Design and Deploy a Prototype of Proximity Shield with IoT Controller

Jorge Landaverde, Daniel Flores
Instituto de Investigación e Innovación en Electrónica
Universidad Don Bosco
Soyapango, El Salvador
ljorge_13@hotmail.com, dann.flores1993@gmail.com

Carlos Bran
Instituto de Investigación e Innovación en Electrónica
Universidad Don Bosco
Soyapango, El Salvador
cbran@udb.edu.sv

***Abstract*— In this project we approach a prototype for a proximity shield using a low cost ultrasonic sensor to create a security sphere that allows autonomous systems like robots or UAVs avoid crashing..**

To deploy the solution its use an Internet of Thing controller, to facilitate the creation of a user Web-based interface, which also allows to adjust the proximity alarms.

Index Terms—Telemetry, Internet of thing, Serial Communication, Ultrasound sensor, Cloud computing, Fog computing.

I. Introducción

El escudo de proximidad, es un sistema que permite obtener datos telemétricos de objetos en el contorno del dispositivo que lo porta, siendo de mucha utilidad en la telemetría de dispositivos móviles como UAV's o robots exploradores.

Con el creciente incremento de las tecnologías de drones se ha vuelto una necesidad el portar dispositivos de este tipo, ya que en muchas situaciones es necesario tener un mecanismo que alerte de posibles colisiones o de errores del controlador humano, para así evitar pérdidas materiales por daño en los dispositivos. En el mercado existen pocas opciones que permiten la detección de objetos, y los que se encuentran tienen precios elevados que para implementaciones experimentales no pueden ser cubiertos.

Para la alternativa que se presenta, se utilizan sensores ultrasónicos, ya que despues de evaluar otras opciones como sensores inductivos[1] se observo que los ultrasonicos presentan mejor alcance, existen soluciones como el RADAR o el LIDAR [2] en donde la señal realiza un barrido por todo el contorno, para detectar objetos sin embargo, para este proyecto se planea hacer uso de múltiples sensores ubicados estratégicamente de tal manera de poder tener una vista de los 3 ejes, debido a que una desventaja de estos dispositivos es que sólo pueden leer distancias en una dirección, por otra parte, tampoco poseen una gran distancia de lectura o una precisión elevada, pero proveen una solución de bajo costo y a distancias considerables son capaces de obtener una buena estimación de superficies o posibles objetos cercanos.

El uso de módulos IoT, permite además de la facilidad de implementación y nuevas posibilidades para futuras expansiones del proyecto. Con la integración del Internet vienen ligados nuevos desafios, referentes a la conexión, procesamiento en la nube y la necesidad de un sub procesamiento en la niebla, de los cuales se hablará con más detalle en las secciones posteriores.

El trabajo se divide en 3 secciones en la primera se describe el modelo, diseño y proceso de implementación del prototipo, luego se presentan los resultados de pruebas realizadas, para finalizar con un conjunto de conclusiones y recomendaciones para la mejora de la propuesta.

II. II. Diseño y Despliegue de la Solución

El el propósito del sistema propuesto se basa en la obtención de las magnitudes de las distancias para cada eje donde se quiere controlar la proximidad de objetos, para luego procesarlas y enviarlas a través de un módulo que comunica los microcontroladores esclavos con el controlador maestro sobre el cual correrá el programa de aplicación y gestión de la información en un modelo de procesamiento en la niebla a fin de evitar posibles errores en caso que no haya conexión a internet, además de proporcionar una forma de visualización de los niveles de aproximación de los objetos a dicho sistema de forma local.

El modelo de controlador considera salidas adicionales que permitiran establecer alertas visuales de los niveles de aproximación que seran ademas replicados en una interfaz web para su visualización remota, donde ademas se incorporaran controles donde el usuario podra ajustar los margenes para las distintas alertas; el modelo general del diseño se presenta en la figura 1.

Las alertas por lo tanto se podran verificar local y remotamente y serviran como mecanismo de evaluación de la solución ademas de su tiempo de respuesta.

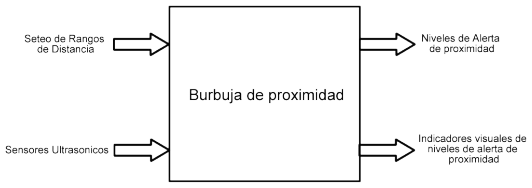


Fig. 1. Modelo de concepto del escudo de proximidad

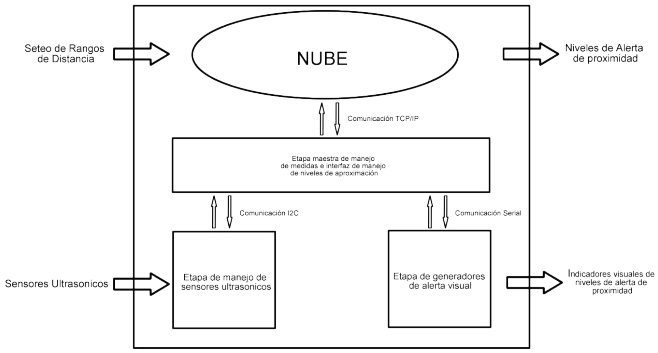


Fig. 2. Modelo funcional del escudo de proximidad

Los componentes principales en las que se estructura el prototipo se muestran en el diagrama funcional de la figura 2.

Las distancias son medidas por medio de unos microcontroladores configurados como esclavos que procesan el funcionamiento de los sensores ultrasónicos, dichos microcontroladores hacen las medidas de la distancia de forma periódica y prepara el dato para luego ser enviada al controlador maestro cuando este se lo solicita a través del protocolo de comunicación I2C, una vez el controlador maestro ha recibido la trama de datos que contiene las distancias medidas por los esclavos, este se encarga de gestionar esa información a través de los niveles que se hayan ajustado por defecto o que el usuario haya seleccionado a través de la interfaz web, para luego dar como resultados las diferentes alertas de aproximación que depende de las cotas y sus medidas usando un controlador de histeris.

Estas alertas son codificadas de forma serial para luego ser enviadas a otro microcontrolador que se encarga de emitir de forma visual esas alertas que ha recibido en el puerto serial. Los niveles de alertas que este controlador maestro genera también son enviados a Internet para ser visualizado a través de la interfaz web para que el operario ser percate de forma esquemática de los espacios disponibles con los que cuenta el dispositivo móvil antes de colisionar con algún obstáculo.

A. Elección y Descripción de los componentes

Tras detallar el modelo funcional del sistema se enumeran los diferentes componentes necesarios para el funcionamiento completo del prototipo además de describir cada uno de ellos.

- Modulo ultrasónico de alcance HC-SR04 [3]: Provee una función de medida sin contacto de 2cm a 400cm, la precisión de alcance puede alcanzar los 3mm. Este módulo incluye un transmisor y un receptor ultrasónico y un circuito de control. Este módulo es el encargado de medir las distancias en cada eje, para conocer el espacio vacío entre la unidad móvil y algún objeto u obstáculo.
- Microcontroladores ATMEGA328PU [4]: es un circuito integrado de alto rendimiento que está basado en una arquitectura RISC. El rol de este circuito integrado es el del manejar la obtención de las medidas de la distancia para los módulos ultrasónicos de alcance HC-SR04 y establecer la comunicación por I2C de este microcontrolador esclavo al controlador maestro.
- Particle SPARK Core [5]: Es una plataforma de producción a prototipo para desarrollo de productos de IoT. En esta parte se encuentra la esencia principal del Internet de las Cosas y es el controlador maestro, este módulo es el encargado de recibir los datos provenientes del microcontrolador esclavo y procesar los datos de forma independiente de la conexión a internet, dando como resultados los diferentes niveles de alerta para cada eje siendo este mismo lo que se envía hacia los microcontroladores para visualizar los datos de forma física haciendo posible el funcionamiento por aparte en la niebla. Este controlador maestro también envía los datos a la nube para que se muestren en la interfaz web.

B. Desarrollo de los componentes

La codificación de cada componente funcional se detalla a continuación:

- Para poder comunicar los microcontroladores esclavos que manejan el funcionamiento de los módulos ultrasónicos de alcance con el controlador maestros, fue necesario generar un código que permita establecer un empaquetado de los datos sobre un protocolo de comunicación I2C entre ellos. En el código del microcontrolador esclavo se estableció el manejo de seis módulos ultrasónicos de alcance, este código hace que el microcontrolador active cada módulo de forma secuencial, tome el registro de las medidas y se empaqueten en doce tramas de dos bytes, para luego ser enviadas por la línea SDA hacia el controlador maestro cuando este mismo los solicite, para el envío de la información a través de la línea SDA se implementó en el código un contador que lleva la cuenta de las veces en que solicita cada trama de dos bytes correspondiente a cada uno de los ejes en los que esta cada módulo ultrasónico de alcance. Así se puede conocer que dato le pertenece a cada eje.
- La codificación del controlador maestro consta de las siguientes sub-etapas:
    - Recepción y desempaquetados de los datos: en esta parte de la codificación el controlador maestro solicita la información al microcontrolador esclavo y en orden de acuerdo a cada eje guarda cada trama de dos bytes en variables individuales para ser procesadas más adelante.
    - Procesamiento de los datos: una vez los datos son guardados en las variables, se compara cada una ellas con los niveles establecidos de forma predeterminada o que el usuario a establecido por medio de la interfaz web, en base al resultado de cada comparación se

genera los datos de los niveles de alerta establecido para luego estos resultados ser enviados a la interfaz web via las variables publicadas a la nube y también codificado de forma serial para ser transmitido a otro microcontrolador por medio de la interfaz serial USART que ambos microcontroladores poseen, y que es conveniente ya que cumple los requerimientos de velocidad necesarios para el envío de estos datos. El microcontrolador que recibe los datos se encarga de generar las visualizaciones físicas de dichas alertas correspondientes a cada eje del sistema, por medio de LED que indican según el color el nivel de la alerta.

- La codificación de los microcontroladores esclavos que están a cargo de las alertas visuales en la niebla solo constatara de un bloque principal, este debe de encargarse de decodificar el dato recibido, ya que se enviara número de indicación de sensor y la alerta que ese eje posee en determinado momento, al recibir esos dos datos el bloque principal es capaz de deducir las alertas que debe imprimir en los LED para que sean visibles de forma física.

C. Conexiones físicas de los componentes

Las conexiones de los módulos ultrasónicos de alcance junto con el microcontrolador esclavo se puede apreciar en la figura 3.

Cada módulo cuenta con un pin de acción de la toma de la medida y otro pin que muestra el resultado en función del tiempo de la señal en estado alto, el ATMEGA se encarga de accionar cada módulo por los respectivos pines de Trigger y obtener los resultados para cada pin de OUT, estas medidas se van almacenando en una sola variable de 12 Bytes para luego ser enviada por el pin de SDA, bajo la comunicación I2C, del ATMEGA hacia el CORE.

El módulo CORE consta de dos pines dedicados a la comunicación I2C (SDA y SCL), estos permiten recibir los datos provenientes del ATMEGA. La figura 4 muestra los pines físicos del módulo CORE; además, tiene dos pines orientados a la comunicación serial USART para los microcontroladores encargados de los visualizadores de los niveles de alerta de aproximación.

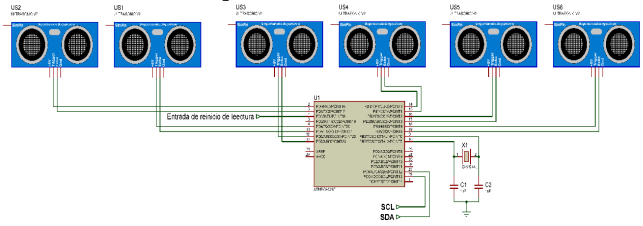


Fig. 3. Diagrama de conexión para los módulos ultrasónicos de alcance con el microcontrolador ATMEGA 328 PU

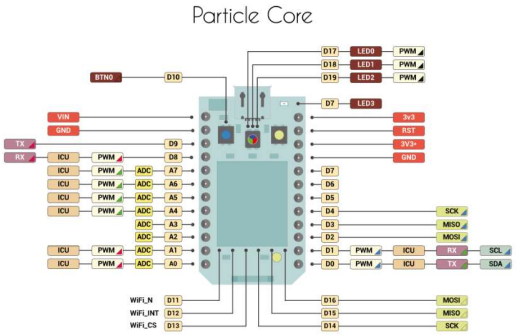


Fig. 4. Distribución de los pines del módulo IoT Spark CORE

D. El Internet de las Cosas y la computación en la niebla como solución.

El Internet de las Cosas (IoT por sus siglas en Ingles) representa una extraordinaria transformación en la manera en que nuestro mundo va interactuar en un futuro muy cercano, así como la creación del World Wide Web conectó los ordenadores a la red, y la siguiente evolución conectó a la gente al Internet, el Internet de las Cosas luce estar preparado para la interconexión de dispositivos, personas, entornos objetos virtuales y maquinas, de una forma que solo se creía parte de la ciencia ficción y nunca se había imaginado que sería posible.

En resumidas palabras, el Internet de las cosas representa la convergencia de la conexión de personas, objetos, datos y procesos, con lo que se está transformando vidas, la forma como se hacen negocio; esto nos permite tener control de todas las cosas por medio del procesamiento en la nube, el usuario es capaz de interpretar los datos y aportarle procesos para así tomar decisiones con información enriquecida; sin embargo no en todo momento se tiene conectividad a la nube, por lo cual es crucial brindarle a los diseños de los sistema un cierto grado de autonomía y así no quedar neutralizados cuando se pierda la conectividad, la capacidad de procesamiento local de los dispositivos IoT se conoce como computación en la Niebla, este concepto permite que los sistemas sean todavía capaces de tomar decisiones por si solos y procesar datos mientras se restablece la conexión a la nube.

El termino computación en la Niebla también conocido *Computation Edge* [6] significa que, en lugar de alojar y trabajar desde una Nube centralizada, los sistemas en la Niebla funcionan en los extremos de las redes, esta concentración significa que los datos se pueden procesar localmente en dispositivos inteligentes, en lugar de enviarse a la Nube para su tratamiento. Es uno de los posibles enfoques para el diseño de sistemas con Internet de las Cosas (IoT).

El procesamiento en la Niebla, al igual que numerosas implementaciones informáticas, surgió en la necesidad de abordar dos situaciones: que sea capaz de actuar en tiempo real con los datos entrantes, y trabajar dentro de los límites de los anchos de banda disponibles. Algunos sensores generan muchos bytes de datos, demasiados datos para enviarlos a la Nube, por lo que si no hay suficiente ancho de banda esto puede implicar mayores costos en conectividad. La informática en la Niebla sitúa algunas de las transacciones y

recursos en el borde la nube; en lugar de establecer canales para el almacenamiento y utilización en la Nube, se reduce la necesidad de ancho de banda al no enviar cada bit de información a través de canales en la nube, y en cambio lo agrega en determinados puntos de acceso. Al utilizar esta estrategia distribuida, se puede reducir los costos y aumentar la eficiencia del sistema.

III. III. Pruebas de Prototipo

Una vez realizadas las conexiones de todos los módulos que integran el sistema se dispuso a realizar las pruebas preliminares con el fin de evaluar el funcionamiento y si este necesitaría algún tipo de ajuste.

Como prueba inicial se realizó la comprobación de los datos obtenidos por el esclavo a cargo de la lectura de los sensores, para lo que se modificó el maestro agregando una pantalla LCD que serviría como mecanismo de visualización. Dando como resultado una correcta lectura de las distancias provistas por el microcontrolador esclavo encargado de la lectura de los sensores.

Luego se dispuso a realizar una prueba del procesamiento de las distancias, verificando si las alertas se generaban para los valores de umbral establecidos, y posteriormente el envío de estos datos a los microcontroladores encargados de la visualización física en los LEDs. Se pudo comprobar que la comunicación se realizaba correctamente y con tiempos de respuesta muy inferiores a 1 segundo.

Como último paso se dispuso la realización de las pruebas del dispositivo con la nube, ya que se debía comprobar que las alertas podían ser visualizadas desde la página web por internet en cualquier lugar, luego de algunas pruebas se comprobó que las alertas si se podían visualizar por medio de la página web, presentándose inconvenientes únicamente al tener congestión en la red, lo cual era de esperarse ya que para las pruebas no se desarrollo una topología de red dedicada lo que es un criterio de diseño de soluciones IoT, sin embargo las alertas locales aun se podían efectuar debido a su independencia por el modelo de procesamiento en la Niebla.

IV. Conclusiones

Para muchas soluciones donde se demande el procesamiento de tareas criticas es muy importante el uso de microcontroladores extras como los usados para el manejo de los módulos ultrasónicos de alcances HC-SR04 y las visualizaciones locales de los niveles de alerta de aproximación, con lo cual se evita la excesiva carga de código en el controlador maestro, convirtiéndolo así en un sistema distribuido por el cual cada componente se enfoca de forma más eficiente en sus funciones para lograr el objetivo global.

La Computación en la Niebla juega un papel muy importante en la implementación de este sistema basándolo bajo una tecnología y modalidad de trabajo orientada hacia el Internet de la Cosas, dándole autonomía al sistema y permitiendo que se procesen los datos sin necesidad que haya acceso a la Nube, también reduciendo el uso ancho de banda con la que el sistema cuenta para alcanzar la nube ya que solamente se transmiten y se procesen los datos que sean pertinentes para dicha finalidad.

Con la implementación de soluciones IoT surge la necesidad de usar redes aislada o privada [7], para la conexión de dispositivos de este tipo, también la red debe poseer el ancho de banda suficiente para el tráfico que caracteriza las implementaciones IoT. Con la implementación planteada se puso observar dificultades en la conexión al tratarse de una red de uso compartido con un tráfico de internet bastante congestionado. Por lo que se recomienda que para implementaciones similares no se conecte a una red de uso compartido y en cuando sea posible, generar una red exclusiva que facilite la conexión con el internet.

Con pequeñas mejoras la solución propuesta puede aplicarse en distintas tareas de sistemas autonomos que se desplazan en el espacio, para evitar los choques o para alertar a operarios de potenciales choques.

REFERENCIAS

[1] Wikipedia, Sensores inductivos. Disponible en: https://es.wikipedia.org/wiki/Sensor_inductivo

[2] E-Safety, Los Sistemas Anticolisión. Disponible en: http://www.centro-zaragoza.com:8080/web/sala_prensa/revista_tecnica/hemeroteca/articulos/R41_A7.pdf

[3] ELEC Freaks, Modulo Ultrasónico de Alcance. Disponible en: http://www.micropik.com/PDF/HCSR04.pdf

[4] Wikipedia, Atemega328. Disponible en: https://es.wikipedia.org/wiki/Atmega328

[5] Particle, Spark Core. Disponible en: https://www.particle.io/

[6] Wikipedia, Computación en la Niebla. Disponible en: https://en.wikipedia.org/wiki/Edge_computing

[7] Wikipedia, Red privada. Disponible: https://es.wikipedia.org/wiki/Red_privada

[8] Guntram Scheible, Bernd Smailus, Martin Klaus, Kai Garrels, Lothar Heinemann, “System for a machine having a large number of proximity sensors, as well as a proximity sensor, and a primary winding for this purpose” U.S. Patent 6749119 B2, jun 15, 2004.

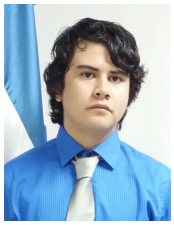
[9] Roy R. Storer, “Proximity operated machine control” U.S. Patent 4918560 A, April 17, 1990.

[10] Keith D. Wheeler, Stanley H. Edwards, Jr., “Two hand operated machine control station using capacitive proximity switches” U.S. Patent 5341036 A, August 23, 1994.

[11] J. Tlusty, G.C. Andrews , “A Critical Review of Sensors for Unmanned Machining,” CIRP Annals - Manufacturing Technology, Volume 32, Issue 2, 1983, Pages 563-572.

[12] Peter T. Bissert (National Institute for Occupational Safety and Health (NIOSH)) | Jacob L. Carr (National Institute for Occupational Safety and Health (NIOSH)) | Joseph P. DuCarme (National Institute for Occupational Safety and Health (NIOSH)) , “Proximity Detection Zones: Designs to Prevent Fatalities Around Continuous Mining Machines ” American Society of Safety Engineers, Volume 61, Issue 6, 2016.

[13] Dominique Guinard, Vlad Trifa, “Building the Web of Things: With examples in Node.js and Raspberry Pi,” Manning Publications; 1 edition, June 18, 2016, ISBN-13: 978-1617292682.



Jorge Landaverde. (M'14) Obtuvo su título de bachiller técnico industrial opción electrónica en el 2012, actualmente se encuentra estudiando quinto año de ing. Electrónica en la Universidad Don Bosco.

Daniel Flores. Graduado de bachillerato técnico vocacional en electrónica, graduado del técnico universitario en electrónica y actualmente cursa el quinto año de ingeniería en electrónica en la Universidad Don Bosco.

Carlos Bran: Director del Instituto de Investigación e Innovación en Electrónica, Graduado de Ingeniero Electricista, Postgrado en Gestión Tecnológica, Maestría en Investigación en Tecnologías de la información y estudios Doctorales en el Centro Singular de Investigación en Tecnologías de la Información CITIUS. Experiencia laboral incluye empresas como Dymel, Dynamo, AT&T, diversas compañías en el área de las telecomunicaciones, Decano de la facultad de ingeniería de la Universidad Don Bosco Director del programa Cisco Networking Academy, entre otras.
Profesor de sistemas embebidos, diseño electrónico, redes de computadoras y seguridad informática, los intereses de investigación incluyen áreas relacionadas con los sistemas embebidos, Internet de las Cosas, diseño de ASIC, protocolos de comunicación, sistemas paralelos, energy harvesting y visión por computadora.

V. ANEXOS

Código Esclavo de la lectura de los sensores ultrasónicos

```
//Vectores de recepción de datos de MCU esclavo
unsigned char X_f[2];
unsigned char X_r[2];
unsigned char Y_f[2];
unsigned char Y_r[2];
unsigned char Z_f[2];
unsigned char Z_r[2];
//Variables a usar en la nube
int Distancia1 = 0; // Variable global de la distancia del Slave X_f
int Distancia2 = 1; // Variabel global de la distancia del Slave X_r
int Distancia3 = 2; // Variabel global de la distancia del Slave Y_f
int Distancia4 = 3; // Variabel global de la distancia del Slave Y_r
int Distancia5 = 1; // Variabel global de la distancia del Slave Z_f
int Distancia6 = 2; // Variabel global de la distancia del Slave Z_r
//Variables de 16bits para las distancias
int Dist1=0; int Dist2=0; int Dist3=0; int Dist4=0; int Dist5=0;
int Dist6=0;
//Vector de valores definidos para los rangos de cada alerta
int values [18]  ;


void setup() {
  Wire.setSpeed(CLOCK_SPEED_100KHZ); // Se ajusta una comunicación a una velocidad de 100KHz
  Wire.begin(); // Se inicia el bus I2C
  Serial.begin(9600);
  Serial1.begin(9600);
  // Asocia las funciones con las variables globales con las funciones
  Particle.variable("Distancia1", &Distancia1, INT);
  Particle.variable("Distancia2", &Distancia2, INT);
  Particle.variable("Distancia3", &Distancia3, INT);
  Particle.variable("Distancia4", &Distancia4, INT);
  Particle.variable("Distancia5", &Distancia5, INT);
  Particle.variable("Distancia6", &Distancia6, INT);
  Particle.function("Seteo", Configuracion);
  pinMode(D2,OUTPUT);
  digitalWrite(D2,LOW); //Se inicia el pin de reinicio de lectura del esclavo en bajo
  // Cotas para X_f
  values[0] = 100;  values[1] = 200;  values[2] = 400;
  // Cotas para X_r
  values[3] = 100;  values[4] = 200;  values[5] = 400;
  //Cotas para Y_f
  values[6] = 100;  values[7] = 200;  values[8] = 400;
  //Cotas para Y_r
  values[9] = 100;  values[10] = 200;  values[11] = 400;
  // Cotas para Z_f
  values[12] = 100;  values[13] = 200;  values[14] = 400;
  // Cotas para Z_r
  values[15] = 100;  values[16] = 200;  values[17] = 400;
  }
void loop() {
    digitalWrite(D2,HIGH); //Enviar señal de reinicio de lectura de valores a el esclavo
  delay(10);
  digitalWrite(D2,LOW);
  // Recepción de las medidas por parte del chip ATMEL
  Wire.requestFrom(0x03, 2);
  while (Wire.available() > 0) {
    X_f[0] = Wire.read();  X_f[1] = Wire.read();
  }
  Wire.requestFrom(0x03, 2);
  while (Wire.available() > 0) {
    X_r[0] = Wire.read(); X_r[1] = Wire.read();
```

```
  }
  Wire.requestFrom(0x03, 2);
  while (Wire.available() > 0) {
   Y_f[0] = Wire.read(); Y_f[1] = Wire.read();
  }
  Wire.requestFrom(0x03, 2);
  while (Wire.available() > 0) {
   Y_r[0] = Wire.read();  Y_r[1] = Wire.read();
  }
  Wire.requestFrom(0x03, 2);
  while (Wire.available() > 0) {
   Z_f[0] = Wire.read(); Z_f[1] = Wire.read();
  }
  Wire.requestFrom(0x03, 2);
  while (Wire.available() > 0) {
   Z_r[0] = Wire.read(); Z_r[1] = Wire.read();
  }
   // Concatenació de niveles de alerta
  Dist1 = (X_f[1]<<8) + X_f[0];   Dist2 = (X_r[1]<<8) + X_r[0];
  Dist3 = (Y_f[1]<<8) + Y_f[0];  Dist4 = (Y_r[1]<<8) + Y_r[0];
  Dist5 = (Z_f[1]<<8) + Z_f[0];  Dist6 = (Z_r[1]<<8) + Z_r[0];
    //Procesamiento de distancias para obtener alertas
  if(Dist1 == 0 ){ Distancia1 = 0; }
  else if(Dist1 <= values[0] ){Distancia1 = 1;}
  else if(Dist1 > values[0] && Dist1 <= values[1] ){Distancia1 = 2;}
  else if(Dist1 > values[1] && Dist1 <= values[2]){Distancia1 = 3;}
  else { Distancia1 = 4;}

  if(Dist2 == 0 ){ Distancia2 = 0;}
  else if(Dist2 <= values[3]){Distancia2 = 1;}
  else if(Dist2 > values[3]  && Dist2 <= values[4]  ){Distancia2 = 2;}
else if(Dist2 > values[4]  && Dist2 <= values[5] ){Distancia2 = 3;}
else { Distancia2 = 4;}

   if(Dist3 == 0 ){ Distancia3 = 0;}
  else if(Dist3 <= values[6]){Distancia3 = 1; }
  else if(Dist3 > values[6]  && Dist3 <= values[7]  ){ Distancia3 = 2;}
  else if(Dist3 > values[7]  && Dist2 <= values[8] ){Distancia3 = 3;}
  else  {Distancia3 = 4;}

  if(Dist4 == 0 ){Distancia4 = 0;}
  else if(Dist4 <= values[9]){Distancia4 = 1;}
  else if(Dist4 > values[9]  && Dist4 <= values[10]  ){Distancia4 = 2;}
  else if(Dist4 > values[10]  && Dist4 <= values[11] ){Distancia4 = 3;}
  else {Distancia4 = 4;}

   if(Dist5 == 0 ){Distancia5 = 0;}
  else if(Dist5 <= values[12]){Distancia5 = 1;}
  else if(Dist5 > values[12]  && Dist5 <= values[13]  ){Distancia5 = 2;}
  else if(Dist5 > values[13]  && Dist5 <= values[14] ){Distancia5 = 3;}
  else  { Distancia5 = 4;}

  if(Dist6 == 0 ){Distancia6 = 0;}
  else if(Dist6 <= values[15]){Distancia6 = 1;}
  else if(Dist6 > values[15]  && Dist6 <= values[16]  ){Distancia6 = 2;}
  else if(Dist6 > values[16]  && Dist6 <= values[17] ){ Distancia6 = 3;}
  else {Distancia6 = 4;}

  //Envío de alertas hacia los esclavos para la visualización
  String cadena1 = String(Distancia1, DEC);  Serial1.println("1"+cadena1);
  String cadena2 = String(Distancia2, DEC);  Serial1.println("2"+cadena2);
  String cadena3 = String(Distancia3, DEC);  Serial1.println("3"+cadena3);
  String cadena4 = String(Distancia4, DEC);  Serial1.println("4"+cadena4);
  String cadena5 = String(Distancia5, DEC);  Serial1.println("5"+cadena5);
  String cadena6 = String(Distancia6, DEC);  Serial1.println("6"+cadena6);

  //Impresion de distancias reales en el monitor serial para depuración
  Serial.print(Dist1);Serial.print(",");Serial.print(Dist2);
  Serial.print(",");Serial.print(Dist3);Serial.print(",");Serial.print(Dist4);
  Serial.print(",");Serial.print(Dist5);Serial.print(",");Serial.println(Dist6);
    delay(100);
 }
 int Configuracion(String command)    // Recepción de las cotas
 {                                    // desde la pagina WEB
  int len = command.length();
  char buf[len + 1];
   command.toCharArray(buf, len + 1); //conversion de cadena String en vector
char
  char separator[] = ",";  // el caracter que se usa para separar los valores

  char *result = NULL;
  int index = 0;

  result = strtok( buf, separator );  // aqui se separan los caracteres
  while (  (index < 18) ) { // (result != NULL) &&
     Serial.print( result ); //Impresion de valores de las cotas para propcitos de
depuración
    Serial.println( );
    values[index++] = atoi(result);
    result = strtok( NULL, separator );
    } return 1; }
```