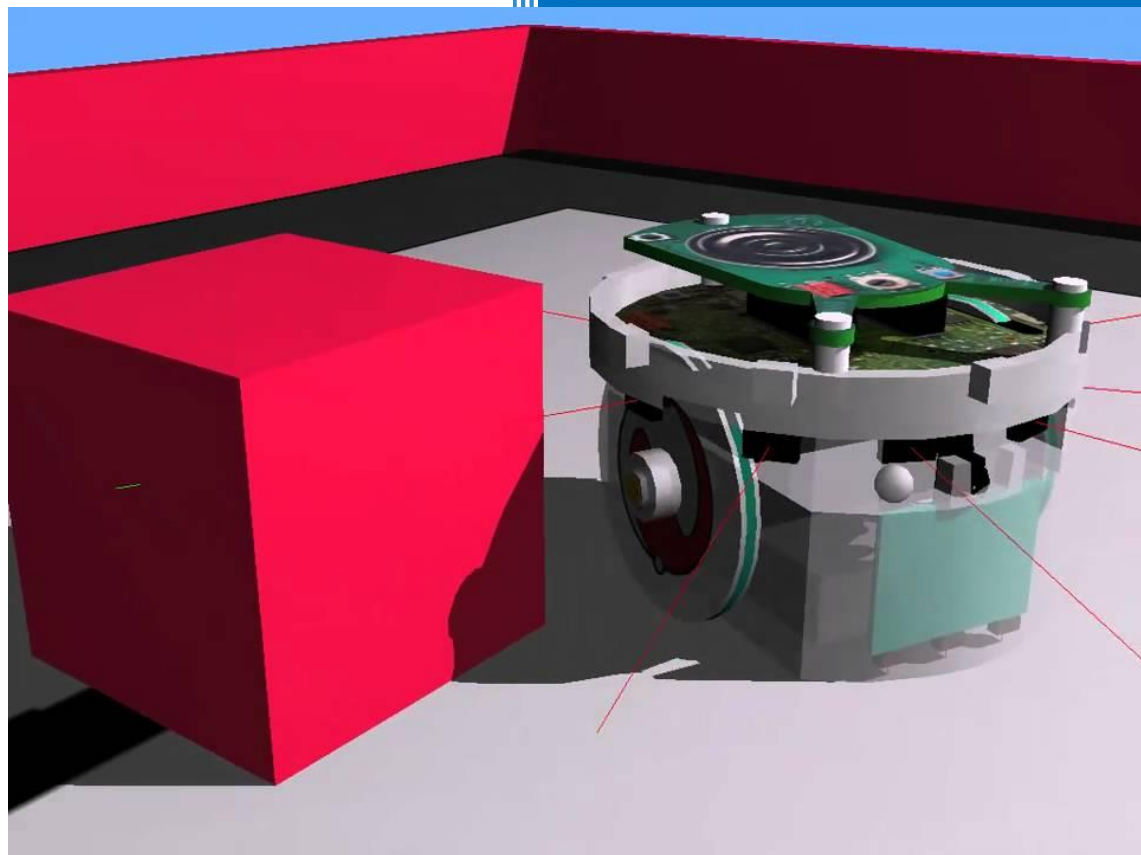


2020

Proyecto Robótica



Titulo: Robot IoT Inteligente

Docente: Balich Néstor

Integrantes del grupo: Balich
Franco y Gino Lucas

Sede: Centro

14-12-2020

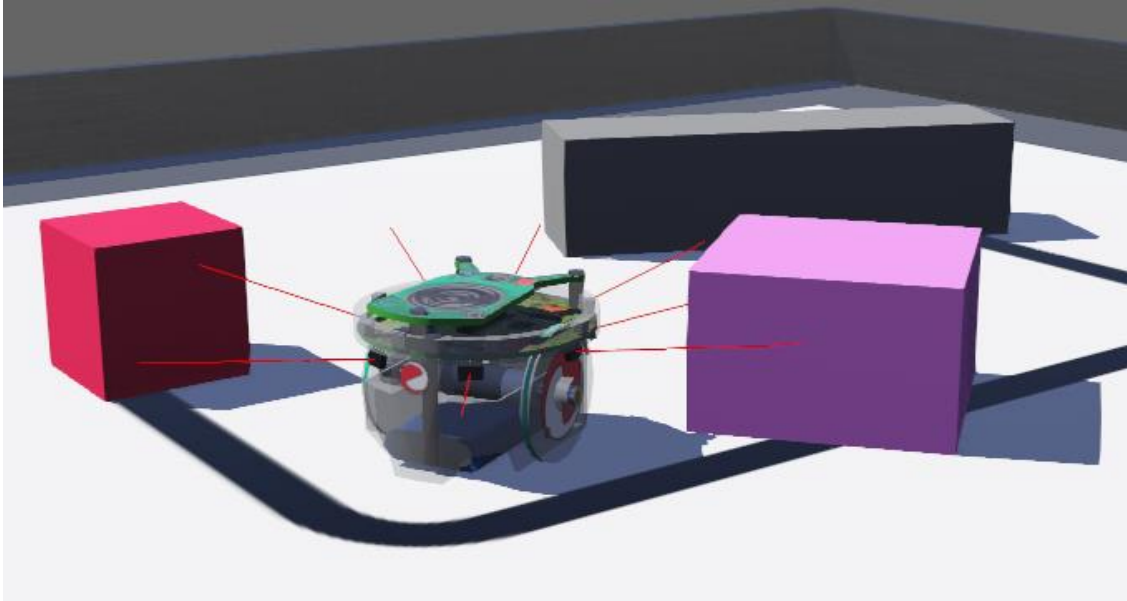
INDICE	
Objetivo del proyecto:.....	2
Preguntas clave:	2
Respuestas:	3
Etapas:.....	4
Material de investigación:	5
Webots:.....	5
MQTT:	6
El Modelo E-Puck:	7
Inteligencia Artificial:.....	8

Proyecto Final – Robótica

(Robot IoT Inteligente)

Objetivo del proyecto:

- Crear un robot que, mediante inteligencia artificial, sea capaz de realizar dos tareas básicas. Una es moverse solo y evitar obstáculos de forma dinámica y la segunda que sea controlado por IoT a través del protocolo de comunicación MQTT.



Preguntas clave:

- ¿Es posible implementar inteligencia artificial en un robot simple?
- ¿Qué ventajas puede darme la implementación de una inteligencia artificial en comparación a un algoritmo simple de evitar obstáculos?
- ¿Es difícil implementar una inteligencia artificial?
- ¿Qué lenguajes de programación son óptimos para la implementación de un proyecto de inteligencia artificial?
- ¿Es muy dificultosa su implementación en Python?
- ¿Existe una alternativa a tener que armarlo físicamente en el contexto del distanciamiento social?
- ¿Existe alguna herramienta alternativa a las vistas en clase durante la cursada?
- ¿Es posible comunicar a nuestro robot mediante algún servicio IoT?
- ¿Es posible enviarle información a nuestro robot por IoT?
- ¿Podemos cambiar como funciona nuestro robot mediante IoT?

Respuestas:

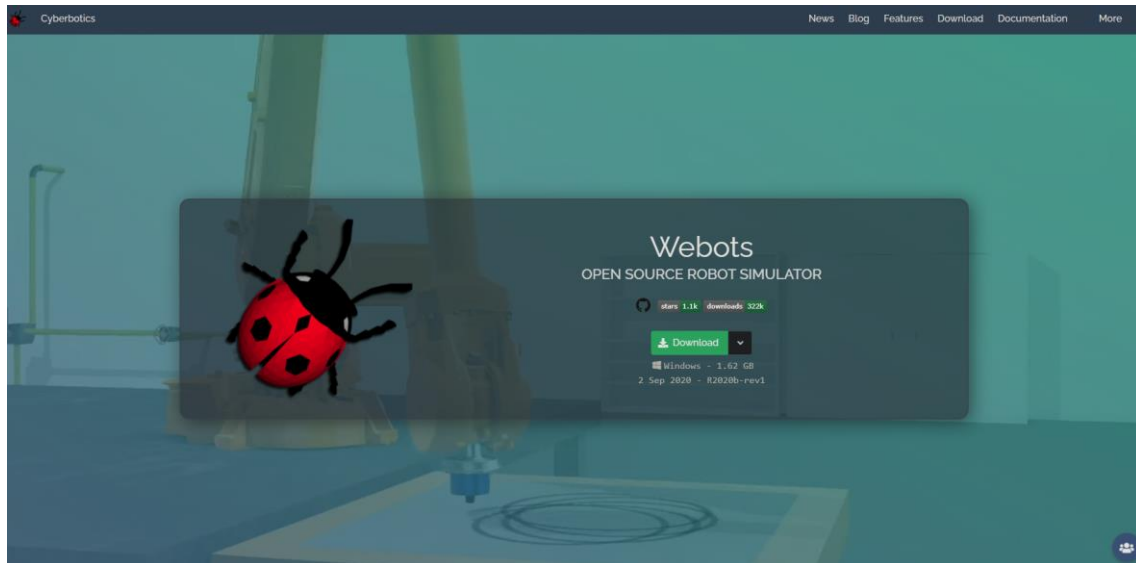
- Resulta que como de costumbre, el avance del tiempo que lleva a su lado a la evolución de las tecnologías no solo abarata su costo, sino que también surgen alternativas mucho más fáciles de aprender, implementar y utilizar.
- Una ventaja muy interesante que tiene la implementación de la IA en un robot simple es que el algoritmo en caso de querer agregarle o cambiarle funcionalidad, resulta mucho más dificultoso de hacer, en cambio en una implementación de IA resulta algo no tan dificultoso debido a que nos permite cierto grado de escalabilidad (dependiendo de la capacidad de nuestra implementación de IA). Una ventaja muy obvia es que la inteligencia artificial al estar entrenando/aprendiendo para cumplir su tarea conseguirá una precisión bastante alta (entre más complejo sea el proyecto, más útil será una implementación de IA), en cambio un algoritmo de esquivar obstáculos depende completamente de la calidad de este, lo cual hace que el rendimiento dependa de la calidad de los programadores.
- La respuesta corta es que no, pero entrando un poco en detalle, depende de la complejidad del dominio y el objetivo que la implementación de inteligencia artificial deba cumplir, en nuestro caso al ser una implementación no tan compleja, resulta algo no tan dificultoso.
- Algunos de los mejores lenguajes de programación para implementación de inteligencia artificial son: C++, MATLAB, R, Python y Smalltalk.
- No, como mencionamos anteriormente, debido al tamaño del proyecto, resulta algo no tan dificultoso, las dificultades aparecen si no se tiene mucho conocimiento del lenguaje o implementaciones de IA.
- Si, existen alternativas, como pueden ser los simuladores de robótica (existen muchos gratuitos fáciles de encontrar en internet) que nos permiten utilizar o armar un robot sin tener que estar cerca físicamente, adaptándonos al contexto del distanciamiento social.
- Si, como puede ser el caso de Webots o Coppelia.
- Si es posible, en nuestro caso particular utilizamos comunicación mediante MQTT, es un protocolo de red abierto OASIS y estándar ISO ligero, de publicación-suscripción que transporta mensajes entre dispositivos. El protocolo normalmente se ejecuta a través de TCP/IP; sin embargo, cualquier protocolo de red que proporcione conexiones bidireccionales ordenadas, sin pérdidas puede admitir MQTT. Está diseñado para conexiones con ubicaciones remotas donde se requiere un "pequeño espacio de código" o el ancho de banda de red es limitado.
- Si, mediante la implementación de MQTT podemos enviarle varios tipos de mensajes al robot de forma ilimitada y sencilla.
- Si, es posible, en nuestro caso somos capaces de cambiar la modalidad del robot entre manual y automático.

Etapas:

- Investigación de diferentes proyectos. (Balich Franco y Gino Lucas)
- Investigación de viabilidad de estos. (Balich Franco y Gino Lucas)
- Elección del proyecto más optimo e interesante. (Balich Franco y Gino Lucas)
- Investigación sobre proyectos y temáticas similares. (Balich Franco y Gino Lucas)
- Recopilación de información. (Balich Franco y Gino Lucas)
- Investigación sobre protocolos de comunicación IoT (MQTT). (Balich Franco)
- Investigación de paquetes NPM para Node.js (aedes). (Balich Franco)
- Desarrollo de la página web (interfaz con MQTT). (Balich Franco)
- Desarrollo del bróker MQTT y del servidor local web. (Balich Franco)
- Investigación de simuladores de robótica. (Gino Lucas)
- Recopilación de información del simulador de robótica Webots. (Gino Lucas)
- Pruebas en el simulador de robótica Webots. (Gino Lucas)
- Elección del robot y mundo para el proyecto. (Gino Lucas)
- Pruebas básicas de movimiento y sensado del robot en el simulador. (Gino Lucas)
- Integración cliente MQTT al simulador (Paho). (Balich Franco)
- Recopilación de información de redes neuronales implementadas en el ámbito de la robótica. (Balich Franco)
- Adaptación de redes neuronales programadas en Arduino para ser implementadas en Python. (Balich Franco)
- Integración de red neuronal al simulador Webots. (Balich Franco)
- Pruebas y corrección de errores en el aprendizaje de la red neuronal. (Balich Franco)
- Integración de la red neuronal del cliente MQTT y de los controles de dirección básico del robot. (Balich Franco)
- Corrección y toques finales al proyecto. (Balich Franco)

Material de investigación:

Webots:



Webots es una aplicación de escritorio de código abierto y multiplataforma que se utiliza para simular robots. Proporciona un entorno de desarrollo completo para modelar, programar y simular robots.

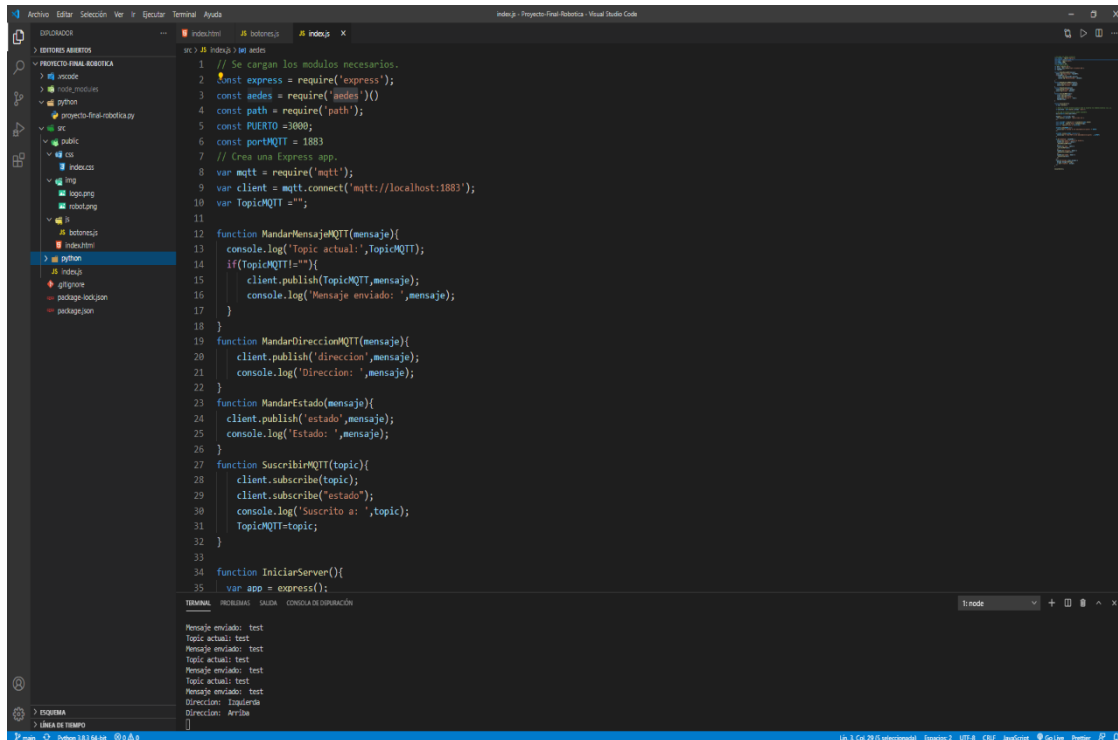
El programa Webots permite construir robots a través de la definición geométrica y dinámica de las partes que lo componen. Igualmente permite especificar colores y texturas para una mejor visualización.

Igualmente incluye una cantidad de sensores y actuadores de uso frecuente en robótica, con sus respectivos modelos dinámicos.

El control del robot puede ser escrito en C, C++, Java, Python, Matlab y ROS.

MQTT:

MQTT es un protocolo de mensajería estándar de OASIS para Internet de las cosas (IoT). Está diseñado como un transporte de mensajería de publicación / suscripción extremadamente liviana que es ideal para conectar dispositivos remotos con una huella de código pequeña y un ancho de banda de red mínimo. Hoy en día, MQTT se utiliza en una amplia variedad de industrias, como la automotriz, la fabricación, las telecomunicaciones, el petróleo y el gas, etc.



```
1 // Se cargan los modulos necesarios.
2 const express = require('express');
3 const aedes = require('aedes')();
4 const path = require('path');
5 const PUERTO = 3000;
6 const portMQTT = 1883;
7 // Crea una Express app.
8 var mqtt = require('mqtt');
9 var client = mqtt.connect('mqtt://localhost:1883');
10 var TopicMQTT = "";
11
12 function MandarMensajeMQTT(mensaje){
13   console.log('Topic actual:', TopicMQTT);
14   if(TopicMQTT!=""){
15     client.publish(TopicMQTT,mensaje);
16     console.log('Mensaje enviado: ',mensaje);
17   }
18 }
19 function MandarDireccionMQTT(mensaje){
20   client.publish('direccion',mensaje);
21   console.log('Direccion: ',mensaje);
22 }
23 function MandarEstado(mensaje){
24   client.publish('estado',mensaje);
25   console.log('Estado: ',mensaje);
26 }
27 function SuscribirseMQTT(topic){
28   client.subscribe(topic);
29   client.subscribe("estado");
30   console.log('Suscrito a: ',topic);
31   TopicMQTT=topic;
32 }
33
34 function IniciarServer(){
35   var app = express();
```

Terminal output:

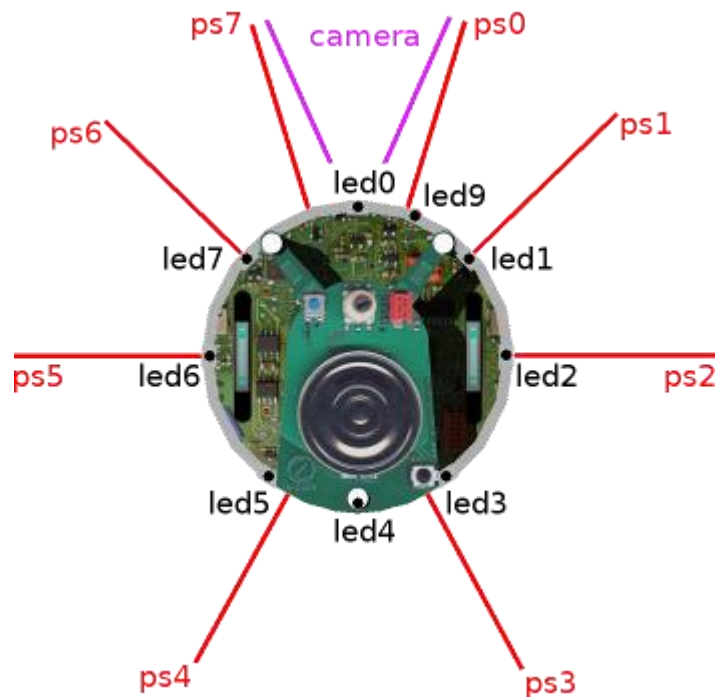
```
Mensaje enviado: test
Topic actual: test
Mensaje enviado: test
Topic actual: test
Mensaje enviado: test
Topic actual: test
Mensaje enviado: test
Direccion: Izquierda
Direccion: Arriba
```

El Modelo E-Puck:

E-puck es un robot móvil en miniatura desarrollado originalmente en EPFL con fines didácticos por los diseñadores del exitoso robot Khepera. El hardware y el software de e-puck es totalmente de código abierto, proporcionando acceso de bajo nivel a todos los dispositivos electrónicos y ofreciendo posibilidades de extensión ilimitadas.



El modelo incluye soporte para los motores de rueda diferencial (también se simulan los codificadores, como sensores de posición), los sensores infrarrojos para mediciones de proximidad y luz, el acelerómetro, el giroscopio, la cámara, los 8 LED circundantes, el cuerpo y los LED delanteros, la comunicación bluetooth (modelada con dispositivos Emisor / Receptor) y la extensión de sensores de tierra. Los otros dispositivos de e-puck aún no se simulan en el modelo actual.



El sitio web oficial de e-puck proporciona la información más actualizada sobre este robot. E-puck también está disponible para la compra de Cyberbotics Ltd.

E-puck fue diseñado para cumplir con los siguientes requisitos:

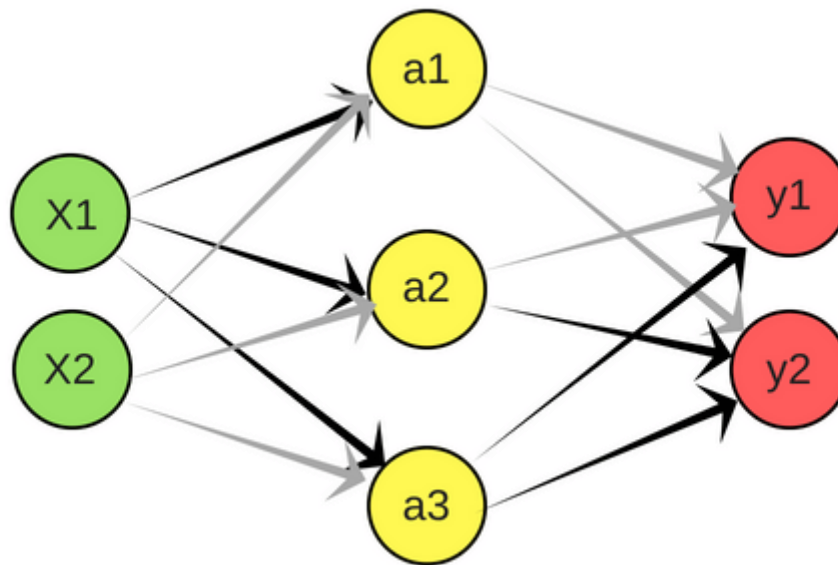
- Diseño elegante: la estructura mecánica simple, el diseño electrónico y el software de e-puck es un ejemplo de un sistema limpio y moderno.
- Flexibilidad: e-puck cubre una amplia gama de actividades educativas, ofreciendo muchas posibilidades con sus sensores, potencia de procesamiento y extensiones.
- Software de simulación: e-puck está integrado con el software de simulación Webots para facilitar la programación, simulación y control remoto del robot (físico).
- Fácil de usar: e-puck es pequeño y fácil de configurar en una mesa al lado de un ordenador. No necesita cables, proporcionando un confort de trabajo óptimo.
- Robustez y mantenimiento: e-puck es resistente bajo el uso de los estudiantes y es fácil de reparar.
- Asequible: la etiqueta de precio de e-puck es amigable con los presupuestos de cualquier universidad

Inteligencia Artificial:

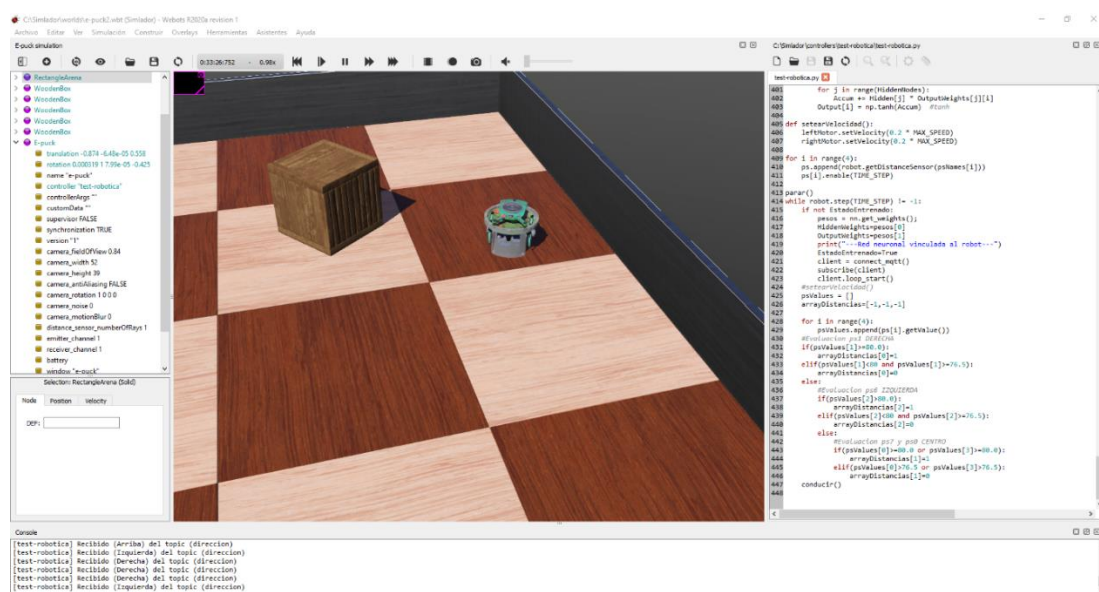
- Redes neuronales: Son algoritmos y estructuras inspirados en las funciones biológicas de las redes neuronales. Se suelen utilizar para problemas de Clasificación y Regresión, pero realmente tienen un gran potencial para resolver multitud de problemáticas. Son muy buenas para detectar patrones. Las Redes Neuronales Artificiales requieren mucha capacidad de procesamiento y memoria y estuvieron muy limitadas por la tecnología del pasado hasta estos últimos años en los que resurgieron con mucha fuerza dando lugar al Aprendizaje Profundo
- Aprendizaje Profundo: Son la evolución de las Redes Neuronales Artificiales que aprovechan el abaratamiento de la tecnología y la mayor capacidad de ejecución, memoria y disco para explotar gran cantidad de datos en enormes redes neuronales interconectarlas en diversas capas que pueden ejecutar en paralelo para realizar cálculos.
- Procesamiento del Lenguaje Natural: El Natural Language Processing es una mezcla entre DataScience, Machine Learning y Lingüística. Tiene como objetivo comprender el lenguaje humano. Tanto en textos como en discurso/voz. Desde analizar sintáctica ó gramáticamente miles contenidos, clasificar automáticamente en temas, los chatbots y hasta generar poesía imitando a Shakespeare. También es común utilizarlo para el Análisis de Sentimientos en redes sociales, (por ejemplo, con respecto a un político) y la traducción automática entre idiomas. Asistentes

como Siri, Cortana y la posibilidad de preguntar y obtener respuestas, o hasta sacar entradas de cine.

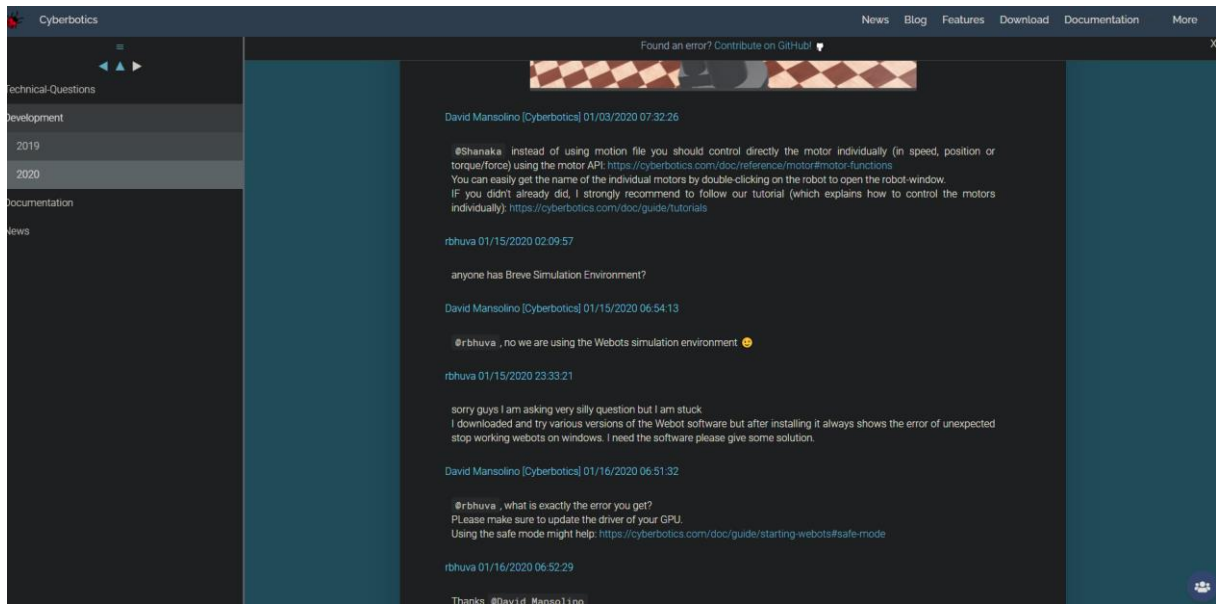
- Forward Propagation: nos referimos al recorrido de “izquierda a derecha” que hace el algoritmo de la red, para calcular el valor de activación de las neuronas desde las entradas hasta obtener los valores de salida.
- Al hacer backpropagation es donde el algoritmo itera para aprender. Esta vez iremos de “derecha a izquierda” en la red para mejorar la precisión de las predicciones. El algoritmo de backpropagation se divide en dos Fases: Propagar y Actualizar Pesos.



Aquí arriba podemos observar la estructura de nuestra red neuronal, tenemos 2 neuronas de entrada (x), 3 neuronas ocultas de activación y 2 de salida. Debajo podemos observar cómo queda nuestra implementación en el modelo e-puck.



Documentación de Webots:



Archivo con historial de conversaciones del canal de Discord de la comunidad de Webots

DistanceSensor Functions

```
wb_distance_sensor_enable
wb_distance_sensor_disable
wb_distance_sensor_get_sampling_period
wb_distance_sensor_get_value
```

C

C++

Python

Java

MATLAB

ROS

```
from controller import DistanceSensor

class DistanceSensor (Device):
    def enable(self, samplingPeriod):
    def disable(self):
    def getSamplingPeriod(self):
    def getValue(self):
    # ...
```

Description

enable, disable and read distance sensor measurements

The `wb_distance_sensor_enable` function allows the user to enable distance sensor measurements. The `sampling_period` argument specifies the sampling period of the sensor and is expressed in milliseconds. Note that the first measurement will be available only after the first sampling period elapsed.

The `wb_distance_sensor_disable` function turns the distance sensor off, saving computation time.

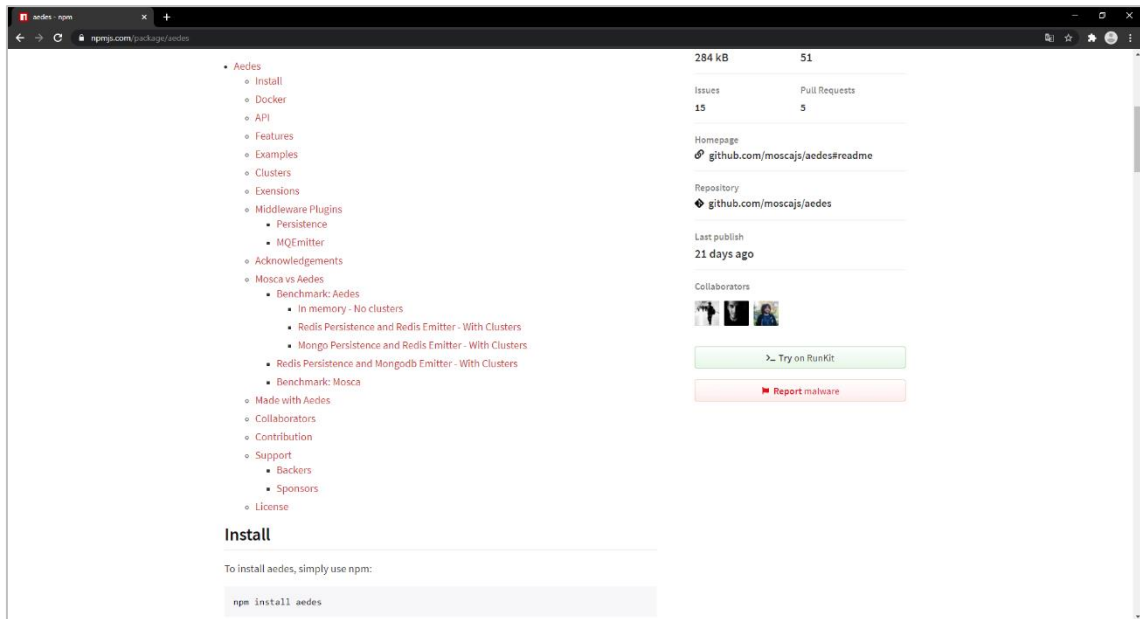
The `wb_distance_sensor_get_sampling_period` function returns the period given into the `wb_distance_sensor_enable` function, or 0 if the device is disabled.

The `wb_distance_sensor_get_value` function returns the last value measured by the specified distance sensor. This value is computed by the simulator according to the lookup table of the `DistanceSensor` node. Hence, the range of the return value is defined by this lookup table.

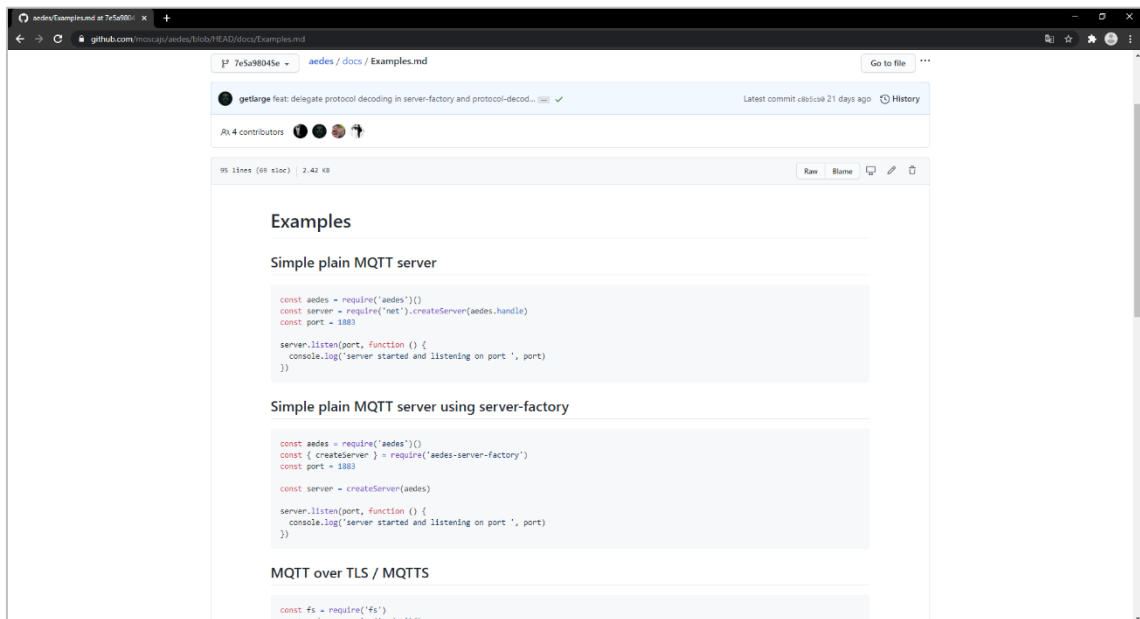
Documentación del sensor de distancia utilizado en Webots

Documentación MQTT

Aedes (Paquete de Node.Js que nos permite crear un broker MQTT)



Documentación oficial de npm sobre el paquete aedes



Github de la documentación de aedes