UNIVERSIDAD POLITÉCNICA SALESIANA
SEDE QUITO

CARRERA:
INGENIERÍA ELECTRÓNICA

Trabajo de titulación previo a la obtención del título de:
INGENIEROS ELECTRÓNICOS

TEMA:

ALGORITMO PARA LA PREVENCIÓN DE COLISIONES DURANTE EL PROCESO DE TRANSPORTE Y CLASIFICACIÓN DE OBJETOS SOBRE UNA BANDA TRANSPORTADORA OMNIDIRECCIONAL CELULAR

AUTORES:

PEDRO GABRIEL HERNÁNDEZ BURGOS

CRISTIAN ALEXANDER TAMAYO TINAJERO

TUTOR:

CARLOS GERMAN PILLAJO ANGOS

Quito, julio del 2020

CESIÓN DE DERECHOS DE AUTOR

Nosotros Pedro Gabriel Hernández Burgos y Cristian Alexander Tamayo Tinajero con documentos de identificación Nº 1723852883 y Nº 1725405409 respectivamente, manifestamos nuestra voluntad y cedemos a la Universidad Politécnica Salesiana la titularidad sobre los derechos patrimoniales en virtud de que somos autores del trabajo de titulación intitulado: **"ALGORITMO PARA LA PREVENCIÓN DE COLISIONES DURANTE EL PROCESO DE TRANSPORTE Y CLASIFICACION DE OBJETOS SOBRE UNA BANDA OMNIDIRECCIONAL CELULAR"**, mismo que ha sido desarrollado para optar por el título de: Ingeniero Electrónico, en la Universidad Politécnica Salesiana, quedando la Universidad facultada para ejercer plenamente los derechos cedidos anteriormente.

En aplicación a lo determinado en la Ley de Propiedad Intelectual, en nuestra condición de autores nos reservamos los derechos morales de la obra antes citada. En concordancia, suscribimos este documento en el momento que hacemos entrega del trabajo final en formato digital a la Biblioteca de la Universidad Politécnica Salesiana.

Pedro Gabriel Hernández Burgos
C.I. 1723852883

Cristian Alexander Tamayo Tinajero
C.I. 1725405409

Quito, julio del 2020.

DECLARATORIA DE COAUTORÍA DEL DOCENTE TUTOR

Yo declaro que bajo mi dirección y asesoría fue desarrollado el Artículo Académico, "ALGORITMO PARA LA PREVENCIÓN DE COLISIONES DURANTE EL PROCESO DE TRANSPORTE Y CLASIFICACION DE OBJETOS SOBRE UNA BANDA OMNIDIRECCIONAL CELULAR" realizado por: Pedro Gabriel Hernández Burgos y Cristian Alexander Tamayo Tinajero, obteniendo un producto que cumple con todos los requisitos estipulados por la Universidad Politécnica Salesiana, para ser considerados como trabajo final de titulación.

Quito, julio del 2020.

Carlos German Pillajo Angos

C.I. 1709255119

Algoritmo para la prevención de colisiones durante el proceso de transporte y clasificación de objetos sobre una banda omnidireccional celular

Pedro Hernández
Ingeniería Electrónica
Universidad Politécnica Salsiana
Quito, Ecuador
phernandezb@est.ups.edu.ec

Cristian Tamayo
Ingeniería Electrónica
Universidad Politécnica Salesiana
Quito, Ecuador
ctamayot1@est.ups.edu.ec

Carlos Pillajo
Ingernieria Electrónica
Universidad Politécnica Salesiana
Quito, Ecuador
cpillajo@ups.edu.ec

***Resumen*—En este artículo se presenta el desarrollo del algoritmo para evasión de colisiones de una banda transportadora omnidireccional celular compuesta por 17 células hexagonales regulares. El algoritmo basado en la teoría de grafos que, junto con la adición de planificación de rutas, formulan posibles rutas de traslado de los objetos. Finalmente, mediante la aplicación del algoritmo de prevención de colisiones y combinado con la simulación en la herramienta de software *Matlab* especifica la ruta de transporte libre de colisión de objetos sobre la banda. La formulación es válida para el prototipo de banda omnidireccional celular desarrollada y con las dimensiones de objetos descritos en el diseño. Sin embargo, puede ser acoplado a cualquier tipo de banda que cumpla las condiciones de una banda omnidireccional celular.**

Palabras Clave—banda transportadora, omnidireccional, planificación, ruta, Dijkstra, algoritmo, grafo.

***Abstract*— This article presents the development of the algorithm for collision avoidance of a cellular omnidirectional conveyor belt made up of 17 cells. The algorithm based on graph theory that together with the route planning addition formulates possible routes for moving objects. Finally, this combined with the simulation in the *Matlab* software tool of the different cases in which objects enter the conveyor belt; specifies the collision-free transport path of objects on the band. This is valid for the prototype of the cellular omnidirectional band developed and with the dimensions of the objects described in the design. However, it can be coupled to any type of band that meets the conditions of a cellular omni-directional band.**

Keywords—conveyor belt, omnidirectional, path planning, Dijkstra, algorithm, graph.

I. INTRODUCCIÓN

Los sistemas de clasificación, vienen desempeñando un rol muy importante en los diferentes procesos industriales y esta se debe a varias razones entre las que se destacan las grandes distancias a las que se efectúa el transporte de diversos materiales como minerales, vegetales, combustibles, fertilizantes, suministros de construcción, etc.[1]

El sistema omnidireccional posee varios grados de libertad que facilitan el posicionamiento de los elementos transportados, por lo que debe existir una sincronización total del sistema para seguir una de entre todas las posibles rutas preestablecidas por las cuales pueden ser enviadas los objetos desde un mismo punto. Estos sistemas se han desarrollado con el fin de realizar una clasificación ágil con reducción de espacios, haciendo que los procesos de transporte permitan realizar movimientos laterales a medida que son guiados en la banda, a lo contrario de las bandas transportadoras convencionales que realizan movimientos lineales y unidireccionales [2].

Tomando en cuenta que dentro de la estructura de la banda omnidireccional celular se puede disponer de varias rutas para realizar tanto la función de transporte como de clasificación de un objeto así como la existencia de varias entradas y salidas en la banda se pueden dar situaciones de colisión entre dos o más objetos ocasionando daños a los mismos, también se pueden generar cuellos de botella que detengan o hagan más lento a un proceso de producción por lo cual es necesario disponer de un algoritmo que detecte con anticipación el posible punto de colisión y genere una ruta alterna para que los objetos lleguen a su destino sin inconvenientes[3].

Este artículo describe una solución viable a la prevención de colisiones en una banda transportadora omnidireccional celular. Todo esto a partir de la elaboración de un algoritmo basado en planificación de rutas e implementado a un prototipo de banda transportadora omnidireccional celular de dimensiones de 640 mm. de ancho por 460 mm. de largo y 135 mm. Este prototipo consta de 3 entradas y 2 salidas previamente definidas, por donde ingresarán/saldrán objetos (cajas de 150 mm de ancho por 150 mm de largo y 90 mm de alto) con una masa de 400 gr.

El contenido del presente documento está organizado de la siguiente forma. En la sección II, se describe las herramientas y métodos necesarios. En la sección III, se detalla el diseño y elaboración del algoritmo junto con el prototipo de banda. En la sección IV, se muestra los resultados obtenidos. Finalmente, en la sección V se proporciona las conclusiones generadas.

II. MARCO METODOLÓGICO

A. *Banda Transportadora Omnidireccional Celular*

Consisten en una banda modular con rodillos de giro libre o la combinación de llantas omnidireccionales. La banda se encarga del transporte, rotación, alineación, desvío, conmutación, clasificación entre otros de objetos. Al ser un dispositivo polifuncional, ayuda en el ahorro de compra de maquinarias tales como selectoras, direccionadoras, entre otras. Dentro de este tipo de bandas existe las denominadas celulares, las cuales constan de varios módulos hexagonales, cada módulo contiene ruedas omnidireccionales que son controladas individual y selectivamente por un motor eléctrico. La disposición de las ruedas, así como, el control selectivo de los accionamientos permite trasladar y posicionar varios objetos simultánea e independientemente en cualquier pista. Al trasladar varios objetos a la vez, existe la posibilidad de generar posibles colisiones en dicha banda, razón por la cual, la prevención de colisiones de objetos dentro de una banda omnidireccional celular permite explorar nuevas técnicas de evasión, permitiendo así la apertura de ejemplificar diversos algoritmos de control de rutas dentro de la misma.

B. Planificación de rutas

La planificación de rutas o path planning determina la secuencia de acciones necesarias para la obtención de la ruta mínima continua entre un lugar de origen hacia uno de llegada[4].

Son posibles diferentes tipos de rutas, dependiendo del caso específico. Por ejemplo, para manipuladores industriales, la ruta estándar generalmente se define por la geometría de la tarea a realizar, que se define de forma estática. En aplicaciones más avanzadas, generalmente en entornos dinámicos, se pueden agregar algunas características adicionales, como la necesidad de evitar obstáculos automáticamente[4].

Es importante recalcar su gran importancia en el campo de la robótica ya que tiene como objetivo encontrar el movimiento sin colisión entre una configuración inicial (inicio) y una configuración final en un entorno ya definido. Los algoritmos de planificación de rutas generalmente se dividen en tres categorías, según las metodologías utilizadas para generar la ruta geométrica, y estas son:

a) Técnica de hojas de ruta: Se define debido a la necesidad de seleccionar el mejor camino para trazar una ruta continua desde el origen hasta el objetivo final dependiendo del espacio y la configuración aplicada a la misma. Es importante dividir en tantas regiones sea posible el espacio o región a trabajar para determinar los puntos más cercanos entre objetos y elegir la ruta más corta.

b) Métodos de Potencial Artificial: Son utilizados en la planificación local en entornos desconocidos, por ejemplo, el sorteo en tiempo real de obstáculos o de los que no se tenga constancia. [5]

c) Algoritmos de descomposicion celular: Este tipo de algoritmo se fundamenta en la descomposición en celdas en las que posiblemente puede viajar algún objeto. Por tanto, se puede conocer el espacio libre o celda disponible para el siguiente movimiento.

Con la descomposición de celdas, el planificador de rutas tiene como objetivo encontrar un grafo de adyacencia que indique cuales celdas libres de obstáculos comparten una frontera común y así identificar el espacio libre(celdas) por donde se puede desplazar un objeto[6].

C. Representacion mediante Grafos

Un grafo es un conjunto de objetos llamados nodos entrelazados mediante enlaces denominados arcos que permiten representar relaciones binarias entre elementos de un conjunto. [7]

Para adquirir esta representación, cada objeto se representa mediante un vértice y cada par se lo asocia un arco[8].

D. Grafos Dirigidos

Este tipo de grafo se encuentra compuesto por un conjunto de *N* nodos y un conjunto *A* de arcos cuyos elementos son pares ordenados de distintos nodos.[9]

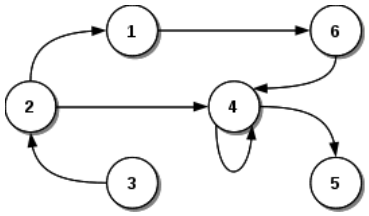

Fig. 1. Representación de un grafo dirigido.

Como se observa en la "Fig. 1", un grafo es representado mediante las siguientes especificaciones. Cada uno de los vértices o nodos tiene un identificador único (1,2,3,4,5,6). Un arco se encuentra conformado por la arista entre dos vértices, por ejemplo (2→ 1). Además, en grafos dirigidos, esta notación hace relevancia desde y hacia dónde se dirige el arco.

- *Red Dirigida:* Consiste en un grafo dirigido cuyos arcos presentan valores numéricos que se denominan pesos.
- *Ruta:* Es un camino sin repetición de nodos. En grafos, existen dos tipos de rutas: de avance y retroceso.
- *Ruta Dirigida:* La ruta dirigida es aquella en la cual no se repiten nodos en su transcurso, en esta ruta, no existen retrocesos del objeto.[9]

E. Algoritmo de Dijkstra

Mejor conocido como *Algoritmo de caminos mínimos*, este algoritmo consiste en calcular los caminos mínimos desde un nodo origen hacia un nodo de destino con el camino de menor peso en un grafo. Dicho algoritmo, calcula el costo o peso de los nodos adyacentes y luego reutiliza esa información para calcular el peso de los demás nodos[10].

Su funcionamiento es de la siguiente manera:

- Empezamos con el grafo dirigido cerrado con N nodos y sea *x* el nodo de origen, D un vector de tamaño *N* que contendrá las distancias desde el nodo de origen hacia el resto de nodos.
- Una vez definido los parámetros anteriores, todas las distancias en D comienzan con valor infinito ya que se desconoce en un principio, a excepción con la distancia hacia el mismo nodo, el cual es cero.
- Se recorren todos los nodos adyacentes *w* y se calcula la distancia entre dicho nodo hacia los adyacentes. Si la distancia obtenida del nodo actual es menor a la del vector distancia entonces dicho vector se actualiza con la nueva distancia y así obteniéndose la ruta mínima con respecto a un peso(distancia) establecido.
- Una vez definida la ruta mínima entre dichos nodos. Nuestro nodo de origen se actualizará al nodo recién encontrado y se repite dicho proceso. [7]

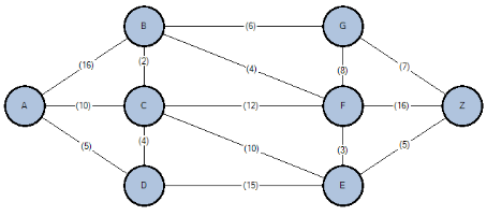

Fig. 2. Ejemplo de grafo con costes.

En la “Fig. 2”, se pretende obtener la ruta mínima entre el nodo de origen A y el nodo de destino Z. A continuación, en la “Fig. 3”, se presenta dicha respuesta mediante la aplicación del algoritmo de Dijkstra.

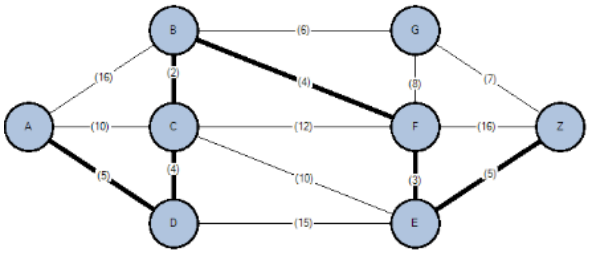

Fig. 3. Solución del ejemplo de grafo

F. MATLAB

Es una herramienta de software matemático basado en matrices, que ofrece un entorno de desarrollo integrado con un lenguaje propio de programación. Sus graficas integradas facilitan la visualización de datos y obtención de información a partir de ellos. Una vasta biblioteca de herramientas (Toolboxes) integradas permiten implementar y simular algoritmos de toda índole[11]. Entre las prestaciones de MatLab tenemos las siguientes:

a) Grafos en MatLab: Matlab nos permite definir grafos dirigidos mediante el comando digraph(A), donde A es una matriz de adyacencia cuadrada.

Otro método para implementar un grafo dirigido es mediante sus arcos ingresando el comando *diagraph(s,t)*. En dicho caso, es necesario definir dos vectores: *s* y *t*, con tamaño igual a m (número de arcos en el grafo), de manera que *s(i)* y *t(i)* indican los vértices de inicio y fin del i-esimo arco, respectivamente. Adicional a esto, se puede añadir coste o peso a los arcos mediante un vector peso de manera que en la posición *i* asigna el peso del arco *s(i)* → *t(i)*. Por consecuente, el comando queda de la siguiente manera: *graph(s,t,peso)*[12].

En la tabla 1 se detallan las principales propiedades del comando graph.

TABLA I. PROPIEDADES DEL COMANDO GRAPH

Propiedad	Descripción
digraph(s,t,weights,nodenames)	Adicional, especifica nombres de los nodos utilizando un arreglo de vectores tipo carácter.
digraph(s,t,weights,NodeTable)	Especifica otros atributos de nodo, utilizando una tabla.
digraph(s,t,weights,num)	Especifica el número de nodos en el gráfico.
digraph(s,t,EdgeTable,___)	Utiliza una tabla para especificar atributos de borde en lugar de especificar peso.

b) Representacion Gráfica: Un grafo puede ser representado o visualizado mediante el comando plot, indicando el grafo correspondiente, por ejemplo: *plot(G).*

Se puede añadir varias propiedades para una mejor visualización del grafo mediante las propiedades declaradas en la Tabla 2[12].

TABLA II. PRINCIPALES PROPIEDADES DEL COMANDO PLOT.

Propiedad	Descripción
layout	Dibuja los vértices y arcos de manera diferente. Entre los modificadores tenemos: circle, force, layered, subspace.
labeledge	Añade nombre a cada uno de los arcos.
highlight	Remarca nodos o arcos de un grafo, para dibujarlos en distinto color o tamaño.

c) Shortestpath

El comando *shortestpath* especifica la ruta minima entre dos nodos individuales. Dicho comando requiere un grafo dirigido o no, y dos vectores, el vértice de inicio y el vértice de fin[13]. Por ejemplo: *CMV = shortestpath(G,s,t).*

Este comando, posee algunas opciones adicionales. Por defecto, si no se asigna ninguna especificación, ocurrirá lo siguiente:

- Si los pesos son positivos, utilizará el algoritmo de Dijkstra.
- Si los pesos son negativos, utilizará el algoritmo de Bellman-Ford.

G. Comunicación RS-485

Actualmente considerado la manera más eficiente de comunicación serie, debido a su alto requerimiento en la automatización industrial, utilizado por las redes industriales como: Modbus, Profibus DP, ARCNET, BitBus, WorldFip, Lon, Interbus y en redes no estándar[14]. El estándar RS-485 o EIA-485 es un estándar de capa física del modelo OSI, su conexión se realiza por medio de un cable de 2 o 3 hilos, donde un hilo es de datos, un hilo es de datos invertidos y a menudo un hilo neutro, proporcionando una alta robustez y resistencia a las interferencias. El estándar RS-485 trabaja a velocidades medianas de 35Mbit/s hasta 10 metros y 100 kbit/s en 1200 metros, además de poder integrar 32 nodos es decir 1 maestro y 31 esclavos[15].

III. DISEÑO

Para la validación del algoritmo se construyó un prototipo de banda transportadora, que posee el acondicionamiento de la parte mecánica, electrónica y de comunicación como se muestra en la “Fig. 4”. El diseño de cada sistema se detalla a continuación.

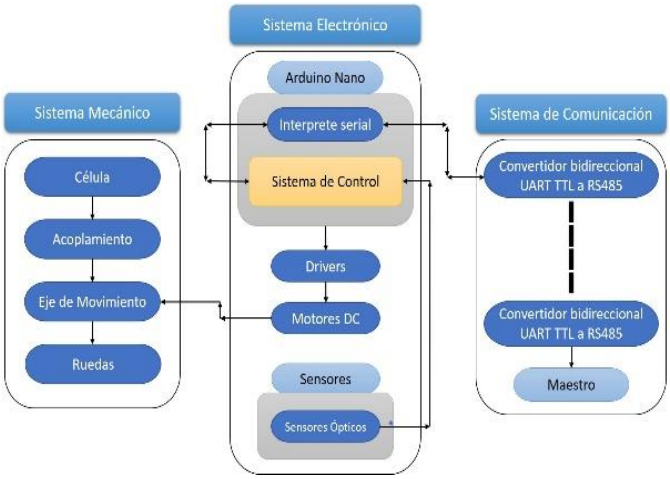

Fig. 4. Sistema general del prototipo de banda omnidireccional.

Para el diseño mecánico, se empezó con el modelamiento de cada parte de la banda transportadora, por medio de las

dimensiones del acople de los motores y las llantas omnidireccionales, se obtuvo celdas de forma hexagonal regular de 80 mm por cada lado. A continuación, se implementó motores TGP01D-A130 los cuales tienen una caja reductora de plástico con una relación 1:48, de dimensiones de 69.2 mm por 22.5 mm por 18.8 mm, con torque máximo de 2kg/cm y con más de 300 rpm a 12 voltios, estos a su vez, acoplados a llantas omnidireccionales dobles de 60 mm de diámetro, elaboradas en una impresora 3D.

La banda Transportadora consta de 17 células como se observa en la "Fig. 5", de las cuales, 13 células poseen 3 motores, 2 células poseen 2 motores y 2 células poseen un motor. Las células que poseen 3 motores son las encargadas de transportar los objetos, por medio de la interconexión con los movimientos de las demás células formando así una trayectoria, las células que poseen 2 motores tienen la finalidad de direccionar al objeto hacia la salida y por último las células que poseen un motor facilitan el traslado en forma horizontal.

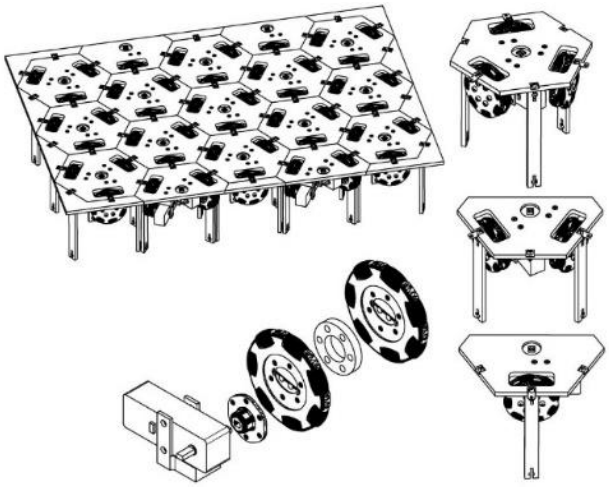

Fig. 5. Estructura general del prototipo de banda.

Para la ejemplificación del algoritmo en la banda y para tener un control de posicionamiento se integró sensores ópticos infrarrojos CNY70, Arduino Nano, el cual es el encargado de la adquisición de datos y el control de cada uno de los actuadores por medio del circuito integrado L293D.

Para interactuar con cada célula se incorporó un convertidor bidireccional UART TTL a RS485, medio por el cual se conecta a la red RS485, donde se emite y recibe la información necesaria para el accionar y detener de los actuadores, así como, conocer el estado lógico en el que se encuentra el sensor de cada célula, otorgando información de la posición del objeto transportado dentro de la banda, todos los elementos se encuentran integrados dentro de una placa PCB como consta en la "Fig. 6". Toda la información se centró dentro de un maestro, el cual está conformado por un Arduino Mega conectado de forma serial al software Matlab.

Fig. 6. Diseño 3D de la placa PCB (Esclavo).

Para la aplicación de las diferentes rutas se simuló las células mediante la herramienta de software *MatLab*, obteniéndose así un grafo del total de la banda. Por último, empleando diferentes herramientas dentro del software se verificó y analizó las rutas dispuestas.

A. *Caracterización de movimientos*

Cada célula se encuentra compuesta por 3 ruedas con disposición angular central de 120° entre ellas, a partir de una apotema del hexágono regular y paralelas a los lados de este. Al controlar el movimiento de cada una de estas llantas es posible generar los siguientes comportamientos detallados en la Tabla 3.

TABLA III. DIRECCIÓN DE MOVIMIENTOS DE CADA CÉLULA.

Dirección de desplazamiento	Rueda 1		Rueda 2		Rueda 3	
	Dirección	Estado	Dirección	Estado	Dirección	Estado
↑	↗	True	↖	True	—	False
↓	↙	True	↘	True	—	False
↖	—	False	↖	True	←	True
↘	—	False	↘	True	→	True
↗	↗	True	—	False	→	True
↙	↙	True	—	False	←	True

De manera gráfica se visualiza los movimientos de las ruedas 1, 2 y 3, de la siguiente manera:

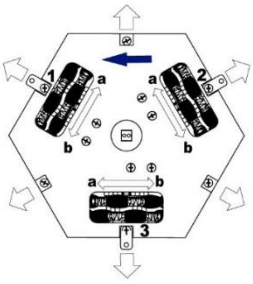

Fig. 7. Esquema de movimientos de cada célula.

Como se observa en la "Fig. 7", es posible generar 6 movimientos, de los cuales, el movimiento 1, 2 y 3 no son permitidos en la banda, ya que genera retrocesos en el transporte de los objetos. Por lo tanto, los movimientos disponibles son 4, 5 y 6. Para realizar dichos movimientos es

necesario el accionamiento combinado de dos ruedas y la tercera rueda sobrante es utilizada como apoyo.

Para la identificación de cada célula (nodo) dentro de la banda se las denomino de con la letra mayúscula E y un numero identificativo del nodo, como se muestra en la “Fig. 8”.

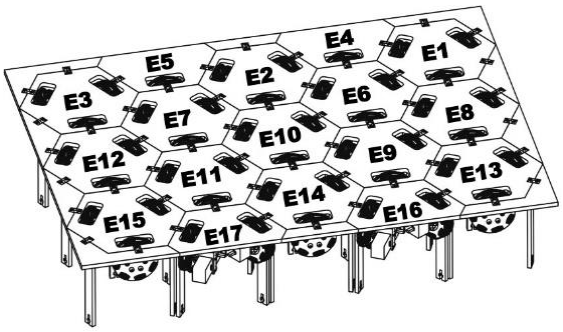

Fig. 8. Denominación de nodos.

Además, en la “Fig. 8”, se observa que existen células con dos y una rueda. Las células E4 y E5 (células con una rueda) se acoplaron como células de apoyo para la incorporación de un movimiento adicional en las células E1, E2 y E3; las cuales permiten desplazamientos entre: E1 hacia E2, E3 hacia E2. Las células E16 y E17 (células con 2 ruedas) corresponden a las células finales de salida de los objetos, por lo tanto, solo poseen un movimiento. Las células restantes poseen desplazamientos previamente predispuestos en la Tabla II.

B. Generación de posibles Rutas

Para el desarrollo del algoritmo de control de rutas se definió la aplicación de path planning, permitiendo la representación de un grafo dirigido de la banda el cual consta de 17 de nodos, uno por cada célula. Recíprocamente se añadió los diferentes arcos entre los nodos, estos arcos indican los movimientos que posee cada célula dando así una representación general de la banda.

Otro parámetro importante en la banda omnidireccional es el tráfico de transporte, el cual, dependerá directamente de la cantidad de objetos trasladados en la banda, el área disponible y el tamaño de los objetos. Por consiguiente, para las dimensiones del prototipo de banda ejemplificado, se estableció 3 células de entrada (E1, E2 y E3) y 2 células salidas (E16 y E17), como se puede observar en la “Fig. 8”.

Mediante la herramienta de software Matlab es posible generar el grafo dirigido de la banda, como se muestra en la “Fig. 9”.

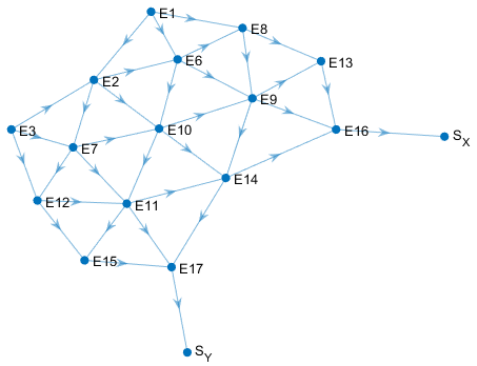

Fig. 9. Grafo generado mediante MatLab.

Con base a las diferentes combinaciones de entradas y salidas se desarrolló una tabla de verdad (Tabla 4), en la cual, se detalla la manera en que las entradas interactúan con las salidas determinando los posibles casos en los que se trasladan los objetos.

TABLA IV. TABLA DE VERDAD DE POSIBLES CASOS.

ENTRADAS			SALIDAS		RESULTADO
E1	E2	E3	E16	E17	
0	0	0	0	0	NO EXISTE
0	0	0	0	1	NO EXISTE
0	0	0	1	0	NO EXISTE
0	0	0	1	1	NO EXISTE
0	0	1	0	0	NO EXISTE
0	0	1	0	1	CASO 1
0	0	1	1	0	CASO 1
0	0	1	1	1	NO EXISTE
0	1	0	0	0	NO EXISTE
0	1	0	0	1	CASO 1
0	1	0	1	0	CASO 1
0	1	0	1	1	NO EXISTE
0	1	1	0	0	NO EXISTE
0	1	1	0	1	CASO 3
0	1	1	1	0	CASO 3
0	1	1	1	1	CASO 2
1	0	0	0	0	NO EXISTE
1	0	0	0	1	CASO 1
1	0	0	1	0	CASO 1
1	0	0	1	1	NO EXISTE
1	0	1	0	0	NO EXISTE
1	0	1	0	1	CASO 3
1	0	1	1	0	CASO 3
1	0	1	1	1	CASO 2
1	1	0	0	0	NO EXISTE
1	1	0	0	1	CASO 3
1	1	0	1	0	CASO 3
1	1	0	1	1	CASO 2
1	1	1	0	0	NO EXISTE
1	1	1	0	1	CASO 4
1	1	1	1	0	CASO 4
1	1	1	1	1	CASO 5

Para las entradas, el valor binario 1 define la existencia de un objeto en la entrada. Por ende, el valor binario 0 indica la ausencia de objetos en su respectiva entrada. Para las salidas, el valor binario 1 describe la salida de destino previamente seleccionada, por lo que, el valor binario 0 indica la no selección de dicha salida.

Mediante la aplicación del algoritmo de planificación de rutas descrito por Dijkstra, es posible generar la ruta dirigida entre el nodo de entrada y el nodo de salida, esto basado en la ruta mínima mediante la aplicación de un peso. En este caso se asignó la distancia como el principal parámetro de peso, esto debido a la simetría regular de la banda que posee entre células y el movimiento a una velocidad constante de las ruedas.

Debido a la forma hexagonal regular de cada una de las células que conforman la banda, la distancia desde el centro de cada una de ellas hacia sus células adyacentes es exactamente la misma en cualquier dirección. Por consiguiente, el peso entre células asignado es el mismo.

Mediante la herramienta de software MatLab y una vez descrito los parámetros necesarios para la aplicación del algoritmo de Dijkstra, se generó las rutas en cada uno de los casos declarados en la Tabla 4. El script de evaluación de rutas declara tanto el origen y destino de los diferentes objetos, los vectores de nodos (vértices) de origen como los nodos de destino y un vector adicional de pesos. A partir de esto, el algoritmo evaluó las posibles rutas y definió dependiendo el caso, la ruta con el menor peso o coste, tal como lo muestra la "Fig. 10".

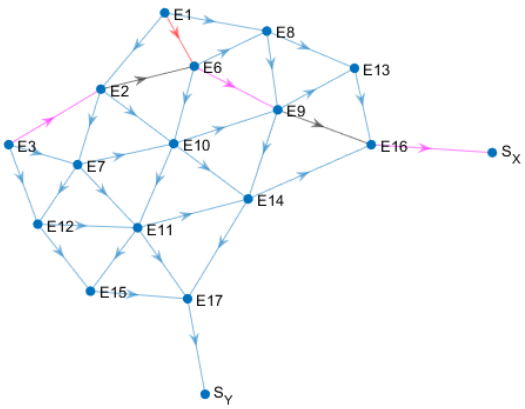

Fig. 10. Ejemplo de rutas obtenidas mediante el algoritmo.

C. *Prevención de colisiones*

Una vez generado las posibles rutas, los grafos obtenidos en los casos donde ingresan dos o tres objetos presentaron casos de colisión. Además, es importante recalcar el consumo energético que puede llegar a tener la banda al transportar dos o tres objetos, debido a que las rutas proporcionadas carecen de restricción alguna al accionar múltiples células. Esto conllevo al desarrollo del algoritmo que nos proporciona datos necesarios para las modificaciones a las rutas pertinentes.

Como punto de inicio se estableció prioridades, es decir, la ruta que realice la menor cantidad de pasos entre su célula de origen hacia su destino será la ruta que menos modificaciones adquiera.

Asimismo, con la utilización de células compartidas para rutas que tienen el mismo lugar de destino se limitara la cantidad de células accionadas y por tanto el consumo energético de la banda disminuirá, como se detalla posteriormente en la sección de resultados.

Basados en la descomposición de celdas se identificó grafos de adyacencia que permitieron los cambios necesarios en el coste de adyacencia de las rutas. Añadido al movimiento que realizan las células E4 y E5, facilitaron la generación de nuevas rutas prácticas y libres de colisiones y a su vez redujeron la cantidad de esclavos activados simultáneamente.

D. *Configuración del prototipo de banda*

Mediante una red de comunicación tipo estrella y acoplado el estándar de comunicación RS-485, como se indica en la "Fig. 11", permitió el intercambio de información entre cada célula (esclavo) y un maestro interprete. Esta red posee una trama compuesta de un carácter identificador de inicio de trama, la dirección del esclavo a la cual se desea acceder, una identificación de función, datos de movimiento y un carácter de fin de trama.

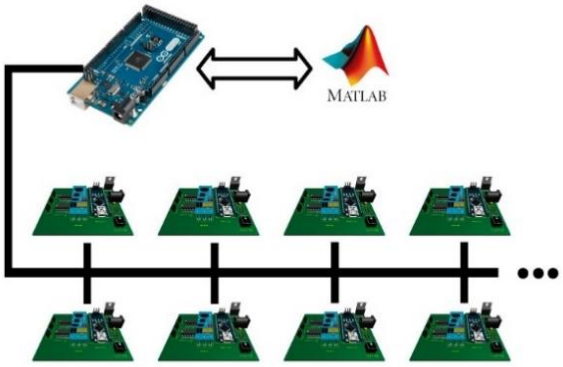

Fig. 11. Comunicación Maestro – Esclavos.

La programación de cada esclavo se encuentra descrito por el diagrama de flujo de la "Fig. 12".

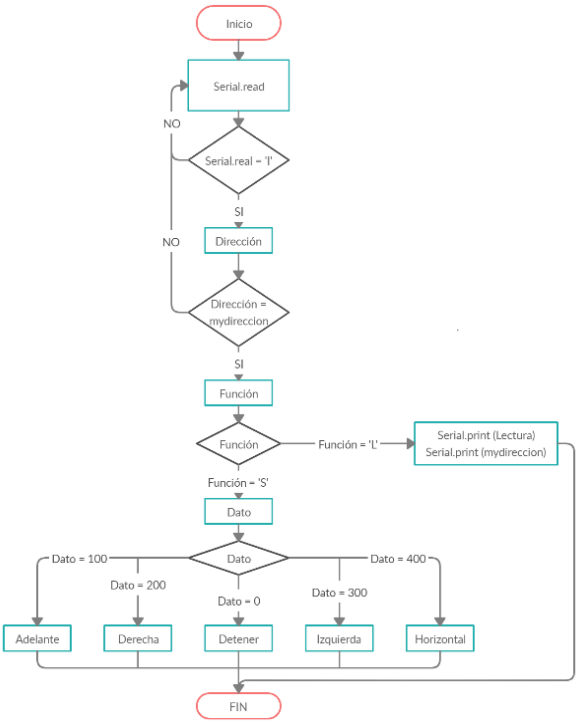

Fig. 12. Diagrama de flujo del esclavo implementado.

E. *Implementacion del algortimo en la banda*

Para la incorporación del algoritmo dentro del prototipo de banda se determinó una tabla de rutas, las cuales se encuentran añadidas en la programación del Arduino maestro y el procesamiento de cada ruta fue establecido por el software Matlab dependiendo los parámetros ingresados por medio de un GUIDE, el cual registra cada uno de estos parámetros y por medio de la ejecución del algoritmo indica las rutas a seguir por el objeto dentro del prototipo.

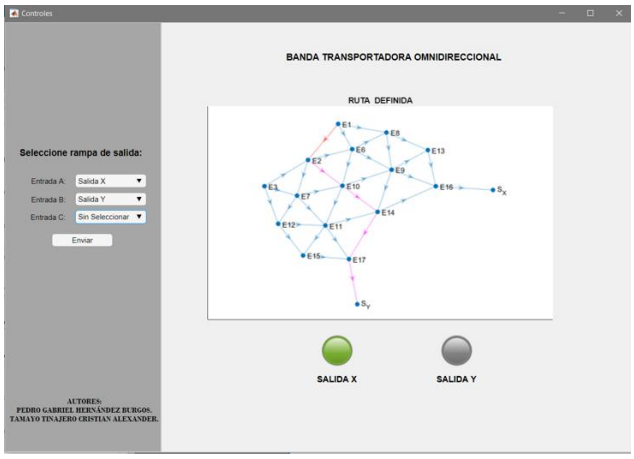

Fig. 13. GUIDE para el usuario.

Como se observa en la "Fig. 13", el GUIDE proporciona al usuario la elección de salida de cada una de las entradas, teniendo como opciones: Salida X, Salida Y y Sin Seleccionar. Un botón denominado *Enviar*, el cual procesa los datos de rampa de salida previamente ingresados y generara un grafico de la ruta a realizar por los objetos que puede ser visualizado en el área definida *Ruta Definida*. Por último, se añadieron dos indicadores que se activarán si un objeto llega a la rampa de salida respectiva.

Mediante la conexión serial con el Arduino maestro, es posible identificar qué tipo de caso recae, por tanto, el maestro asocia el caso dispuesto en su tabla de rutas, previamente incorporada y direcciona cada uno de los esclavos su respectivo movimiento, llegando a cumplir la combinación correcta movimientos para la ejemplificación de la ruta descrita previamente por el algoritmo. Como se observa en la "Fig. 14", se observa los diferentes casos añadidos a la tabla de rutas en el Arduino Maestro.

```
354⊟  void E15_I(){ //************ Envio de Datos al esclavo 1 ************ //
362
363    //*********************CASOS DE 1 ENTRADA 1 SALIDA*********************//
364⊟  void A_X(){
400⊟  void B_X(){
434⊟  void C_X(){
470⊟  void A_Y(){
508⊟  void B_Y(){
544⊟  void C_Y(){
580    //*********************CASOS DE 2 ENTRADA 2 SALIDA SIN CRUCE*********************//
581⊟  void AB_XY(){
627⊟  void AC_XY(){
673⊟  void BC_XY(){
719    //*********************CASOS DE 2 ENTRADA 2 SALIDA CON CRUCE*********************//
720⊟  void AB_YX(){
763⊟  void BC_YX(){
808⊟  void AC_YX(){
852     //*********************CASOS DE 2 ENTRADA 1 SALIDA*********************//
853⊟  void AB_XX(){
892⊟  void BC_XX(){
934⊟  void AC_XX(){
978⊟  void AB_YY(){
1020⊟ void BC_YY(){
1062⊟ void AC_YY(){
1106   //*********************CASOS DE 3 ENTRADA 2 SALIDA*********************//
1107⊟ void ABC_XYY(){
1156⊟ void ABC_XXY(){
1205⊟ void ABC_XYX(){
1251⊟ void ABC_YXY(){
1297⊟void Detener() {
```

Fig. 14. Programación de rutas.

IV. Resultados

Las diferentes pruebas de acoplamiento del algoritmo fueron realizadas con objetos con dimensiones de 150 [mm] por 10 [mm] de lado de arista. Como se aprecia en la Tabla 4, existen 5 casos, los cuales fueron categorizados de acuerdo por el número de objetos que ingresaron en el prototipo de banda y su respectiva salida. Para efectos demostrativos las células de ingreso: E1, E2 y E3 fueron asignadas a las entradas: A, B y C, y las células de destino E16 Y E17 describen las salidas X, Y. Además, en las gráficas siguientes se añadieron flechas las cuales representan la dirección y pasos de el/los objeto/s transportados.

A. *Traslado de un objeto hacia una salida.*

En dicha predisposición, se ha detecto un único objeto en la entrada, en dicho caso, el algoritmo de control direccionara los esclavos del prototipo de banda de la siguiente manera:

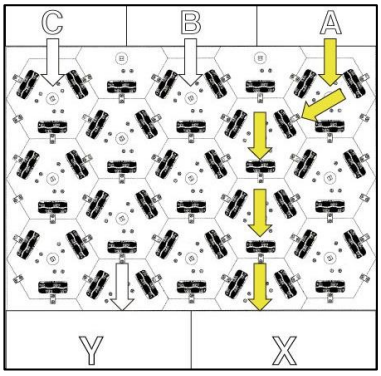

Fig. 15. Ejemplo de ruta desde A hacia X.

En la "Fig. 15", se observa un ejemplo de los 6 posibles, donde detalla una ruta dirigida desde E1 hacia E16 pasando por la secuencia de células E6, E9. En base a lo experimentado se observa que en estos casos los objetos fueron trasladados de manera exitosa por la secuencia previamente descrita.

B. *Traslado de dos objetos hacia dos salidas.*

- *Ruta sin cruce hacia las salidas*
  Como se muestra en la "Fig. 16", la ruta dirigida desde A hacia X, se observa el accionamiento de las células E1, E6, E9, E16. En cuanto, la ruta dirigida desde B hacia Y se activó las células E2, E7, E11, E17. El algoritmo describe rutas independientes con trayectorias definidas por el caso descrito en la parte A, esto debido a que no poseen rutas de cruce.

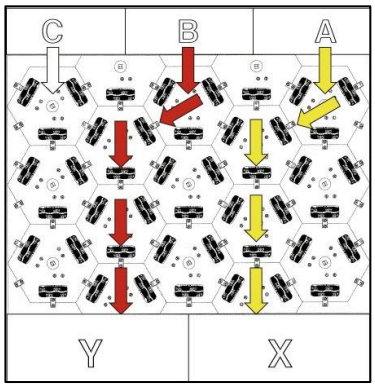

Fig. 16. Ejemplo de ruta desde A y B hacia X y Y, respectivamente.

- *Rutas con cruce hacia la salida*
  Basado en la premisa anterior, si el algoritmo describiera rutas independientes sin tomar en cuenta las posibles colisiones, como se muestra en la "Fig. 17"; causaría llegada de los objetos a una misma célula, en la misma cantidad de pasos (colisión).

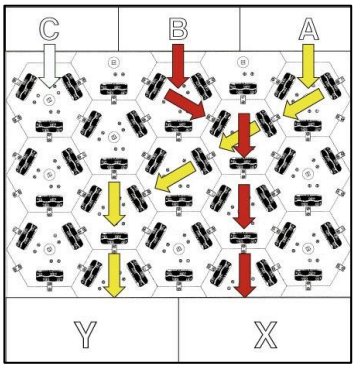

Fig. 17. Ejemplo de ruta con cruce sin algoritmo de colisiones.

En este caso y casos similares, es indispensable el actuar del algoritmo, ya que determina un nuevo coste o peso lo que conlleva a una nueva ruta libre de colisiones como se aprecia en la "Fig. 18".

Para satisfacer el cambio de dirección de la célula en común(E2), se utilizó la información del estado lógico de los sensores dispuestos en el prototipo.

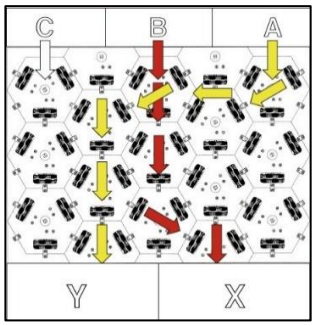

Fig. 18. Ruta con cruces con algoritmo de colisiones.

- *Rutas hacia una misma salida*

Para ejemplo de este caso, se ingresó dos objetos en las entradas A, B dirigidas hacia la salida Y, como se muestra en la "Fig. 19". Se observó que sin el actuar del algoritmo generó rutas no compartidas al tener una misma rampa de destino.

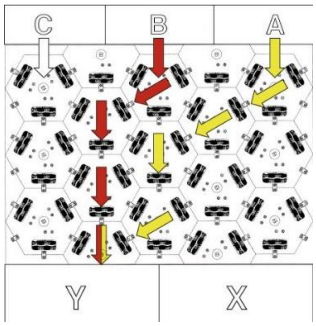

Fig. 19. Ruta con cruces sin algoritmo de colisiones.

Como se observa en la "Fig. 20", implementado el algoritmo de colisiones detalló rutas con células compartidas y con la misma dirección en las células E2, E10, E14 y E17. A comparación de la "Fig. 19", se destaca la utilización de células compartidas, reduciendo así el número de células activadas y por tanto el consumo energético. Debido a que en la "Fig. 20" se encuentran activas 6 células y en la "Fig. 19", las células activas son 8. Esto representa un 25% de ahorro energético dentro de esta ruta.

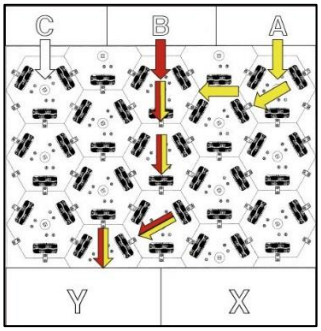

Fig. 20. Ejemplo de ruta con cruces con algoritmo de colisiones.

C. *Traslado de tres objetos hacia dos salidas.*

Este caso, el algoritmo combino las condiciones especificadas en los casos anteriores y generó rutas como se muestra en la "Fig. 21". Estas rutas al ser una combinación de casos anteriores tuvieron células compartidas y con la misma dirección (E9 y E16), correspondientes a las rutas dirigidas que comparten un mismo destino. Además, se aprecia una célula en común con cambio de dirección (E2).

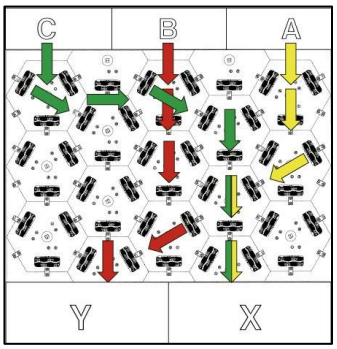

Fig. 21. Ejemplo de ruta de tres objetos en el prototipo de banda.

Para mayor detalle de los casos y resultados obtenidos, se los puede apreciar accediendo a la siguiente dirección: https://youtu.be/QJ53vhoeuMM. En el video se ejemplifica y se verifica el funcionamiento cada uno de los casos descritos en la TABLA XX en el prototipo de banda omnidireccional celular.

V. CONCLUSIONES

Con base a lo experimento, se apreció el correcto comportamiento del algoritmo dentro del prototipo de banda en donde se destaca la compatibilidad con los controladores implementados. Además, se determinó que existe una disminución de consumo energético en rutas que comparten una misma rampa de salida.

La estabilidad y confiabilidad de la red es un requisito indispensable en el traslado de los objetos debido a que un

fallo en ella ocasionaría percances, debido a la velocidad en la cual se transportan estos.

Con la adición de células aledañas (siempre y cuando se encuentren disponibles) a la ruta prevista por el algoritmo, se cumple de manera satisfactoria cada una de las rutas, evitando el descarrilamiento de los objetos fuera de estas.

Como futuro trabajo se destaca la implementación del algoritmo desarrollado en este artículo, en otro tipo de bandas semejantes; siempre y cuando cumplan con condiciones similares de hardware del prototipo presentado. Por ejemplo: adicionando mayor número de células, adecuación de distintos sistemas de sensorización, diferente protocolo de comunicación, etc.

VI. Referencias

[1] M. Jaramillo, "Diseño e implementación de un sistema de clasificación para supervisar un proceso industrial en el Laboratorio de Hidrónica y Neutrónica.," pp. 1–7, 2015, [Online]. Available: http://repositorio.espe.edu.ec/handle/21000/10049.

[2] IRP-INTRALOGÍSTICA, "Qué son las bandas transportadoras," 2018. https://irp-intralogistica.com/que-son-las-bandas-transportadoras/ (accessed Dec. 11, 2019).

[3] A. Córdova and W. Quimbiamba, "Diseño, control e implementación de una celda robotizada para clasificación de productos mediante el trabajo colaborativo de dos robots SCARA.," Universidad de las Fuerzas Armadas ESPE, 2018.

[4] A. Gasparetto, P. Boscariol, A. Lanzuti, and R. Vidoni, "Path Planning and Trajectory Planning Algorithms: a General Overview," vol. 29, no. September 2017, pp. 3–27, 2015, doi: 10.1007/978-3-319-14705-5.

[5] A. Cruz Martín, "Planificación de Trayectorias en Sistemas Multirobot," Universidad de Málaga, 2004.

[6] F. Prieto, "Métodos de Generación de Trayectorias," Unoversidad de Sevilla, 2017.

[7] I. Cabezas, "Aplicación de Path Planning de un Robot Móvil basado en Raspberry Pi," universidad de Sevilla, 2017.

[8] J. A. Escalera, F. J. M. D. Abu-Dakka, P. J. Alhama, and M. Abderrahim, "Obtención del modelo dinámico simbólico de robots ramificados utilizando grupos de Lie y grafos.," *Actas las XXXVII Jornadas Automática*, pp. 755–761, 2016, [Online]. Available: http://hdl.handle.net/10016/26287.

[9] I. Montes, "Tema 5 : Introducción a la Teoría de grafos.," 2015. http://ocw.uniovi.es/ (accessed Nov. 20, 2019).

[10] E. Aramayo, "Búsqueda de Caminos en Aplicaciones Robóticas," Universidad Politécnica de Cataluña, 2017.

[11] MathWorks, "Descripción del producto MATLAB: Guía de Usuario (R2019b)," 2019. .

[12] MathWorks, "Digraph: User´s Guide (R2018a)," 2018. https://la.mathworks.com/help/matlab/ref/digraph.html (accessed Nov. 12, 2019).

[13] MathWorks, "Shortestpath: User´s Guide (R2015b)," 2015. https://la.mathworks.com/help/matlab/ref/graph.shortestpath.html (accessed Nov. 13, 2019).

[14] O. Weis, "¿Qué es RS485? Guía de comunicación RS485," *Dec 11,2019*, 2019. https://www.eltima.com/es/article/rs485-communication-guide/.

[15] N. G. Forero Saboya, "Normas de Comunicación en Serie :," *Rev. Ingenio Libr.*, no. 2, pp. 86–94, 2012, [Online]. Available: http://www.unilibre.edu.co/revistaingeniolibre/revista-11/art13.pdf.