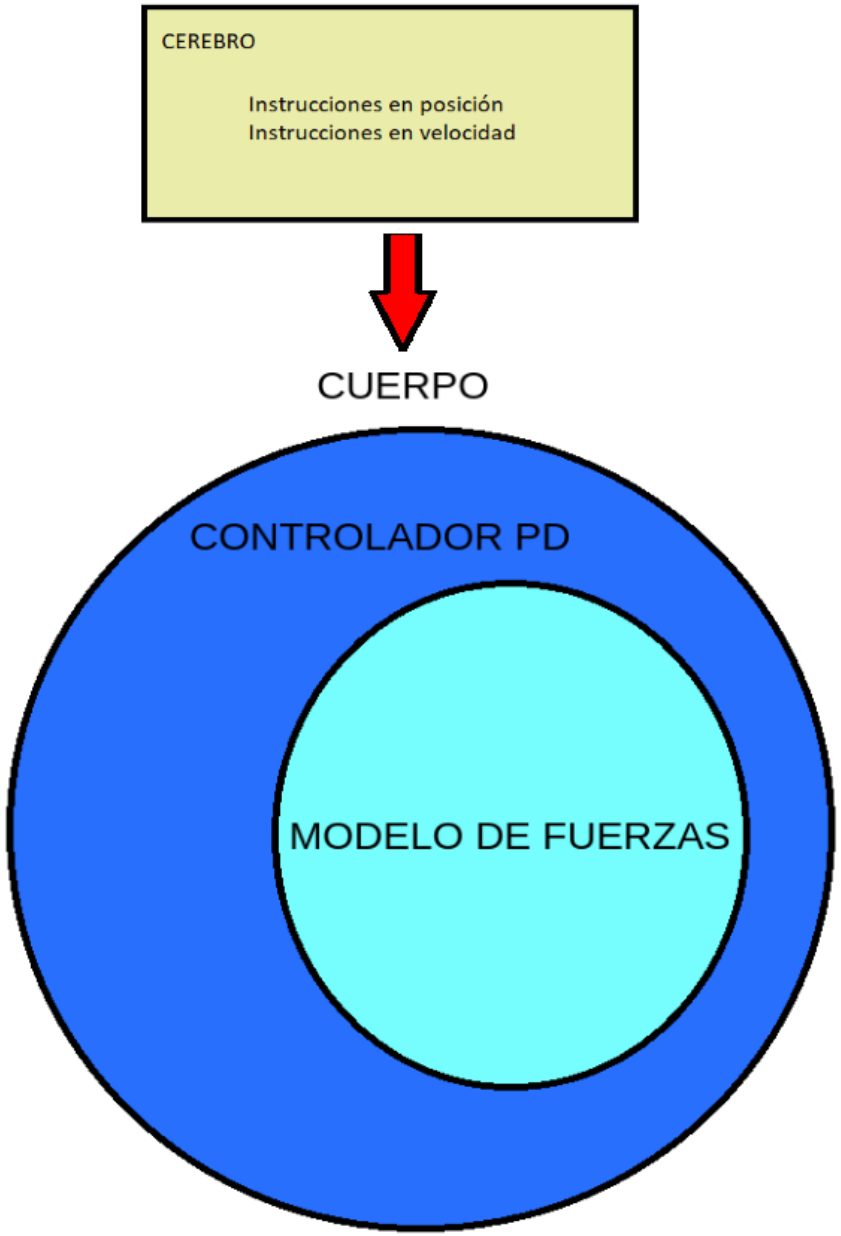
```
updatePosition(rotation, velocity, robotPos) {
    if(simEnabled){
        let x = velocity.x / 10 * Math.cos(rotation.y * Math.PI / 180);
        let z = velocity.x / 10 * Math.sin(-rotation.y * Math.PI / 180);
        let y = (velocity.y / 10);
        robotPos.x += x;
        robotPos.z += z;
        robotPos.y += y;
    }
    return robotPos;
}
```

Nuevo motor de físicas complementario

1. DISEÑO

$$Fuerza\ robot = Fuerza\ autónoma + Fuerza\ gravedad + Fuerza\ fricción$$

- Fuerzas gravedad y fricción: materializadas por *CANNON a* su propio ritmo.
- Fuerza autónoma: materializada por el motor complementario a otro ritmo y teniendo en cuenta las velocidades deseadas por el cerebro programado.



Parámetros del modelo de fuerzas	
mass	Masa del robot
inertia	Momento de inercia del robot
Fmax	Fuerza máxima aplicable
Tmax	Torque máximo aplicable
accelerationMax	Aceleración lineal máxima
angularAccelerationMax	Aceleración angular máxima
linealSpeedMax	Velocidad lineal máxima que puede alcanzar el robot
angularSpeedMax	Velocidad angular máxima que puede alcanzar el robot
Parámetros de *A-Frame*	
restitution	Conservación de la energía cinética en un choque entre partículas
gravity	Gravedad
friction	Fricción (rozamiento estático y dinámico)
linearDamping	Amortiguación lineal (rozamiento dinámico en el movimiento lineal)
angularDamping	Amortiguación angular (rozamiento dinámico en el movimiento angular)

2. MODELO DE FUERZAS

- A partir de la definición de la masa y el momento de inercia del robot, se calcula la aceleración a aplicar en el siguiente diferencial de tiempo.

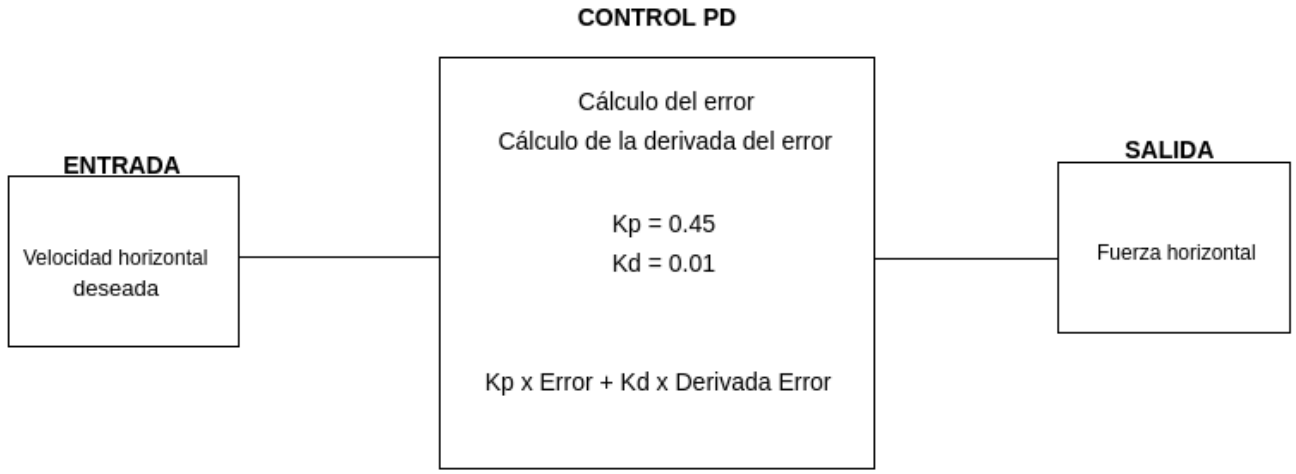
- Esa aceleración determina la evolución de las velocidades y posiciones del robot.

- Parámetros
  - Fuerza máxima.
  - Par máximo.
  - Masa.
  - Momento de inercia.
  - Velocidad lineal máxima.
  - Velocidad angular máxima.

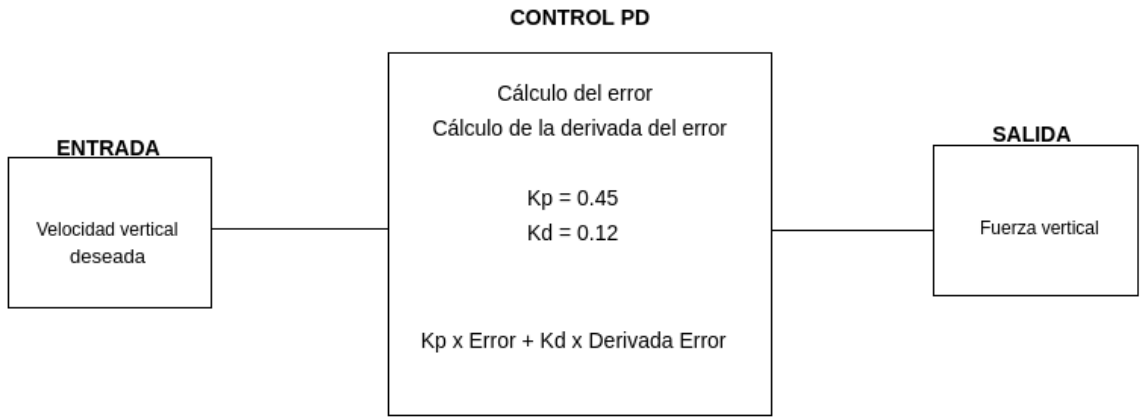
3. CONTROLADOR PD

- Traducción de las velocidades deseadas en cada momento del cerebro a la fuerza autónoma a aplicar al robot.

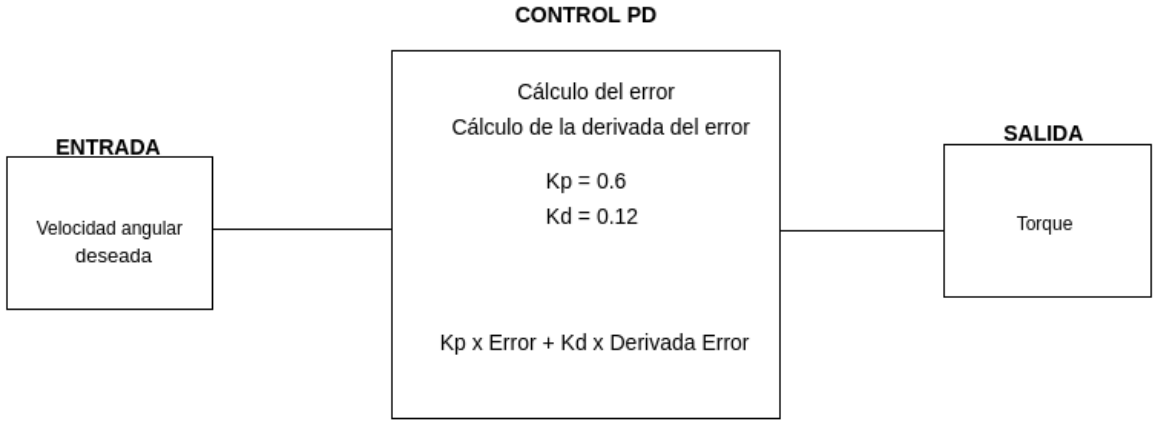
1. Controlador PD en velocidad del plano horizontal (para drones y robots de suelo)



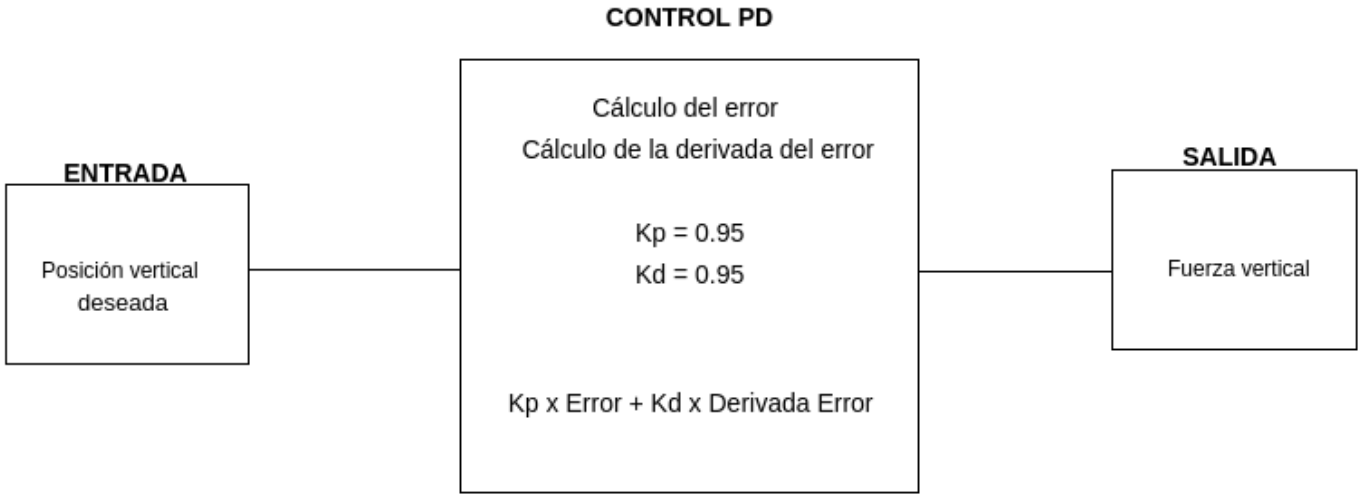
2. Controlador PD en velocidad del eje vertical (para drones)



3. Controlador PD en velocidad angular horizontal (yaw) (para drones y robots de suelo)



4. Controlador PD en posición para la altura (para drones cuando Vz = 0)



4. TIMING

Coexistencia de motores ⟶ Necesidad de combinarlos

Tienen que trabajar en la misma escala temporal:

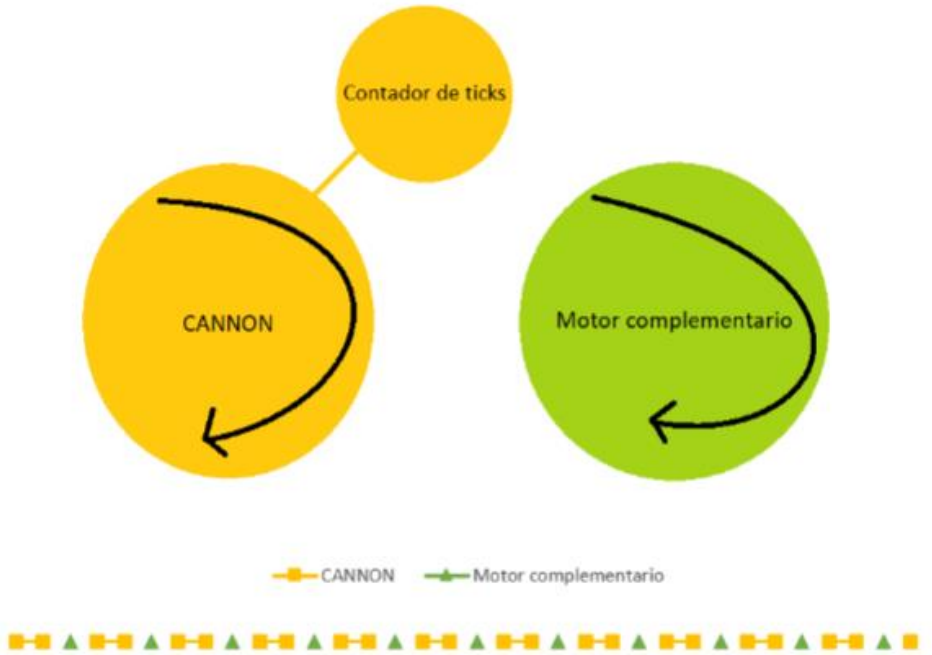
$$Aceleración\ autónoma = iteracionesCANNON\ x\ aceleración\ calculada$$

- ¿Timing del nuevo motor complementario? 20 ms

  setTimeout(this.auxiliaryPhysics.bind(this), 20);

- ¿Timing de *CANNON*? Desconocido

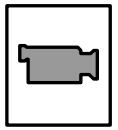
  No mide tiempos explícitamente, depende de la carga de la máquina.



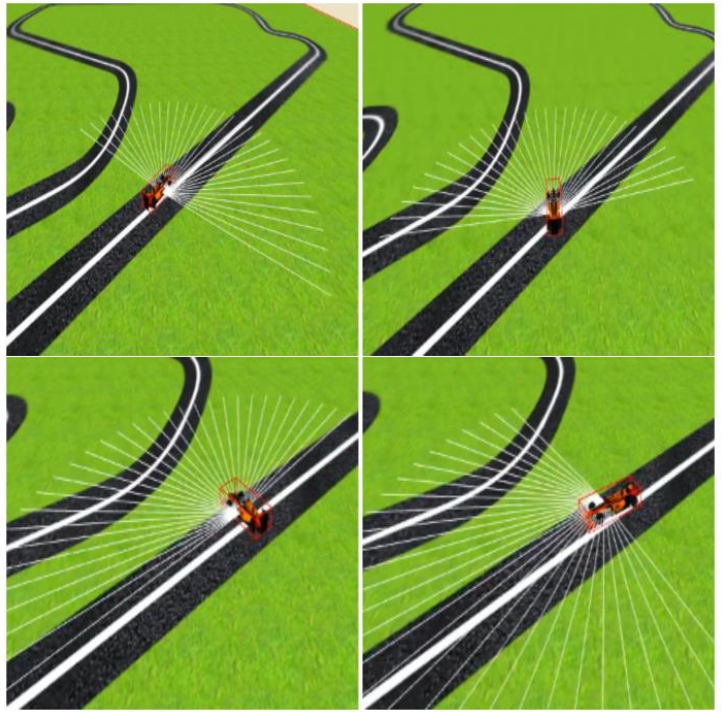
Validación experimental

Simulación realista de robots terrestres

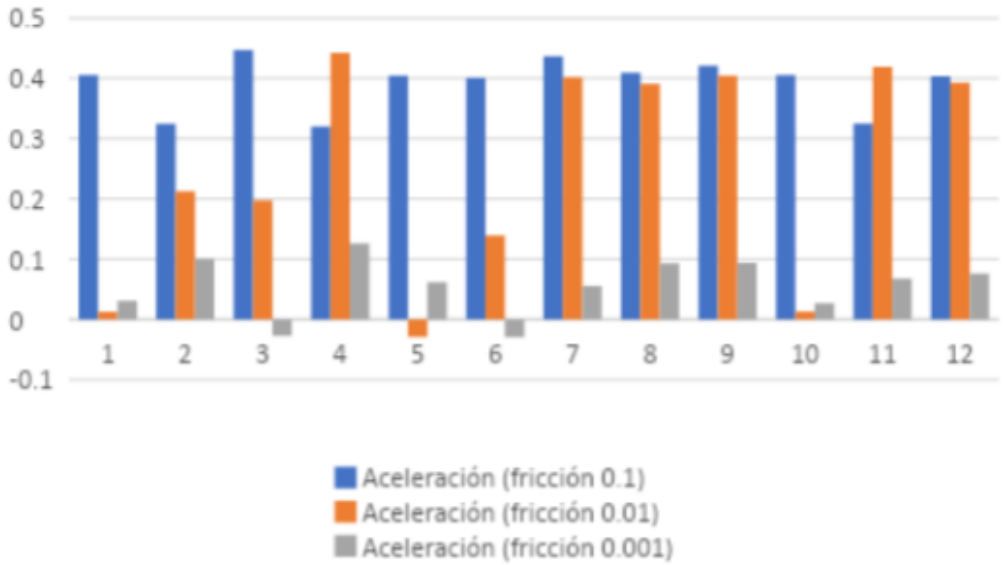
mBot subiendo una rampa con una fuerza máxima insuficiente



Giro de 90º es un escenario con una fricción muy baja (pista de hielo)

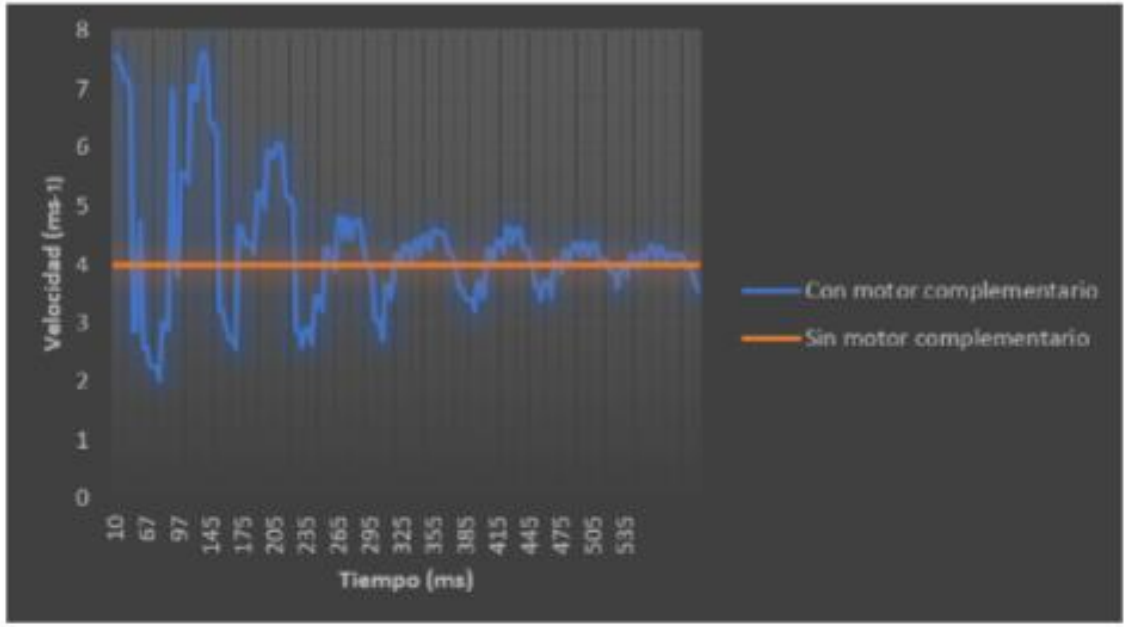


Variación de la aceleración en función de la fricción



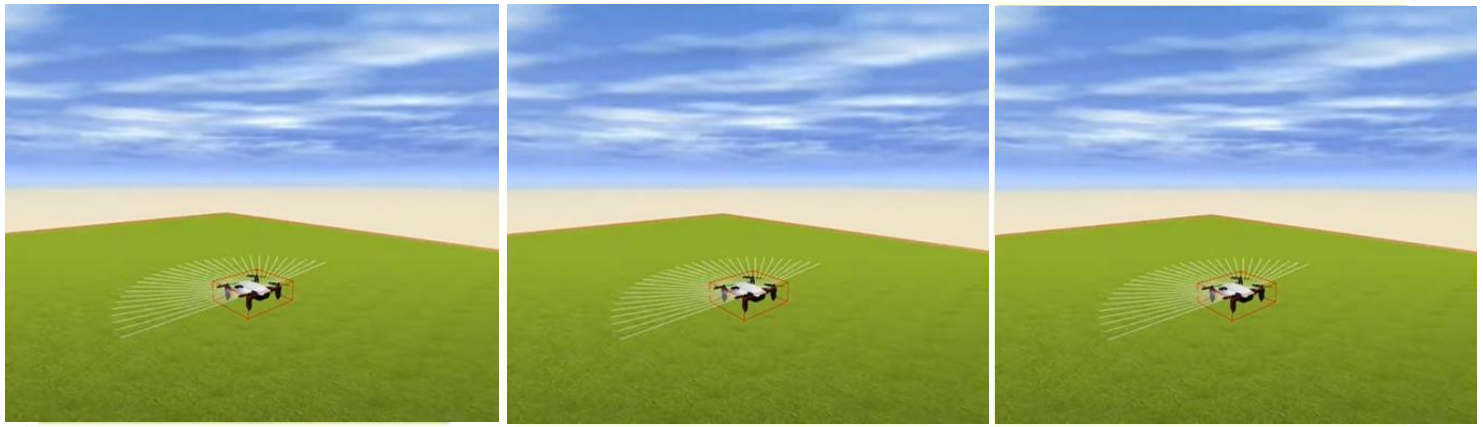
Controlador PD en velocidad del plano horizontal

Tiempo – Velocidad



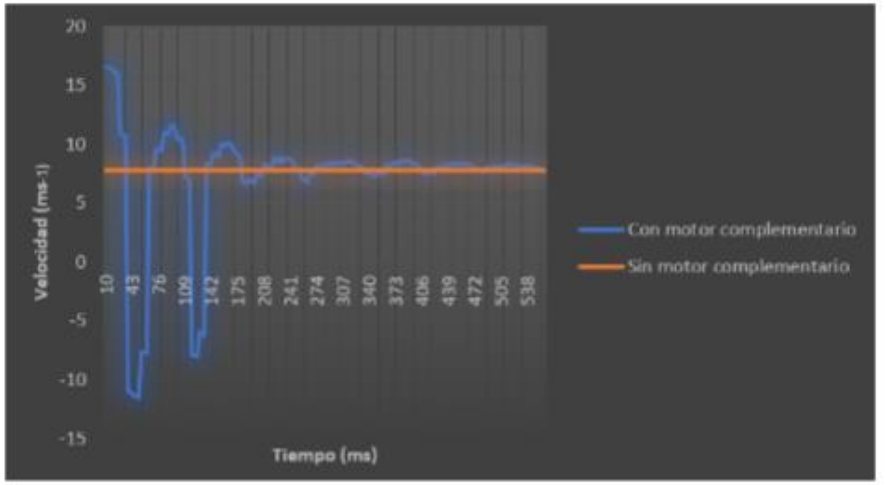
Simulación realista de drones

(a) Despegue del drone Tello de 1 Kg

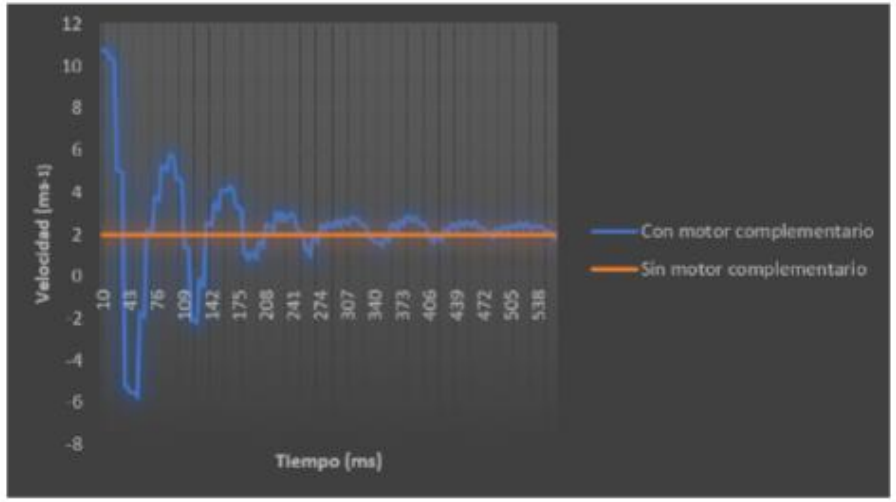


(b) Despegue del drone Tello de 100 Kg

Controlador PD en posición para la altura
Tiempo – posición



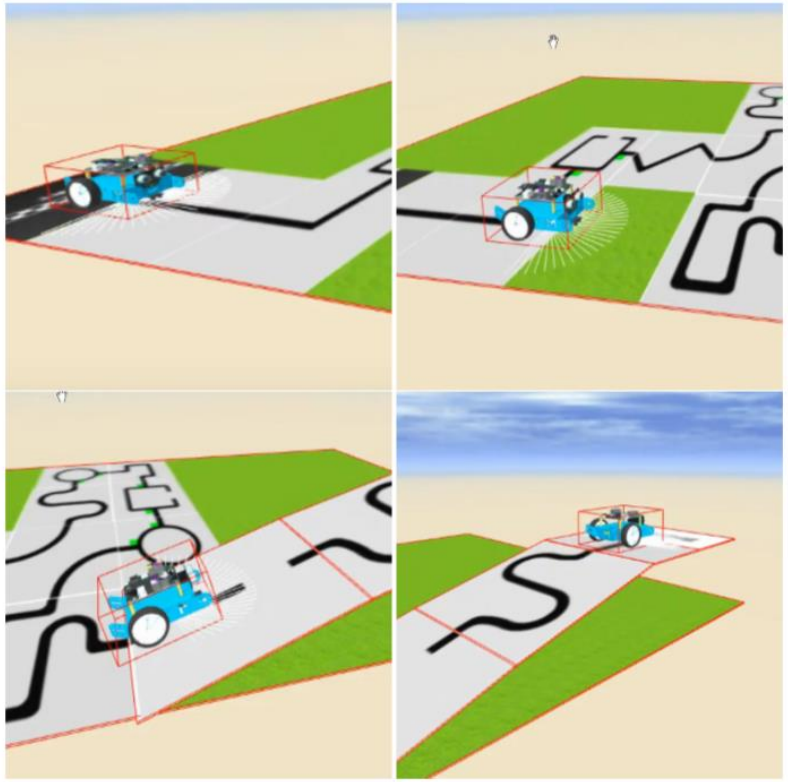
Controlador PD en velocidad del eje vertical
Tiempo – velocidad



Nuevos ejercicios con físicas realistas

Sigue-Líneas con rampa

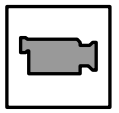
Aprovecha el motor de físicas complementario en la subida de la rampa.

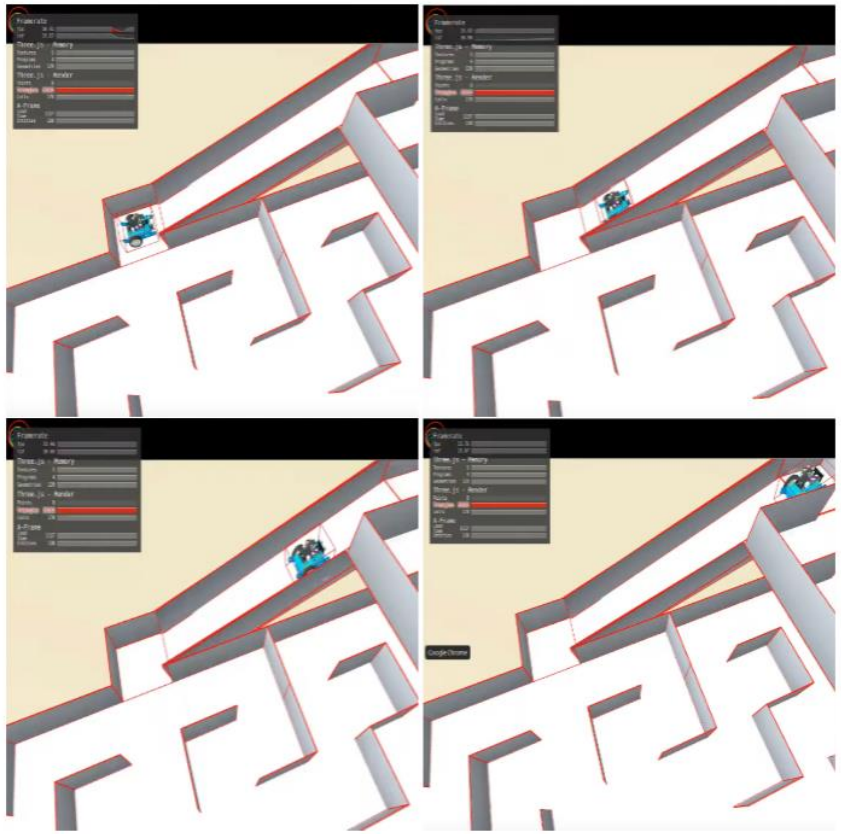


Parámetros del modelo de fuerzas	
mass	1
inertia	1.3
Fmax	10
Tmax	1
accelerationMax	10
angularAccelerationMax	0.77
linealSpeedMax	10
angularSpeedMax	5
Parámetros de *A-Frame*	
restitution	0.3
gravity	-9.8
friction	0.00003
linearDamping	-1.3
angularDamping	-1.3

Laberinto 3D para mBot

Aprovecha el motor de físicas complementario en la subida de la rampa.

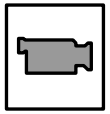




Parámetros del modelo de fuerzas	
mass	1
inertia	1.3
Fmax	10
Tmax	1
accelerationMax	10
angularAccelerationMax	0.77
linealSpeedMax	10
angularSpeedMax	5
Parámetros de *A-Frame*	
restitution	0.3
gravity	-9.8
friction	0.0005
linearDamping	-1.3
angularDamping	-1.3

Laberinto para drone

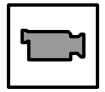
- Con señalización y sin señalización
- Aprovecha el motor de físicas complementario durante el vuelo del drone (controlador PD en velocidad) y mientras que el drone permanece quieto a una cierta altura (controlador PD en posición)

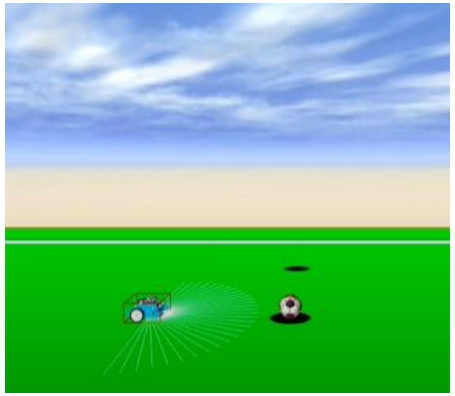


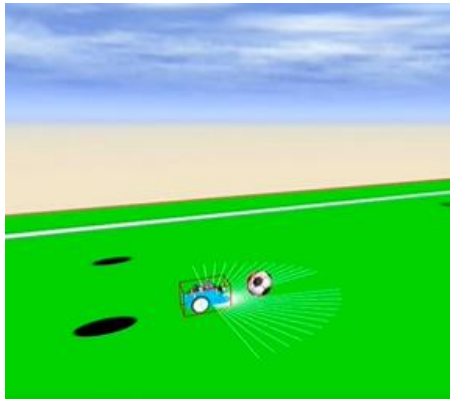
Parámetros del modelo de fuerzas	
mass	1
inertia	1.3
Fmax	10
Tmax	1
accelerationMax	10
angularAccelerationMax	0.77
linealSpeedMax	10
angularSpeedMax	5
Parámetros de *A-Frame*	
restitution	0.3
gravity	-9.8
friction	0.0000001
linearDamping	0.01
angularDamping	0.01

Fútbol competitivo

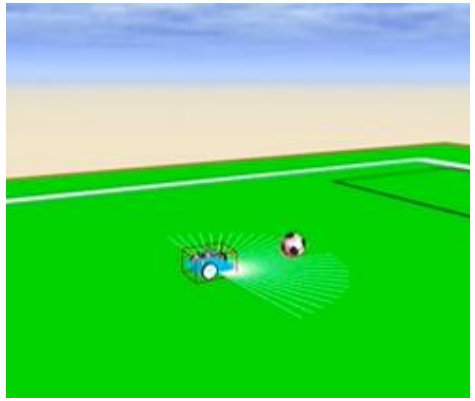
- Ejercicio competitivo uno contra uno
- Cuenta con un evaluador automático
- Saca provecho de las nuevas físicas al golpear el balón (colisión) y cuando el balón rueda por el suelo (fricción)

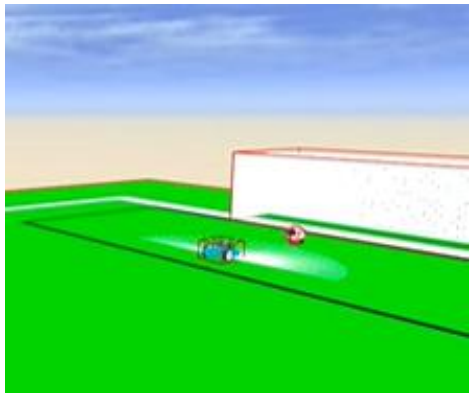
Sin el nuevo motor complementario 

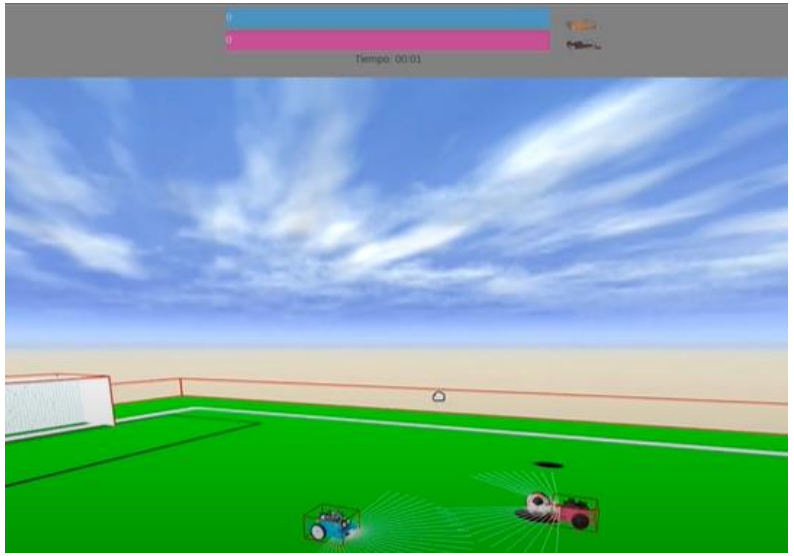


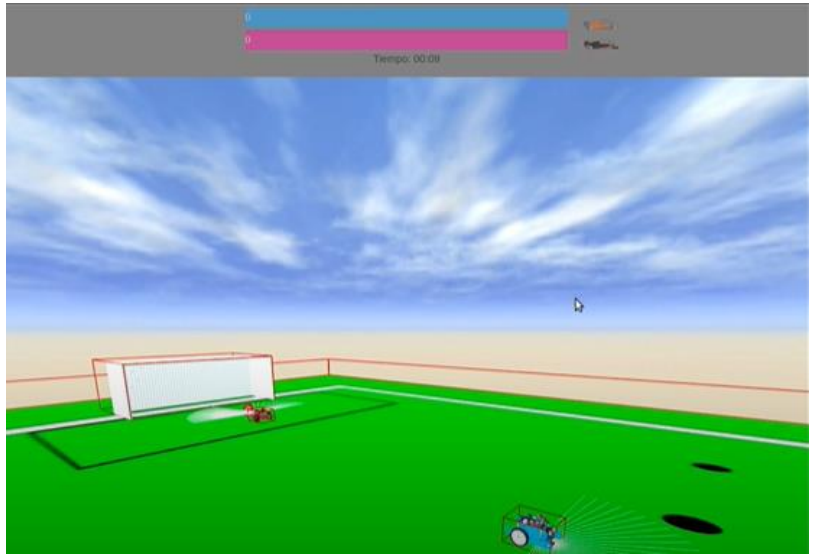


Con el nuevo motor complementario 









Parámetros del modelo de fuerzas	
mass	1
inertia	1.3
Fmax	10
Tmax	1
accelerationMax	10
angularAccelerationMax	0.77
linealSpeedMax	10
angularSpeedMax	5
Parámetros de *A-Frame*	
restitution	0.5
gravity	-9.8
friction	0.0005
linearDamping	0.01
angularDamping	0.01

Conclusiones

Conclusiones

- Creado un motor de físicas mejorado para *WebSim* basado en tecnologías web y que dota al simulador robótico de unas físicas más realistas ⟶ En producción ✓
- Robots con distinta masa ✓
- Coexiste con el motor *CANNON* (materializa gravedad, fricción y colisiones) ✓
- Nuevos ejercicios que aprovechan el nuevo motor de físicas ✓

Líneas futuras

- Ejercicios competitivos para cuatro jugadores 
- Nuevo motor de físicas *ammo.js* 
- Efectos de sonido a los ejercicios 

¡Muchas gracias!