Seguidor de pared con sistema operativo en tiempo real

Wall tracker with real-time operating system

DOI: 10.46932/sfjdv5n5-007

Received on: Apr 02nd, 2024
Accepted on: Apr 22nd, 2024

Moisés García Villanueva
Maestria en Ingeniería Eléctrica
Institución: Universidad Michoacana de San Nicolás de Hidalgo, Facultad de Ingeniería Eléctrica
Dirección: Av. Francisco J. Mújica S/N, C.P. 58030, Morelia Michoacán
Correo electrónico: moises.garcia@umich.mx

Salvador Ramírez Zavala
Maestria en Ingeniería Eléctrica
Institución: Universidad Michoacana de San Nicolás de Hidalgo, Facultad de Ingeniería Eléctrica
Dirección: Av. Francisco J. Mújica S/N, C.P. 58030, Morelia Michoacán
Correo electrónico: salvador.ramirez@umich.mx

RESUMEN
Los módulos de IoT (por sus siglas en Inglés de Internet of Things) actualmente se encuentran presentes en la robótica de servicio, casas inteligentes, sistemas de control remoto o solución de adquisición de datos. De las consideraciones principales para la construcción de estos módulos se encuentra el utilizar microcontroladores de bajo consumo de energía, la posibilidad del procesamiento en paralelo, en donde varias soluciones de sistemas operativos de tiempo real (RTOS por sus siglas en Inglés de Real-Time Operatig System) se encuentran disponibles en el mercado. En la técnica de seguir pared para un sistema robótico, se tiene la necesidad de adquirir los datos de los sensores de proximidad en tiempo real y también un mecanismo de comunicación para la visualización de datos adquiridos en el sistema sensorial del robot. En este trabajo se presenta el uso de RTOS para las tareas que se han planteado como necesidades en el sistema robótico que sigue una pared.

Palabras clave: IoT, RTOS, Robot Seguidor de Pared, ESP32.

ABSTRACT
IoT modules (Internet of Things) are currently present in service robotics, smart homes, remote control systems or data acquisition solutions. Of the main considerations for the construction of these modules is the use of low-power microcontrollers, the possibility of parallel processing, where various real-time operating system solutions (RTOS for its acronym in English of Real-Time Operatig System) are available on the market. In the wall following technique for a robotic system, there is a need to acquire data from proximity sensors in real time and also a communication mechanism for the visualization of data acquired in the sensory system of the robot. This work presents the use of RTOS for the tasks that have been raised as needs in the robotic system that follows a wall.

Keywords: IoT, RTOS, Wall Follower Robot, ESP32.

1 INTRODUCCIÓN

Actualmente se están utilizando ampliamente los módulos de hardware en sistemas de adquisición de datos, en casas inteligentes, en la robótica de servicio o en sistemas de control. Una parte significativa de estos módulos están construidos en soluciones que emplean microcontroladores diseñados específicamente para estas tareas (Dudak *et al*., 2022).

El problema de procesamiento de datos obtenidos por dispositivos electrónicos sensoriales en soluciones embebidas se refiere a un problema de aplicación. Por lo tanto, es solo el método de lectura de los datos requeridos o el diseño de una estructura del programa con el que esta lectura se simplifica a las acciones leer o escribir el valor. Sin embargo, el problema es más complejo si la cantidad de tareas a completar se incrementa, es decir, cuando además de leer datos de un conjunto de sensores y almacenarlos localmente, se deben controlar actuadores (motores), se necesita que el sistema mantenga comunicación a través de una red local (WiFi o ethernet) y finalmente proporcionar un servicio Web. El hecho de que existen diferentes tareas que requieren ser completadas más o menos asíncronos entre sí, hace que sea muy fácil argumentar que la complejidad adicional que pueda existir se podrá resolver y valdrá la pena empleando un RTOS (Murikipudi *et al*., 2015).

Los robots móviles son sistemas que tienen la capacidad de desempeñar tareas específicas en su ambiente sin la intervención humana, es decir, deforma autónoma. Un robot inteligente tiene un gran sistema de percepción que le permite decidir la mejor acción y esto no depende precisamente del diseño del hardware, sino de un elemento más importante “el cerebro” del robot, lo que se traduce en el sistema operativo implementado para controlar el robot (Mirikipudi *et al*., 2012).

En el caso de estudio de este trabajo, comprende la implementación de un robot seguidor de pared y en el que se crean las necesidades descritas previamente, el escenario general esta descrito de la siguiente manera: a través de un servidor de páginas Web es necesario observar la información adquirida por el conjunto de sensores con que cuenta el robot, además de almacenar una cantidad de está información para su análisis en un futuro.

2 IMPLEMENTACIÓN DE RTOS EN MICROCONTROLADORES

Cada generación de microcontroladores tiene como objetivo ampliar sus capacidades en el tiempo, podemos ver ahora la necesidad de conectividad al internet y el manejo de pantallas táctiles, lo que hace imprescindible el manejo de un Sistema Operativo en Tiempo Real (RTOS) en las nuevas aplicaciones (Dudak *et al*., 2022, Willian Stallings, 1997). Algunas de las ventajas sobre el enfoque clásico (programación en un sólo ciclo) son:

a) RTOS es determinista y garantiza las respuestas a eventos dentro de un intervalo de tiempo definido.the text preceding the first marker ends with a colon;
b) diseño multitarea preventivo adecuado a los cambios en cada versión. Este diseño asegura que la respuesta de cada uno de los eventos sea independiente uno de otro;
c) controladores de comunicación integrados. Frecuentemente los RTOS incluyen comunicación para diferentes periféricos, tales como USB, TCP/IP y recientemente WIFI y BT;
d) herramientas adecuadas para depurar, que permiten detectar uso inadecuado y sobre flujo en memoria, respuesta tardía a eventos o uso muy grande de recursos del CPU;
e) uso eficiente de los recursos del CPU.

Mistry (*et al*., 2014) Harvey (1993, p. 112) describe,

> FreeRTOS es un RTOS de código abierto diseñado originalmente para arquitecturas de un solo núcleo. La modificación para arquitecturas multinúcleo se realiza a nivel comunitario, por lo que la implementación no está completa. De cualquier forma, la modificación hecha por la comunidad voluntaria ya es funcional; El concepto de multiprocesamiento simétrico (SMP por sus siglas en ingles de Symmetric Multiprocessing) es apoyado por el RTOS. Debido a la apertura, muchos científicos también están intentando crear una adaptación

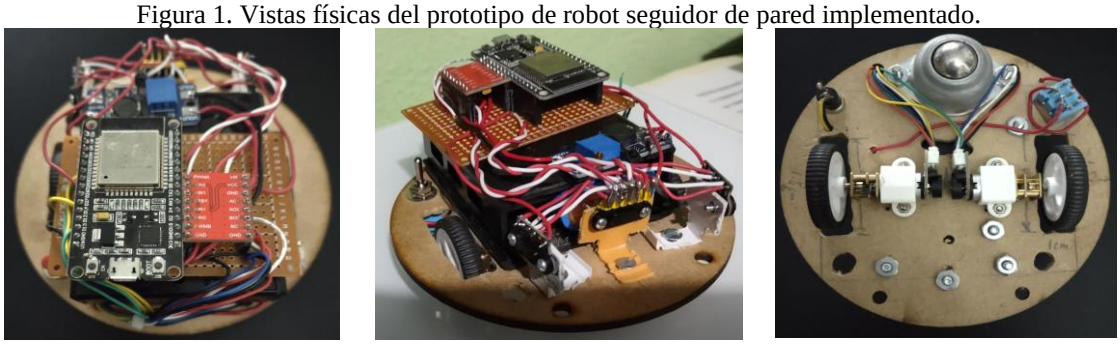
En el presente trabajo se hace uso del microcontrolador ESP32, el sistema operativo en tiempo real sobre este dispositivo es el FreeRTOS. Es de código abierto, diseñado para sistemas embebidos y proporciona funciones básicas para aplicaciones de alto nivel. Las funciones principales permiten la administración de memoria, de las tareas y la sincronización de la API (Kolban, 2017). Una evaluación del desempeño de FreeRTOS en el microcontrolador ESP32 multinúcleo se encuentra en el trabajo de Arm (*et al*., 2022)

2.1 IMPLEMENTACIÓN DEL ROBOT SEGUIDOR DE PARED

La estrategía de seguir una pared, permite resolver otras diversas actividades, que van de lo simple a algo más complejo. Cuando se utiliza un robot para resolver alguna de estas actividades y que involucra el hecho de seguir una pared, el robot tiene que identificar primero la pared y entonces seguirla, convirtiendose en una tarea critica o de suma importancia para el robot, si desea desempeñarse de una manera eficiente en resolver la actividad más compleja que se le ha encomendado. Numerosas técnicas se han utilizado para detectar una pared y seguirla (Teng, *et al*., 2020). Diferentes tipos de sensores se han utilizado en los robots para asegurarse que se encuentran en la ruta planeada (Sun *et al*., 2019), esto le permite al robot tomar decisiones en tiempo real y es considerada una de las formas más sencillas para lograr la evasión de obstáculos. Sin embargo, es reconocido que los sensores tienen un campo de visión limitada, considerando la ruta que el sistema global debe seguir (Bullen; Rajan, 2009). En esta sección se

describen los principales componentes y diagramas de conexión del prototipo robot seguidor de pared que se implemento, el cual se muestra en la Figura 1.

Figura 1. Vistas físicas del prototipo de robot seguidor de pared implementado.



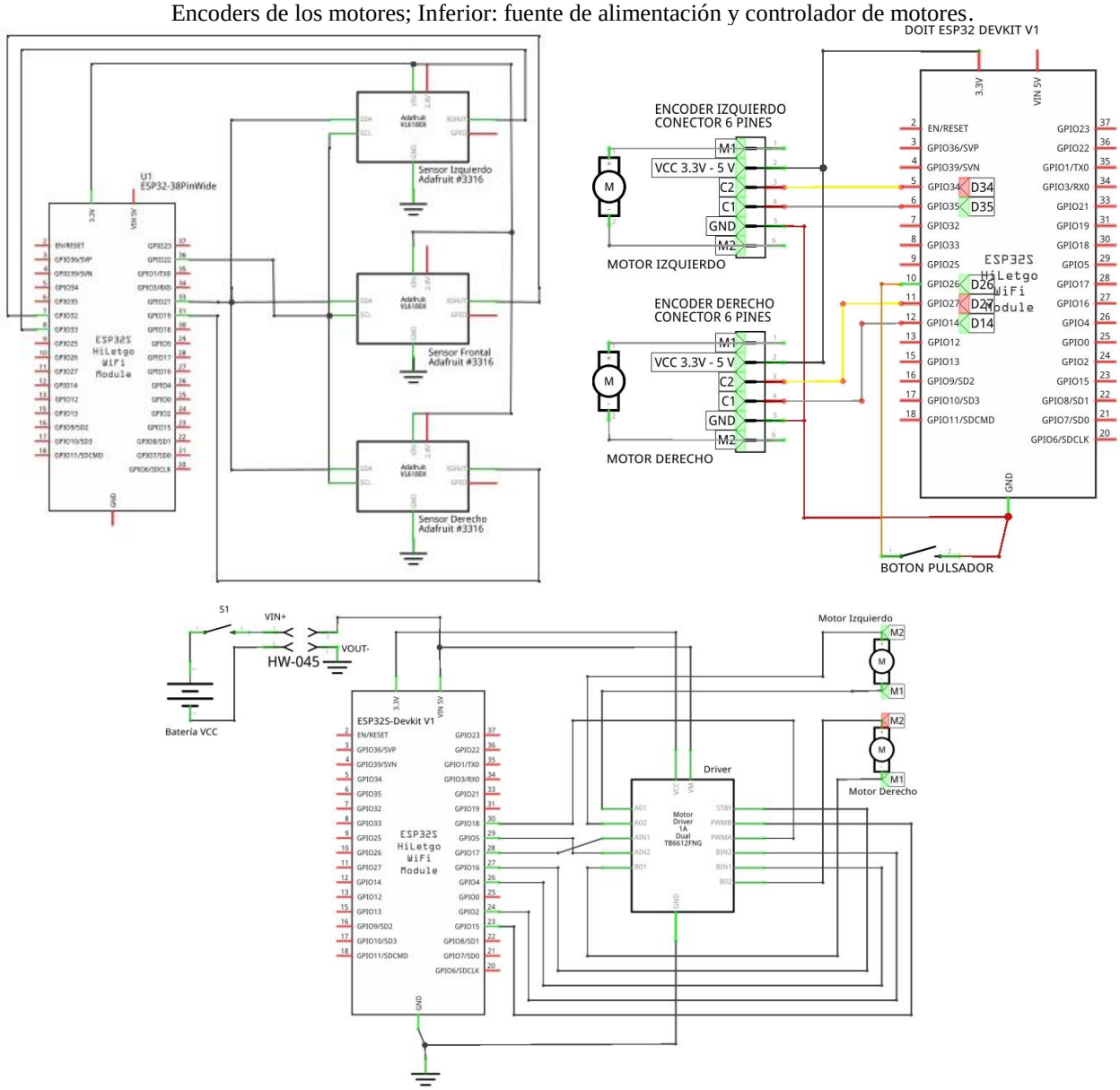
Fuente: Elaboración propia.

De acuerdo a la Figura 1 se puede observar que el robot implementado contiene los siguientes elementos:

- microcontrolador: ESP32-WROOM-32, contiene dos microprocesadores Xtensa® 32-bit LX6, 520 KB en RAM, comunicación WiFi y Bluetooth;
- sensores de proximidad: TOF050C, es un sensor de rango láser, utiliza el chip VL6180 en su medición de distancia que es máxima de 50 cm, una zona ciega de 0 a 2 cm y cuenta con la comunicación es I2C;
- controlador de motores: TB6612FNG;
- batería de alimentación: 2 baterías en paralelo 18650 a 3.7 V, 2600 mAh;
- regulador de voltaje de subida CD-CD: Mt3608 con voltaje mínimo de entrada de 2-24V y un voltaje máximo de salida de 28V a 2A.;
- motores. Micromotores N20 con encoder acoplado, 530 rpm a 6V.

Los diagramas de conexión de los diferentes componentes principales del robot seguidor de pared se muestran en la Figura 2. Los tres sensores de proximidad utilizan el mismo canal de comunicación I2C en el ESP32.

Figura 2. Diagramas de conexión del robot seguidor de pared. Superior izquierdo: sensores de proximidad; Superior derecha: Encoders de los motores; Inferior: fuente de alimentación y controlador de motores.



Fuente: Elaboración propia.

2.2 IMPLEMENTACIÓN DE MULTIPROCESOS

Las dos tareas principales del proyecto que aquí se plantean son: a) Mediciones de proximidad por los 3 sensores TOF050C que además conlleva el cálculo del control PID para determinar las velocidades de desplazamiento del robot; y b) Servicio WiFi y Web para la adquisición y visualización de los datos de proximidad. El microcontrolador ESP32 es un multiprocesador con 2 núcleos, cada uno es identificado con un nombre: el CPU de protocolo (PRO_CPU o CPU 0) y el CPU de aplicación (APP_CPU o CPU 1). EL CPU 0 controla el WiFi, Bluetooth y otros periféricos internos, mientras que el CPU 1 está disponible para nuestro programa de usuario. FreeRTOS proporciona la función xTaskCreatePinnedToCore() para crear una tarea y pueda ejecutarse en alguno de los núcleos del

procesador, el séptimo argumento de la función nos indica el núcleo en donde se ejecutará la tarea creada. El fragmento de código mostrado en el recuadro 1 corresponde a la forma de invocar en la función setup() de nuestro programa las llamadas de la función xTaskCreatePinnedToCore() y debido a que dichas funciones debe existir un ciclo infinito para mantener la ejecución en de las instrucciones de cada tarea, la función loop() por lo tanto podrá estar sin código.

Recuadro 1.- Invocación de la función xTaskCreatePinnedToCore() en la función setup() para asignar la ejecución de tareas en cada núcleo del ESP32

```
void setup(){
                :
        xTaskCreatePinnedToCore(codigoTarea1, “ServicioWiFi”, 10000, NULL,
tskIDLE_PRIORITY,NULL, 0)
        xTaskCreatePinnedToCore(codigoTarea2, “MedicionesProximidad”, 10000, NULL, 1,NULL,
1)
                :
}
```

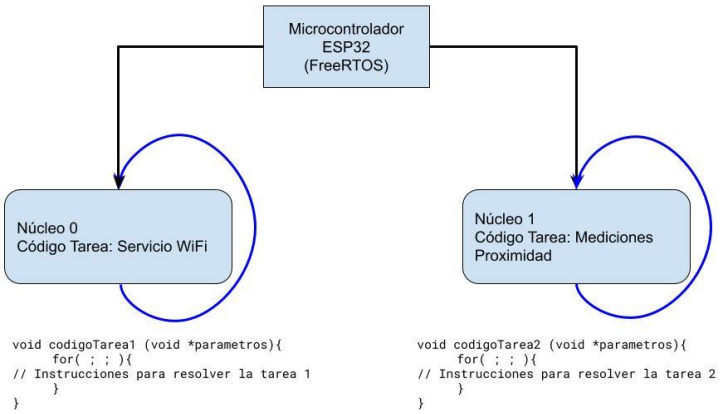
Fuente: Elaboración propia.

La Figura 3 nos muestra el diagrama de flujo general del microcontrolador empleando sus dos núcleos y a la asignación de una de las tareas descritas a cada uno de ellos. Se observa que dentro de cada función deberá existir un ciclo infinito con las instrucciones que permiten completar la tarea asignada.

2.3 SISTEMA DE ADQUISICIÓN DE DATOS

El ESP32 tiene el sistema de comunicación WIFI integrada, lo que le permite funcionar como un punto de acceso inalámbrico (Ver la Figura 4(a)) o como un dispositivo que se puede unir como cliente WIFI a una red (Véase la Figura 4(b)). Una vez conectado a la red, el ESP32 ofrece el servicio WEB y entonces es posible acceder a este mediante un dispositivo cliente a través de un navegador Web.

Figura 3. Diagrama de flujo en el microcontrolador ESP32 asignando una tarea a cada uno de sus núcleos, se indica además el código de la función y su estructura básica que incluye el ciclo infinito.



Fuente: Elaboración propia.

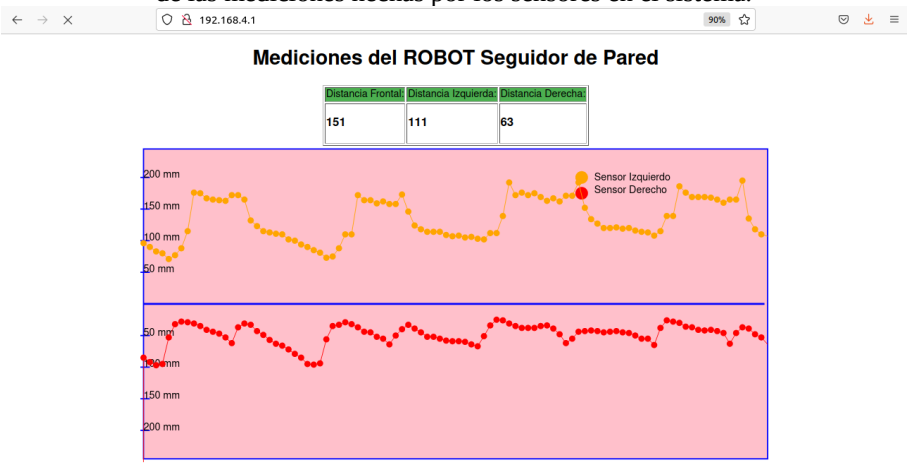
Figura 4. Arquitecturas de conexión inalámbrica del ESP32. (a) Punto de acceso; (b) Cliente en una red inalámbrica.

Fuente: Elaboración propia.

3 RESULTADOS

El servicio WEB permite el acceso a todos los elementos del robot, gracias a ello es posible capturar los datos que miden los sensores. Se debe considerar el procesamiento en ambos núcleos para que el servicio WEB y el procesamiento de control de desplazamiento en la tarea de seguir la pared no se vea interrumpido. La interfaz que se genera en el servicio WEB tiene los siguientes elementos: etiquetas para indicar las mediciones realizadas por los sensores en el robot; y gráfica que muestra las mediciones de los sensores izquierdo y derecho, indicando la distancia a la que se encuentran de la pared. La Figura 5 muestra la interfaz del servicio WEB al que es posible acceder a través de un navegador para visualizar la información. La gráfica de mediciones señala un desplazamiento recto cuando las mediciones arriba y abajo de la línea central así lo describen, hecho que nos indica que el comportamiento del robot es un desplazamiento que se mantiene a un valor de distancia de una de las paredes que va siguiendo (valor conocido como referencia a la pared). Por otro lado, en la parte central de la gráfica es posible observar cuando el robot se encuentra más cercano a una de las paredes, por un momento a la pared izquierda y por otro a la pared derecha.

Figura 5. Interfaz WEB para visualizar los datos almacenados en el robot durante su desplazamiento. Visualización gráfica de las mediciones hechas por los sensores en el sistema.



Fuente: Elaboración propia.

4 CONCLUSIONES

Es imprescindible el manejo de un RTOS en sistemas que requieren completar múltiples tareas en un sistema embebido. Se implementó una interfaz que permite la visualización de la información a través de un navegador Web en la red local que sirve el dispositivo ESP32 en el modo de punto de acceso. La actualización de la interfaz se implementó a través de instrucciones javascript que recargan la página en forma automática especificando el tiempo de recarga de los datos, se concluyó que un tiempo de 500 milisegundos permite una sensación adecuada de visualización de los datos. También se observó que el robot hace la tarea de seguir una pared de una manera muy fluida, sin las interrupciones que se generan cuando el código se ejecuta en forma secuencial sobre un sólo núcleo.

Un trabajo futuro para el sistema es el almacenamiento de datos en un servidor externo o dispositivo de memoria SD. Con los datos de proximidad se podrá calcular el error medio cuadrático de la trayectoria que sigue el robot, entonces cuantificar para diferentes valores de las constantes del control PID el mínimo, con ello se contará con una herramienta que permita evaluar el desempeño de los valores asignados al control, disminuyendo así el ensayo de prueba y error que se realiza para sintonizar el control.

REFERENCIAS

Arm, J., Baštán, O., Mihálik, O., & Bradáč, Z. (2022). Measuring the Performance of FreeRTOS on ESP32 Multi-Core. *IFAC-PapersOnLine*, *55*(4), 292-297.

Bullen IV, H. W., & Ranjan, P. (2009). *Chaotic Transitions in Wall Following Robots*. arXiv preprint arXiv:0908.3653.

Dudak, J., Gaspar, G., Sedivy, S., & Budjac, R. (2022). Utilization of RTOS Solutions in IoT Modules Based on RISC Microcontrollers. In *Computer Science On-line Conference* (pp. 80-93). Springer, Cham.

Kolban, N. (2017). *Kolban's Book on ESP32*. USA: Leanpub.

Mistry, J., Naylor, M., & Woodcock, J. (2014). Adapting FreeRTOS for multicores: An experience report. *Software: Practice and Experience*, *44*(9), 1129-1154.

Murikipudi, A., Prakash, V., & Vigneswaran, T. (2015). Performance analysis of real time operating system with general purpose operating system for mobile robotic system. *Indian Journal of Science and Technology*, *8*(19), 1-6.

Stallings, W. (1997). *Operating Systems* (2nd ed.). Prentice-Hall, ISBN 0-13-180977-6.

Sun, G., Li, X., Li, P., Yue, L., Yu, Z., Zhou, Y., & Liu, Y. H. (2019, November). Adaptive vision-based control for rope-climbing robot manipulator. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems* (IROS) (pp. 1454-1459). IEEE.

Teng, T. W., Veerajagadheswar, P., Ramalingam, B., Yin, J., Elara Mohan, R., & Gómez, B. F. (2020). Vision based wall following framework: A case study with HSR robot for cleaning application. *Sensors*, *20*(11), 3298.

Yeak, W. C., Zakaria, M. F., & Poad, H. M. (2012). *Real Time Operating System for Mobile Robot Using PICos 18*. Departement of Mechatronics and Robotics Engineering, Departement of Computer Engineering, Universiti Tun Hussein Onn Malaysia.