BePIM: Plataforma Inteligente Multipropósito basada en el reconocimiento de Beacons

ADAGIO, Matías, (mati.adagio@gmail.com)
CARBALLO CONTTI, Pablo, (pmccontti@gmail.com)
COLLAZO, Javier, (collazojn@gmail.com)
FORMICA VIDELA, German, (ger_formica@yahoo.com.ar)
GUASCO, Maximiliano Gabriel (guascomaxi@gmail.com)

Departamento de Ingeniería
Universidad Nacional de La Matanza
Florencio Varela 1903, San Justo, Buenos Aires, Argentina

Resumen

Se propone desarrollar una plataforma inteligente multipropósito sustentada en el uso de tecnologías IoT y señales Bluetooth a los efectos de simplificar las tareas de transporte de mercaderías en distintos ámbitos o negocios. La misma recibirá el nombre de BePIM. El sistema realiza un reconocimiento del área de desplazamiento mediante un entrenamiento asistido, en el cual captura parámetros tales como la distancia entre diferentes puntos de destino o de bifurcación, el sentido de direccionamiento, y la potencia emitida por dispositivos físicos denominados beacons.

Posteriormente el usuario podrá seleccionar el destino (Nodo) al que desea que la plataforma se dirija de manera automática y BePIM optará por el camino más óptimo para cumplir su objetivo sorteando los diferentes obstáculos que puedan aparecer.

Palabras clave: plataforma inteligente, Bluetooth, IoT, aplicación móvil, placa arduino, conexión Wi-Fi.

1. Introducción

En la actualidad, en los negocios/empresas u organismos impera la necesidad de administrar eficientemente los recursos humanos, toda vez que se requiere mayor eficiencia en la gestión de los recursos, a fin de evitar malgastar tiempo y costos innecesarios. Por otra parte no hay eficiencia en el traslado de mercaderías y las normas de seguridad cada vez son más estrictas, debiéndose evitar cualquier riesgo que se pueda ocasionar a la salud de las personas.

Actualmente se utilizan dispositivos de transporte que requieren esfuerzo y tiempo del personal para trasladar la carga hacia su destino; esto conlleva a riesgos de seguridad y administración ineficiente del tiempo productivo del personal.

Si bien existen soluciones parciales para el traslado de materiales, como el uso de maquinarias eléctricas, las cuales evitan o reducen el esfuerzo del personal aun así siguen requiriendo el apoyo humano. Asimismo, requieren de conocimientos especializados en el manejo de tales maquinarias. Otras alternativas, que también siguen los lineamientos, requieren de gran infraestructura y adaptación del lugar para su implementación.

Ante este escenario, se considera de gran valor la posibilidad de plantear un aporte tecnológico que acompañe apropiadamente esta evolución de la gestión del personal, diseñando y construyendo una plataforma inteligente multipropósito, facilitando las tareas del personal, gestionando con mayor eficiencia los recursos, brindando la seguridad adecuada, sin requerir conocimiento especializado y sin necesidad de cambios en la infraestructura de la locación.

2. Entorno de desarrollo

Al momento de elegir las herramientas necesarias para construir el desarrollo de nuestro sistema se decidió utilizar Arduino IDE 1.8.9, el cual es de código abierto, brindándonos una gran cantidad de material de soporte debido a la popularidad y cantidad de desarrolladores existentes en la red, que utilizan el sistema para IoT a lo largo del mundo resultando en una continua integración y desarrollo sostenido durante años. Para el desarrollo de la aplicación móvil se utilizó Android Studio 3.5, siendo software de código abierto y gratuito muy ampliamente utilizado en el mundo para el desarrollo de aplicaciones, a raíz de contar con una gran comunidad de soporte, ligado a la integración con servicios de la empresa Google tal

como Firebase, el cual proporciona una gran facilidad en el manejo de notificaciones y control de acceso al sistema. Se utilizará un Web Server programado en PHP, que será almacenado en la nube, para simplificar la comunicación con los demás sistemas a través de peticiones en lenguaje JSON, funcionando similar a una API. En cuanto a la persistencia de los datos, se decidió utilizar un sistema de base de datos MariaDB debido a su compatibilidad con el lenguaje PHP.

Como se dijo anteriormente, para el posicionamiento de la plataforma dentro del área de desplazamiento, se optó por utilizar dispositivos físicos Bluetooth (*Beacons)*. La principal razón por la cual utilizar este tipo de dispositivos, es su enorme practicidad para poder brindar un sistema auto-instalable, debido a su tamaño casi imperceptible.

3. Beacon

Para reforzar el direccionamiento de la plataforma, se decidió utilizar dichos Beacons. Los mismos permiten el intercambio de paquetes de datos entre dos terminales.

Los beacons tienen la particularidad de transmitir pequeños paquetes de datos bajo el protocolo de comunicación bluetooth 4.0, lo que los hace más eficientes y precisos en la transmisión de señal.

Figura 1: Beacon

Aprovechando esta particularidad, se decidió traducir la potencia RSSI emitida por el beacons, en la distancia métrica hacia el mismo. De esa manera es posible confirmar con exactitud la ubicación de la plataforma.

RSSI a Distancia

El indicador de potencia de señal recibida o RSSI por sus siglas en inglés (Received Signal Strength Indicator) que es una escala de referencia utilizado para medir la potencia en la antena del módulo receptor asociado al último paquete de información recibido. Este parámetro está ampliamente estudiado ya que también es utilizado en tecnología Wi-Fi.

De la teoría electromagnética es sabido que la potencia de la señal disminuye con el cuadrado de la distancia según lo indica la ecuación de Friis para transmisión en el espacio libre[1]. Sin embargo, existen ecuaciones para el parámetro RSSI basadas en datos empíricos[2] que establecen que la pérdida de propagación de la señal recibida por un nodo a una distancia d del nodo transmisor es:

$$P_L(d_i)[dB] = P_L(d_0)[dB] + 10n\log_{10}\frac{d_i}{d_0}$$

Donde PL(d0) es la pérdida de propagación a una distancia de referencia conocida d0 (*generalmente d0=1m*). n es una constante que depende del medio y de los obstáculos que existan entre los nodos. En el espacio libre n=2, sin embargo, en un entorno real la propagación de la señal se ve afectada por fenómenos de reflexión, difracción y dispersión debido a los obstáculos del entorno, por lo que debe ser medida empíricamente.

De la ecuación anterior se tiene que:

$$n = \frac{P_L(d_i) - P_L(d_0)}{10\log_{10}\frac{d_i}{d_0}}$$

Dicha ecuación permite la estimación de la constante n a partir de la medición de potencia en las antenas de los nodos y la distancia que los separa.

Finalmente, la potencia de señal recibida será dada por [3]:

$$RSSI[dBm] = -10n\log_{10} d + A[dBm]$$

Donde n es la constante de pérdida ya mencionada, d es la distancia entre el nodo transmisor y el receptor en metros y A es el valor de RSSI en la antena de un receptor a una distancia de 1 metro del transmisor.

Despejando se obtiene:

$$d = 10^{-\frac{RSSI-A}{10n}}$$

4. Software

En esta sección se exponen los algoritmos, procesos y aspectos involucrados para lograr el desplazamiento de la plataforma, desde que se inicia la solicitud de movimiento

hasta el arribo al destino, y proveer de todos los servicios requeridos en el sistema.

Este sistema está pensado para ser implementado en depósitos de mercadería o transporte de cargas, en donde se requiera optimizar el personal de trabajo así como los tiempos para realizar dichas tarea.

No obstante, debido a su gran versatilidad, el sistema puede ser implementado en diversos sitios, como ser: aeropuertos (*transportando equipaje*), restaurantes (*asistencia al cliente*) o incluso para el uso doméstico.

El funcionamiento del sistema puede dividirse en dos modos: Modo Entrenamiento, en el que el sistema registra las diferentes distancias y direcciones necesarias para el traslado de la plataforma entre dos nodos. Y el Modo Operativo, en el que la plataforma se dirige de manera autónoma a diferentes nodos eligiendo la ruta más óptima.

4.1. Modo Entrenamiento

El primer modo de funcionamiento consiste en tomar una medición de los pasos que deberá dar la plataforma para llegar de un nodo A a un nodo B, así como también identificar el ángulo de giro, la orientación y la potencia percibida de cada beacon.

Es indispensable en primer medida ubicar los beacons en los diferentes nodos o bifurcaciones dentro del espacio de desplazamiento.

Para definir, lo más exacto posible, la distancia entre dos nodos, se decidió utilizar diferentes patrones de medición. En primer lugar se optó por contar los pasos realizados por el motor hasta llegar a destino, no obstante, debido al desvío por irregularidades en la superficie, se decidió reforzar el direccionamiento, midiendo el ángulo declarado entre el Norte y el sentido del magnetómetro.

Si bien estos parámetros brindan una aproximación bastante exacta, los beacons nos proporcionan una mayor precisión y mayor flexibilidad en el manejo de rutas entre nodos. Pudiendo definir asi rutas más cortas y establecer mayores puntos de control durante el traslado de la plataforma.

El primer paso importante para el entrenamiento, es ubicar los beacons en diferentes puntos estratégicos dentro de un área de desplazamiento. Luego se procede a obtener la dirección MAC (*Media Access Control*) de cada dispositivo, la cual nos servirá para identificar unívocamente cada uno de ellos.

Para la obtención de la dirección MAC se decidió implementar dos librerías de Android que facilita su obtención mediante la lectura de un código QR.

Para que el sistema pueda obtener la potencia de los mismos, se utiliza una función compatible con el módulo NodeMCU ESP32 que recibe como parámetro la MAC del dispositivo y nos devuelve la potencia capturada del beacon más cercano a la plataforma.

La función de escaneo de beacon, se desarrolló utilizando diferentes librerías, las cuales proporcionan mayor facilidad para realizar dicha tarea [4].

Finalmente al momento de confirmar un punto de destino (*Nodo*), BePIM recopila los datos de cantidad de pasos dados por el motor hasta llegar a destino, los diferentes ángulos que fue tomando la plataforma durante su trayecto y por último la potencia leída de cada beacon al transitar por cada nodo.

Cabe destacar, que para obtener una respuesta rápida y una menor latencia en el envío de mensajes (*ya que es necesario obtener diferentes parámetros en diferentes tiempos*), se decidió que en el Modo Entrenamiento, la comunicación entre el sistema embebido y el dispositivo móvil, se realiza de forma directa. Es un caso particular en el que el sistema embebido trabaja como servidor.

4.2. Modo Operativo

El modo operativo permite al usuario enviar peticiones a al sistema y éste, de estar disponible, procederá a realizar la tarea solicitada.

Las peticiones de los diferentes usuarios se almacenan en el sistema bajo el concepto de Productor-Consumidor[2] (Figura 2), el cual permite sincronizar las diferentes peticiones y mantener la posición de la plataforma actualizada.

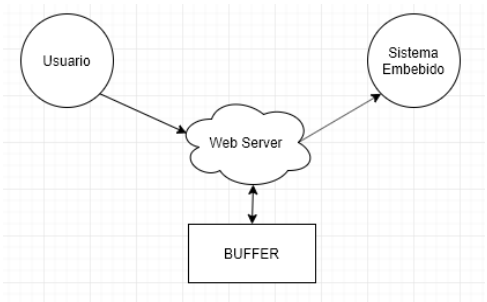

Figura 2: Productor-Consumidor

Es necesario tener en cuenta que en el manejo de peticiones, para esta ocasión y debido a que se dispone de un Web Server para administrar la comunicación entre el sistema embebido y el dispositivo móvil, el ESP32 mantendrá un pulling hacia el servidor cada 2 segundos.

Tal como se aclaró anteriormente, las peticiones se almacenan en una estructura Productor-Consumidor y se decidió representar a la misma como una tabla, como indica la Figura 3 en el servidor.

Al realizar una petición al servidor, la misma es almacenada en la Tabla de peticiones. Luego el sistema embebido obtiene la petición almacenada y borra el registro de la tabla.

idPeticion	idUsuario	idPlataforma	cod	dato	estado
121	1	3051573452	ENVIO	#	1

Figura 3: Tabla de peticiones

5. Registro y obtención de las peticiones

En la Figura 3 puede observarse el campo *dato* que contiene cada instrucción a realizar por la plataforma.

Se decidió implementar la ruta en formato de texto (*Array de char*), el cual se encuentra delimitado por dos caracteres diferentes (*inicio: # y fin:* -) para indicar el inicio y el fin de la instrucción en cuestión.

Durante el Modo entrenamiento, el sistema irá construyendo la ruta en función de las diferentes indicaciones que le brinde el usuario. Cada una de estas indicaciones queda representada por un carácter único e interpretable por el sistema embebido, seguido por un valor numérico el cual, según la indicación, puede representar números de pasos o grados de giro que debe realizar la plataforma y un último valor final representando la potencia emitida por el beacon. La combinación e indicación - valor es seguida por un carácter de separación "|" para facilitar el parseo del array

Cada indicación del usuario queda representada por los siguientes caracteres:

F = Avanzar hacia adelante

S = Detener el avance

D = Giro hacia la derecha sobre su propio eje

I = Giro hacia la izquierda sobre su propio eje.

P = Potencia Beacon

Ejemplo de instrucción: *#F400D15F200I15P60-*

Una vez generada la instrucción, el sistema embebido solo deberá interpretar en sentido inverso, cada uno de los componentes de dicha instrucción y de esa manera reconstruir la ruta realizada en el entrenamiento.

Durante las primeras pruebas del sistema, se han observado imprecisiones en las trayectoria de la plataforma debido a irregularidades en el suelo o incluso movimientos bruscos. Es por eso que se decidió reforzar el traslado de la plataforma con beacons, los cuales, como se dijo anteriormente, brindan una mayor precisión en cuanto a la ubicación de la plataforma sobre el área de traslado.

6. Algoritmo Dijkstra

Al momento de elegir una de las diferentes rutas que pueden ser establecidas entre dos nodos, BePIM optará por realizar la ruta más óptima. Para dicha tarea, se decidió implementar el algoritmo de Dijkstra[5] desarrollado en PHP.

Dicho algoritmo recibe como parámetro una matriz de costos (número de pasos dados por la plataforma) entre los diferentes nodos y devuelve como resultado un *array de char* con la concatenación de las rutas necesarias para completar el trayecto. Para facilitar el entrenamiento del sistema, se decidió implementar el algoritmo como grafo bidireccional; de esta manera, el usuario solo deberá entrenar la plataforma entre dos nodos, en un solo sentido de dirección.

desde	hasta	ruta	estado	costo
1	2	D0|F400|D90|F400|D90|F400|D90|P60	1	1200
2	4	D0|F500|D90|F500|D90|F500|D90|P60	1	1500
1	3	D0|F600|D90|F600|D90|F600|D90|P60	1	1800
3	5	D0|F700|D90|F700|D90|F700|D90|P60	1	2100
5	1	D0|F800|D90|F800|D90|F800|D90|P60	1	2400
5	2	D0|F900|D90|F900|D90|F900|D90|P60	1	2700

Figura 4: Matriz de costo almacenada en el Servidor

7. Evasión de Obstáculos.

Para la evasión de obstáculos se ha dotado a la plataforma de 4 sensores infrarrojos conectados a la placa Arduino MEGA, que permitirán la detección de un obstáculo en todas las direcciones posibles. Los mismos pueden ser regulados para la detección acorde a la distancia que uno

considere previamente, dependiendo el ambiente donde se implementara la solución.

Una vez detectado un obstáculo que impida el movimiento de la plataforma se dará proceso a dos procedimientos para saltear dicho obstáculo:

En principio, una vez detectado el obstáculo la plataforma procederá a realizar un giro de 90 grados, verificando a través del sensor ubicado al frente que no haya ningun obstaculo en el nuevo frente, inmediatamente procederá a avanzar la plataforma hasta que el sensor lateral deje de detectar el obstáculo original, avanzando una distancia extra acorde a la dimensión de la plataforma para evitar colisionar con el mismo en caso de girar. Cabe destacar que en forma previa, se establecerá una distancia máxima a recorrer para sortear cualquier obstáculo "en el momento", de tal manera, si la distancia recorrida excede esta medida previamente pre-establecida, se dará por anulada dicha ruta, reintegrandose en forma inversa hacia donde se inició el recorrido, iniciando el procedimiento en sentido inverso dando un nuevo giro de 180 grados, para verificar si se puede sortear el obstáculo desde el otro lado.

En caso de verificar una ruta posible para avanzar (a través de la detección del sensor lateral), se girará nuevamente 90 grados en la dirección que permite rodear el obstáculo y se continuará avanzando hasta que el sensor lateral dé por finalizado el obstáculo (siempre contemplando la máxima distancia a recorrer preestablecida), girando la plataforma 90 grados nuevamente para volver a su ruta original y retomar posteriormente la misma.

La ruta recorrida "en el momento" se registrará siempre a los fines de poder retornar a su última posición válida o para una vez sorteado el obstáculo recalcular la distancia necesaria faltante en ese trayecto, valiéndose además de las mediciones de señales beacon del próximo nodo.

En caso de que el procedimiento detallado no dé por efectiva ninguna ruta "establecida en el momento", la plataforma guardará el inconveniente eliminando la posible ruta entre dichos nodos de forma temporal y retornará al último nodo válido, donde procederá a efectuar un nuevo cálculo de costos sin este camino. De encontrar un nuevo camino procederá a realizarlo normalmente; en caso contrario avisará al usuario del inconveniente con la negativa del envío, pudiendo éste restablecer la plataforma a su origen o dejándola en el lugar hasta remover el obstáculo que no permite realizar la ruta y reiniciar el recorrido.

8. Hardware

Se decidió implementar el sistema como IoT (Internet of Things) para poder brindar un control del sistema a una mayor distancia, así como también, facilidad en el soporte remoto del sistema y un cierto resguardo de la información, debido al almacenando los datos de los usuarios en la nube. Otra de las ventajas, y quizá la más importante, es la facilidad de implementación, debido a que el usuario sólo necesitará disponer de un servicio de conexión a internet para poder utilizar el sistema.

Para lograr que el sistema funcione bajo el concepto de IoT, se decidió complementar al Arduino con un módulo NodeMCU ESP32[7], el cual, además de conexión a internet, permite un gran manejo de la información, siendo éste compatible con estructuras tipo JSON y reconocimientos de bluetooth 4.0.

Figura 5 : NodeMCU ESP32.

En cuanto al Arduino, la mejor opción fue utilizar la versión Mega debido que resulta óptimo en cuanto a la cantidad de entradas y salidas digitales (total 54) de las cuales 15 son PWM y 16 entradas analogicas necesarias para los sensores y actuadores, con una memoria flash de 256 Kb de espacio para representar el código, resultando 8 veces superior al de Arduino UNO, denostando mayores prestaciones por sobre otras placas, siendo el costo razonablemente económico en cuanto a las prestaciones que brinda.

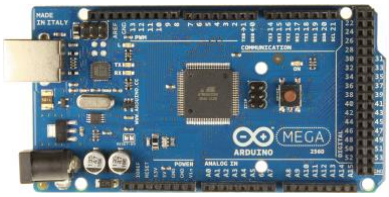

Figura 6 : Arduino Mega.

Para la plataforma física se utiliza una estructura en madera MDF de diseño circular con las ruedas en el centro para facilitar rotaciones en su propio eje evitando colisiones al girar y abarcando siempre la misma superficie. Se busca disminuir el peso de la misma para que no afecten la capacidad de carga en peso, tratándose de un prototipo.

Los motores que se han utilizado para traccionar las ruedas de la plataforma son motores paso a paso Nema 17 de alto torque (7.1kgcm) que cuenta con la fuerza suficiente para su propósito.

Figura 7 : Motor Nema 17 de alto torque.

Tanto la alimentación de los circuitos electrónicos como la de los motores se realizarán por medio de una batería utilizada comúnmente en alarmas domiciliarias con 12 volt / 7 amper, para dotar de autonomía suficiente a la totalidad de la plataforma.

Figura 4 : Bateria 12 volt 7 Ampere

9. Comunicación con NodeMCU ESP32

La comunicación entre el modulo ESP32 y Arduino, se realiza por dos medios diferentes según la tarea que se esté realizando. Para el envío de la ruta generada (En formato *array de char*) se decidió utilizar el puerto Serie, ya que brinda la posibilidad de transmitir diferentes tipos de caracteres. Por otro lado, las instrucciones directas de del usuario (F, S, D e I) se envían utilizando 3 pines digitales para representar cada instrucción como un valor binario, quedando representada de la siguiente manera:

S: 0 0 0

F: 0 0 1

D: 0 1 0

I: 1 0 0

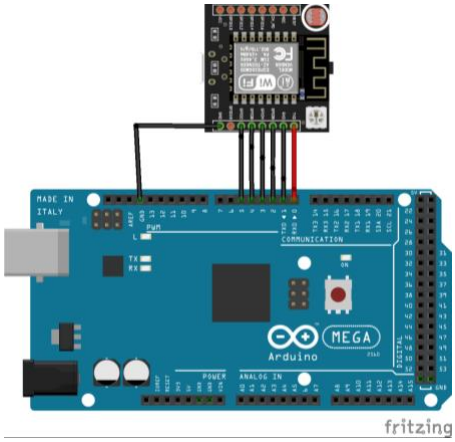

Figura 8: Esquema de conexión Arduino - ESP32

Tal como indica la Figura 8 puede observarse que el modulo ESP32 y el Arduino, comparten el pin GND, esto es para unificar el canal de transmisión y evitar que se pierda información.

Peticiones POST

Para enviar y recibir información del usuario, se decidió implementar la librería *ArduinoJson.h* ya que permite el manejo de estructuras de datos del tipo JSON. Las mismas son enviadas mediante peticiones POST al sistema embebido, el cual, decodifica el mensaje y envía el texto simple directamente al arduino.

Durante el entrenamiento del sistema, el módulo WIFI, se comporta como Access Point, recibiendo las peticiones POST directamente de la aplicación móvil. Para lograr este comportamiento, se utilizo la libreria *WebServer.h.*[8]

Para dar respuesta al usuario, la librería *WebServer.h* permite utilizar el método **server.on("/training", HTTP_POST, entrenamientoPOST);** el cual recibe como parámetro una extensión de URL a la que el usuario podrá realizar la petición correspondiente, seguido del tipo de petición (POST) y por último la función a ejecutar en el caso de recibir alguna petición.

Durante el modo operativo, el módulo ESP 32 con comunicaciones a través de señal WIFI, estará realizando peticiones POST al servidor cada 2 segundos esperando a que algún usuario haya solicitado realizar alguna tarea específica. Para realizar dichas peticiones, se utilizó la librería *WiFiClient.h,* la cual permite el envío de datos a una URL especificada previamente.

10. Conclusion

Para la construcción de este proyecto, se propuso obtener un sistema capaz de realizar el traslado de productos desde

un lugar a otro en forma automática, sin precisar la intervención humana en el manejo de la plataforma, de forma inteligente debido al cálculo de la ruta más eficiente y multipropósito ya que se puede adaptar acorde a las necesidades de cada negocio, sin grandes inversiones de infraestructura, valiéndose de tecnología hardware bluetooth de bajo costo existente en el mercado (beacons). Consideramos clave que el sistema sea auto-instalable y no requerir conocimiento especializado para su instalación, con un fácil aprendizaje de rutas necesarias. Los problemas más complejos radicaron en dar una precisión en la medición de las distancias y ángulos en el interior de un recinto. Otro gran inconveniente que se presentó es la propagación de errores mecánicos al efectuar las mediciones de recorridos almacenados durante el modo de entrenamiento, como así también los errores mecánicos en su ejecución durante el uso efectivo. El apoyo a través de la medición de señales de beacons nos dio un mayor nivel de precisión y doble verificación de las distancias y proximidades al objetivo en cada ubicación, con la consecuente reducción de errores mecánicos y reajuste de distancias acorde a las señales medidas. Cabe destacar que está proyectado seguir avanzando en esta línea de desarrollo para incluir mejoras de rendimiento, precisión, y eficiencia.

11. **Referencias**

[1] C. A. Balanis (2005). Antenna Theory, Analysis and Design. Wiley-Interscience. 3º Ed, CAP 2, pag 94.

[2]Oguejiofor O.S., Okorogu V.N., Adewale Abe, Osuesu B.O (2013). Outdoor Localization System Using RSSI Measurement of Wireless Sensor Network. International Journal of Innovative Technology and Exploring Engineering (IJITEE). ISSN: 2278-3075, Volume-2, Issue-2.

[3]E. Lau, B. Lee, S Lee, W. Chung (2008). Enhanced RSSI-Based high accuracy real-time user location tracking systems for indoor and outdoor environments. International Journal On Smart Sensing And Intelligent System, VO L. 1, NO. 2, 534-548.

[4]https://github.com/nkolban/ESP32_BLE_Arduino/blob/master/src/BLEDevice.h

[5]Silva, Manuel 1985 Editorial AC Madrid ISBN: 84-7288-045-1. Las Redes de Petri : en la Automática y la Informática; 1ª ed.

[6]Alvaro H. Salas S. Acerca del Algoritmo de Dijkstra

[7]Aitor Carricondo Monter Desarrollo de un sistema de monitorización domiciliaria basado en la plataforma NodeMCU V3

[8]https://github.com/espressif/arduino-esp32/tree/master/libraries/WebServer