UNIVERSIDAD INTERNACIONAL SEK

Facultad de Arquitectura e Ingeniería

Ingeniería Mecánica Automotriz

Diseño de un prototipo para sistema de monitoreo del nivel de llenado en contenedores de basura por protocolo de comunicación inalámbrica IEEE 802.15.4 (Zigbee)

Cristian Yaser Ronquillo Muñoz

Nota del autor

Cristian Yaser Ronquillo Muñoz, Facultad de Arquitectura e Ingeniería, Universidad Internacional SEK.

Director Ing. Gustavo Adolfo Moreno Jiménez, M.Sc.

Cualquier correspondencia concerniente a este trabajo puede dirigirse a:

cyaserrm@gmail.com

Declaración Juramentada

Yo, Cristian Yaser Ronquillo Muñoz, con cédula de identidad 172282820-7, declaro bajo juramento que el trabajo aquí desarrollado es de mi autoría, que no ha sido previamente presentado para ningún grado a calificación profesional; y que se ha consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración, cedo mis derechos de propiedad intelectual correspondientes a este trabajo, a la UNIVERSIDAD INTERNACIONAL SEK, según lo establecido por la Ley de Propiedad Intelectual, por su reglamento y por la normativa institucional vigente.

Cristian Yaser Ronquillo Muñoz
C.C.: 1722828207

Agradecimientos

Quiero agradecer a:

Mis padres, por impulsarme cada día a ser mejor persona, permitirme crecer profesionalmente a lo largo de mi vida, y brindarme su sabiduría y consejo en cada decisión que he tomado.

Mis hermanos que me han inspirado a seguir adelante y me han brindado su apoyo incondicional en cada situación que he enfrentado.

La Institución SEK junto con sus docentes, quienes me han dotado de un gran conocimiento y se han mostrado dispuestos a brindarme su apoyo y consejo, tanto en el ámbito técnico como personal, a lo largo de mi carrera.

Dedicatoria

Dedico mi tesis a:

Mis padres y hermanos, quienes han sido mi ejemplo a seguir a lo largo de mi vida y me han abierto caminos a muchos logros que me convirtieron en la persona de hoy.

Mis amigos y docentes de mi carrera, personas con quienes he trabajado en equipo y me han enseñado una amplia gama de conocimientos que me permitieron cumplir con este trabajo. Su continuo apoyo me ha permitido desarrollar grandes trabajos y proyectos que resultaron en éste.

Índice de Contenido

Índice de Tablas y Figuras

Tablas

Figuras

Abreviaturas

- **UIT:** Unión Internacional de Telecomunicaciones
- **IEEE:** Institute of Electrical and Electronics Engineers
- **CIMI:** Cities in Motion Index (Índice de ciudades en movimiento)
- **IoT:** Internet on Things (Internet en las cosas)
- **WPAN:** Wireless Personal Area Network (Red de Área Personal)

Resumen

El siguiente trabajo presenta el diseño inicial de un sistema capaz de monitorear el nivel de llenado en contenedores de basura, mediante comunicación inalámbrica de protocolo Zigbee, dentro de una ciudad, generando un prototipo que demuestre su funcionalidad. Además, se pretende introducir a Quito una propuesta de proyecto para iniciar con su transformación hacia una ciudad inteligente en el futuro. Para lograr esto, se plantea una red compuesta de 1 coordinador y 2 antenas routers instaladas una en cada contenedor de basura. Estas antenas contienen sensores HC-sr04 que detectan el nivel de llenado desde la tapa del contenedor. Los sensores se conectan a un microcontrolador que interpreta su señal, y a través de un Xbee S2c la envía por radiofrecuencia al coordinador de la red. Al haber extraído los datos del coordinador, se los grafica empleando una interfaz de usuario, permitiendo así un monitoreo remoto del nivel de llenado de los contenedores dentro de la red.

Los resultados del trabajo comprueban la funcionalidad del sistema empleando una topología de red tipo Cluster Tree. El nivel de basura dentro de los contenedores testeados se obtuvo en forma de porcentaje en la interfaz, tomando en cuenta que ambos contenedores fueron de diferente tamaño.

Palabras clave: Zigbee, Ciudad Inteligente, Gestión de residuos, Diseño Iterativo, Xbee S2c, Modo API, Cluster Tree, XCTU, Arduino, Processing.

Abstract

The following work presents the initial design of a system capable of monitoring the fill level in garbage containers, using wireless Zigbee protocol communication, within a city, generating a prototype that demonstrates its functionality. In addition, it is intended to introduce a project proposal to Quito to start its transformation towards a smart city in the future. To achieve this, a network consisting of 1 coordinator and 2 antennas routers is proposed, one installed in each garbage container. These antennas contain HC-sr04 sensors that detect the fill level from the container lid. The sensors are connected to a microcontroller that interprets its signal, and through an Xbee S2c sends it by radio frequency to the network coordinator. Having extracted the data from the coordinator, it is plotted using a user interface, thus allowing remote monitoring of the fill level of the containers within the network.

The results of the work verify the functionality of the system using a network topology like Cluster Tree. The level of garbage within the containers tested was obtained as a percentage in the interface, taking into account that both containers were of different sizes.

Key words: Zigbee, Smart City, Waste Management, Iterative Design, Xbee S2c, API Mode, Cluster Tree, XCTU, Arduino, Processing.

Introducción

Antecedentes

Ciudades Inteligentes

En la actualidad, más de la mitad de la población mundial vive en zonas urbanas. Un estudio de la UIT (Unión Internacional de Telecomunicaciones) señala que para 2050, el porcentaje de pobladores en ciudades incrementará al 70%. Esta rápida urbanización puede magnificar varios problemas, puesto que actualmente las ciudades ya representan el 70% de las emisiones mundiales y entre el 60% y 80% del consumo energético. (UIT, 2020) Es debido a esta preocupación que los gobiernos buscan apoyo en tecnologías de la información y comunicación para construir ciudades cada vez más inteligentes y volverlas más sostenibles. Una ciudad inteligente se caracteriza por ser accesible, inclusiva, amigable, eficiente, integradora, asequible, segura, participativa y sostenible.

Hasta la fecha, ya existen varios ejemplares de ciudades inteligentes en el mundo que continuamente van evolucionando y son evaluadas por la Universidad IESE bajo el índice CIMI (Cities in Motion Index). Entre las ciudades dentro del top 10 en este ranking están Nueva York, Londres y Tokio. (Travel + Leisure, 2018) Desde el año 2013 se ha realizado esta evaluación a diferentes ciudades del mundo, lanzando rankings bajo diferentes estándares, de los cuales, los más importantes se encuentran en la figura 1.

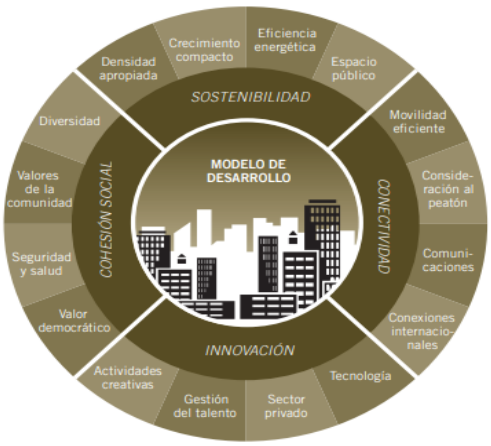


Figura 1. Principales estándares para calificar ciudades inteligentes bajo el índice CIMI (Berrone & Ricart, 2018)

Una de las principales características de las ciudades inteligentes es la aplicación de redes de comunicación que conecten a todo, es decir, que cada cosa en la ciudad tenga la capacidad de conectarse con las demás dentro de la red de la ciudad, abriendo paso al concepto del Internet de las Cosas (IoT). Como se puede observar en la figura 2, también se puede aplicar otros conceptos de comunicación inalámbrica para armar redes entre diferentes puntos de las ciudades, tales como Bluetooth o Zigbee.

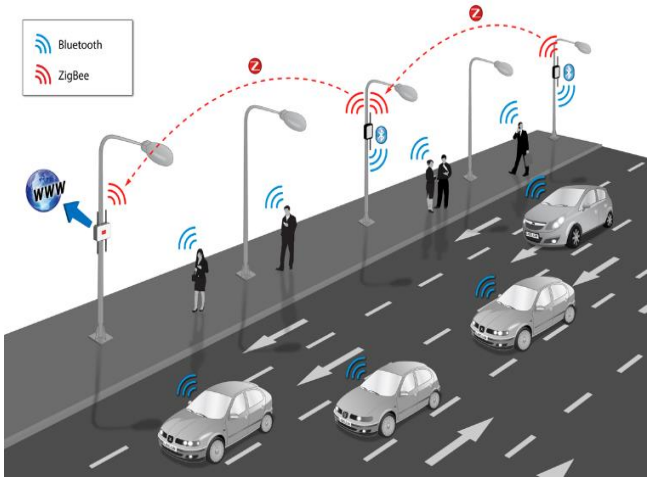


Figura 2. Modelo de comunicación entre vehículos, personas y alumbrado público en ciudades inteligentes (Pye, 2020)

La conexión de todas las cosas en las ciudades permite tener un control mayor sobre todas las actividades y procesos que se realizan en ésta. De esta manera se puede optimizar el consumo energético, detectar problemas en algún dispositivo o zona con rapidez, mejorar el flujo de las actividades como el tráfico, entre muchas ventajas más

Trabajos en el Área

Dado que el desarrollo de ciudades inteligentes se ha convertido en un prometedor avance, muchos investigadores y emprendedores destinan sus trabajos a introducir proyectos en ciudades que desconocen el tema. Por ejemplo, en 2015, Loyola Pinos y su equipo de trabajo desarrolló un manual para aplicación de tecnología Zigbee en edificios de Cuenca, Ecuador. (Pinos et al., 2015) Del mismo modo, se realizó un estudio en la ciudad de Guayaquil para establecer un modelo de ciudad inteligente a base de las características de otras ciudades ya existentes como Nueva York, Singapur o Madrid. (Montiel Salazar, 2019)

En el campo de gestión de residuos han destacado 2 productos muy significativos y aplicables a ciudades inteligentes para tratar los residuos. Primeramente, EcoBox es una máquina creada en Colombia para reciclar botellas plásticas. El principio que permite impulsar a las personas al reciclaje y recolección de botellas es que la máquina paga a los usuarios por depositar las botellas. Este producto ha crecido de forma que en la actualidad se venden versiones para recolectar botellas de vidrio, plástico y aluminio. (La Nota Positiva, 2019)

Figura 3. EcoBox, máquina que paga por reciclar (La Nota Positiva, 2019)

También se destaca a ECO-BITS, siendo una gran iniciativa por parte de la consultora tecnológica Druidics en asociación con MicroGestión, lanzaron este producto con tecnología cognitiva de IBM Watson. El sistema de ECO-BITS (figura 4) facilita tremendamente la gestión de residuos dado que, mediante sensores, entrega datos en tiempo real del nivel de llenado de sus contenedores de basura empleando una conexión directa a internet, permitiendo así poder monitorearlos mediante cualquier dispositivo con conexión a internet.

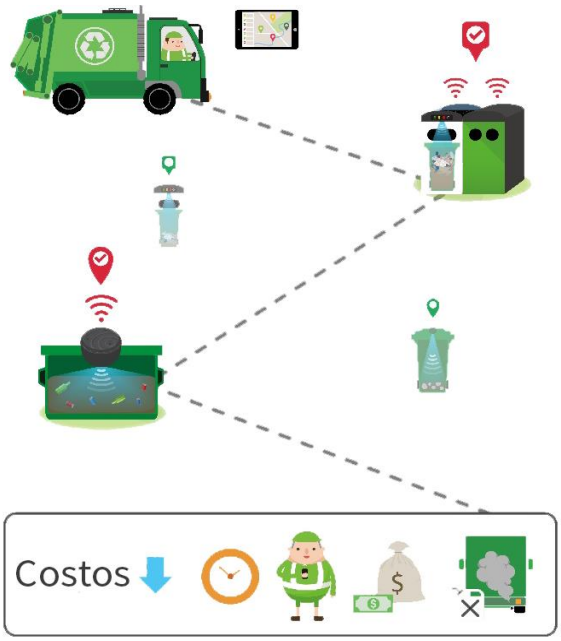


Figura 4. ECO-BITS, Proceso de gestionamiento de residuos (Ruggeri, 2018)

La figura 4 ilustra cómo los camiones recolectores trazan su ruta óptima en base a la detección en tiempo real de los contenedores llenos y proceden a su recolección. En la parte inferior se observa que este proceso reduce costos de sobretiempos, se optimiza la recolección y se reduce las emisiones contaminantes al tomar rutas óptimas de recolección.

Figura 5. Funcionamiento de ECO-BITS (Manaure, 2019)

Situación Actual en Gestión de Residuos en Quito

La ciudad de Quito cuenta actualmente con una población aproximada de 2’781.641 de personas, lo cual representa el 16% de la población nacional. (La Cámara, 2020) Esta gran concentración de personas genera altas cantidades de residuos que son gestionados por el municipio de la ciudad mediante la ubicación de contenedores de basura en distintos puntos, que posteriormente son visitados por camiones para su recolección. De forma rutinaria se realizan recolecciones diarias (EMASEO EP, 2017) para todo Quito.

La empresa EMASEO trabaja con un proceso mecanizado para la recolección en el que los camiones toman mecánicamente los contenedores de basura y los vacían en su depósito como se muestra en la figura 6.

Figura 6. Proceso mecanizado de recolección de basura por EMASEO (EMASEO EP, 2016)

Este proceso mecanizado de recolección presenta varios puntos clave para el diseño rutas, que según lo menciona la página oficial de EMASEO son los siguientes:

- "Se considera que cada familia Quito está compuesta por 4 miembros.
- Cada persona promedio genera 0,85 Kg por día.
- 1 contenedor atiende a 40 familias.
- Se determina que el usuario no camine más de 100 metros al contenedor más cercano.
- Todo contenedor está identificado con un código y es georreferenciado.
- Se lleva un control de ubicación y uso del contenedor, con esto se establece la frecuencia de recolección y lavado de las rutas.
- Se prioriza la ubicación de los contenedores sobre calzada con línea de parqueo.
- Accesibilidad por la derecha en el sentido del tránsito." (EMASEO EP, 2016)

Los datos que se consideran son refrescados cada 90 días (EMASEO EP, 2016) para determinar, en base al historial de recolección y comentarios de los ciudadanos, la ubicación exacta donde se fijará al contenedor.

Planteamiento del Problema

La conectividad de cada elemento en la ciudad permite que todo trabaje en conjunto y se vaya retroalimentando conforme pasa el tiempo, de forma que es la ciudad misma la que se mantiene y facilita la vida de sus habitantes. Una ciudad inteligente brinda elevados ahorros en varios aspectos como: ahorro energético optimizando procesos y reducción de agentes contaminantes por vehículos mediante un monitoreo continuo del flujo de tráfico u optimización de las rutas diarias que realizan los distintos vehículos de trabajo como municipales.

Específicamente, en este trabajo se abordará la problemática generada en el gestionamiento de residuos. Distintas ciudades en Ecuador han presentado quejas por desbordamiento de basura en los contenedores municipales, incluso con bolsas de basura por fuera. Los siguientes artículos de La Hora en 2019 y El Telégrafo en 2017 lo verifican:

"Desde varios sectores de la ciudad, las personas sintieron molestias por la acumulación de los desechos tanto al interior como en los alrededores de los contenedores de basura. Inclusive se denunció que, en el sector del Tajamar Regalado, la basura estuvo en el contenedor por ocho días." (La Hora, 2019)

"Los quiteños están preocupados por la acumulación de desechos en varios sectores de la ciudad que, aseguran, empeoró después de los festejos por Navidad." (El Telégrafo, 2017)

Justificación

Se plantea implementar un sistema de detección del nivel en los contenedores de basura para monitoreo remoto, optimización en rutas de recolección, puntos estratégicos para ubicación de estos y análisis de modalidades de trabajo que se podría adoptar en la

gestión de residuos para diferentes temporadas del año. Además, al optimizar las rutas de los camiones recolectores se conseguiría reducir consumo de combustible, menos emisiones de CO2 y reducir costos de mantenimiento.

Alcance

El alcance del siguiente trabajo es realizar un prototipo de un sistema que permita observar en tiempo real, dentro de una interfaz, el nivel de llenado de 2 contenedores de basura aplicando módulos Xbee S2C para comunicación inalámbrica. De esta forma se podrá monitorear de forma remota el nivel de llenado de los contenedores.

Objetivo del Estudio

- Diseñar el prototipado para un sistema que detecte el nivel de llenado en tiempo real a contenedores de basura mediante sensores y comunicación inalámbrica por protocolo ZIGBEE para la optimización de procesos en los sistemas de gestión de residuos en ciudades inteligentes.

Objetivos Específicos

1. Demostrar el funcionamiento de una red de comunicación inalámbrica con topología “Cluster Tree” empleando antenas de protocolo Zigbee, para que se establezca una red de comunicación inalámbrica entre contenedores de basura.
2. Diseñar un prototipo para el cuerpo de las antenas mediante el programa de Autodesk, INVENTOR.
3. Proponer un rediseño a contenedores municipales para la posible aplicación de este sistema en la ciudad de Quito, mediante el programa Autodesk, Inventor.
4. Crear el prototipo inicial que tendría un router y duplicarlo para poder realizar las pruebas respectivas en verificación de la funcionabilidad del sistema.

5. Diseñar una interfaz gráfica de usuario que permita observar el nivel de llenado en contenedores de basura mediante el programa Processing (versión 3.5.3.).

Hipótesis

Un sistema de comunicación inalámbrica de protocolo IEEE 802.15.4 (Zigbee) de corto alcance y bajo consumo energético permitirá monitorear en tiempo real y de forma remota el nivel de basura en los contenedores comunes de Quito logrando un ahorro importante en recursos humanos y económicos.

Estado del Arte

Protocolo IEEE 802.15.4

La IEEE (Institute of Electrical and Electronics Engineers) es un instituto de ingenieros eléctricos y electrónicos, como lo establece su nombre, que ha designado varios protocolos para las comunicaciones inalámbricas. En febrero de 1980 se crea la familia de estándares 802 (Microsoft, 2018), de allí obtiene este nombre.

Específicamente, el protocolo 802.15.4 es un estándar que define el tipo de redes inalámbricas de área personal con bajas tasas de transmisión de datos en corto alcance (low-rate Wireless Personal Area Network, LR-WPAN) (Crespo, 2016). Las redes inalámbricas que se emplean bajo este estándar son normalmente para sensores de control que se manejen en baja potencia y complejidad, debido a que para esta actividad no se necesita altas cantidades de transferencia de datos. (Linero et al., 2015).

Basándose en el estándar de la IEEE 802.15.4 se crea un protocolo abierto llamado Zigbee por parte de miembros de la Zigbee Alliance. Está clasificado como parte de las redes de área personal inalámbricas (WPAN). Funciona en rangos de frecuencia de 868MHz y 900MHz, dentro de una banda libre de 2.4GHz. Estos dispositivos al transferir

bajas tasas de datos son orientados a aplicaciones de grandes redes, llegando a un máximo de 65000 nodos por red. (Agudelo et al., 2018)

Nodos de Red Zigbee

Las redes Zigbee pueden tener diferentes configuraciones en base al tipo de uso que se las vaya a dar. Básicamente existen 3 tipos de nodos en las redes:

- **Coordinador:** es el dispositivo central de la red que manda y recibe los datos de los router y end device (Dispositivo final). Establece las comunicaciones en la red y procesa los datos obtenidos. (Crespo, 2016) Solamente existe 1 coordinador en cada red y siempre debe haber uno.
- **Router:** nodo que crea y mantiene la información sobre la red para determinar la mejor ruta para enrutar un paquete de información. (Crespo, 2016)
- **End Device:** Reciben únicamente la información de su nodo padre, y no envían información a otro router. Su consumo es mínimo al tener la capacidad de apagarse automáticamente cuando no esté enviando o recibiendo datos, y suele estar alimentado por una fuente que es la que actúa en el circuito final. (Crespo, 2016)

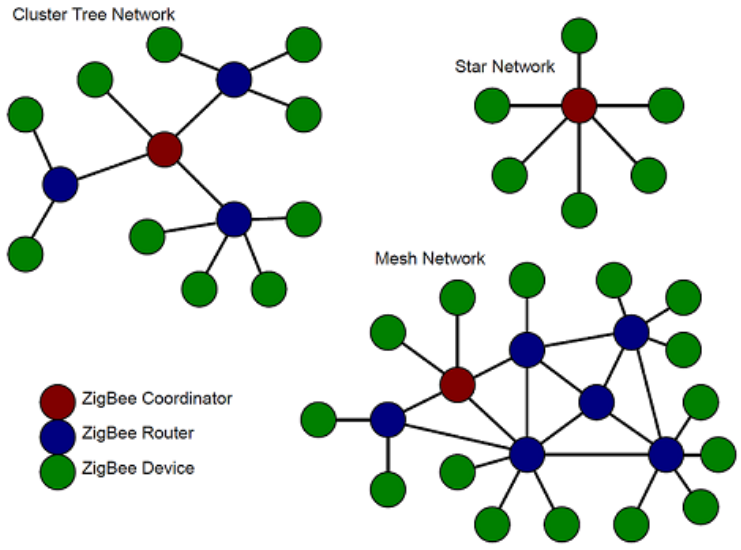


Figura 7. Topología de redes Zigbee (Cruz, 2018)

La red “Mesh” es una configuración que asegura la transferencia de datos por un alto número de routers que se comunican entre sí y no únicamente con el coordinador. De esta manera existirán múltiples rutas para la transferencia de datos, permitiendo así que el mensaje llegue al coordinador incluso cuando alguno de los routers deje de funcionar. En “Cluster tree” los routers son comandados solamente por el coordinador, de forma que su rango de cobertura reduciría en comparación con “Mesh”.

Aplicaciones de Redes Zigbee

El estándar Zigbee es conocido por varias aplicaciones en el campo de la domótica, siempre buscando un mayor monitoreo y control de áreas. Algunas de las aplicaciones más comunes son:

- “**Gestión y eficiencia energética:** para proporcionar más información y control de uso de energía.
- **Automatización de casas:** para proporcionar una administración flexible de iluminación, calefacción y refrigeración, seguridad, etc.
- **Automatización de edificios:** donde se puede integrar y centralizar la administración de luces, calefacción, refrigeración y seguridad.
- **Automatización industrial:** para ampliar la confiabilidad de los sistemas de control de procesos y fabricación existentes.” (Albornoz & Soto, 2018)

La tendencia actual está dada por sistemas de red para aplicaciones que pertenecen al Internet de las Cosas (IoT), definida como una red mundial de objetos interconectados basada en protocolos de comunicación estándar El costo de las tecnologías de sensores ha disminuido y con ello, el desarrollo de redes inalámbricas de sensores ha ido en aumento,

abarcando áreas de aplicación como educación, monitoreo ambiental, industria médica, agricultura y domótica. (Cruz, 2018)

Comparación con Otros Tipos de Red

Las comunicaciones inalámbricas por radiofrecuencia de corto alcance para distintas aplicaciones compiten entre Bluetooth, Wifi y Zigbee. Estos tres estándares tienen únicamente una similitud, y es que todos operan aproximadamente bajo la misma frecuencia que es de 2,4 GHz. (Elinoff, 2017) La mayor ventaja que ofrece el protocolo de Zigbee es que puede formar una red de dispositivos vinculados entre ellos y comandar sus datos a través de un nodo central, otorgando mayor cobertura en rango de distancia. (Horner, 2019)

Tabla 1

Comparativa entre Wi-Fi, Bluetooth y Zigbee (Fontuño, 2012)

	Wi-Fi	**Bluetooth**	**Zigbee**
Velocidad	<50 Mbps	1Mbps	<250 kbps
Num. Nodos	32	8	255 / 65535
Duración batería	Horas	Días	Años
Consumo Transmitiendo	400 mA	40 mA	30 mA
Consumo en reposo	20 mA	0,2 mA	3 uA
Precio	Caro	Medio	Barato
Configuración	Compleja	Compleja	Simple
Aplicaciones	Internet en edificios	Informática y móviles	Domótica y monitorización

En este proyecto se decide emplear el estándar de Zigbee puesto que, en comparación con los demás, se puede transferir datos a largas distancias mediante redes de topología “Cluster Tree” o “Mesh”, (figura 7) su consumo energético es mínimo y la

cantidad de datos transferidos es la indicada para el control de sensores de nivel que se emplearán.

Configuración Zigbee

Los módulos Zigbee tienen la capacidad de comunicarse mediante dos modos distintos:

Modo AT.

Este modo también es conocido como modo transparente. Al configurar los módulos con este modo, todo lo recibido por el puerto serie de la antena "A" (figura 8) será exactamente enviado a la antena "B" (figura 8), quién la recibirá igual de forma serial en el orden enviado. Este modo es muy sencillo pero limitado a conexiones punto a punto, es decir, redes pequeñas. También, trabajando en este modo es necesario configurar en las antenas la dirección de destino (DH y DL) a la que llegará el mensaje en el programa XCTU, ya que sólo podrá ser enviado a una sola antena. (Crespo, 2016)

Figura 8. Comunicación por modo AT (Crespo, 2016)

Modo API.

Este modo transfiere los datos mediante una serie de paquetes organizados (tramas) y con un orden determinado (figuras 10 y 11). Permite armar redes mucho más robustas entre varios módulos y controlar la configuración de todos los módulos de la red mediante el coordinador. Entre las ventajas que ofrece este modo de comunicación descritas por Crespo en el 2016:

- “Configurar módulos locales y remotos en la red y sin necesidad de entrar en modo comando.
- Comunicar con uno o varios destinatarios
- Identificar el módulo que ha mandado la comunicación
- Recibir el estado de la transmisión de los paquetes
- Obtener la fuerza de la señal de los paquetes recibidos
- Hacer gestión y diagnóstico de la red.
- Hacer funciones avanzadas domo actualización de firmware remota.” (Crespo, 2016)

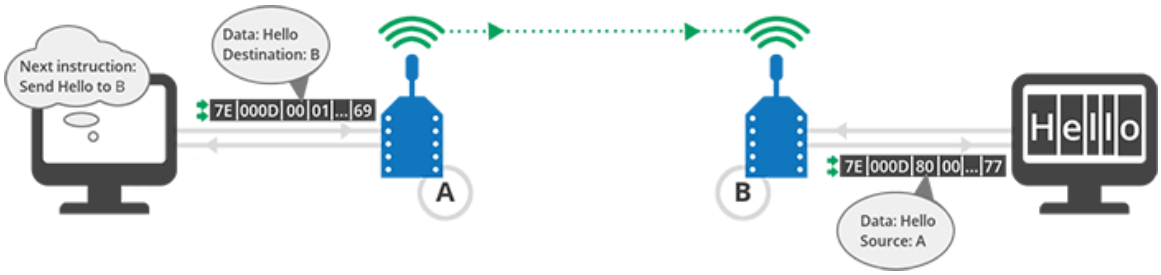


Figura 9. Comunicación por modo API (Crespo, 2016)

En la figura 9 se observa cómo la antena “A” transfiere una trama o paquete con datos específicos direccionados a la antena “B”, la cual recibe la trama e identifica que viene de “A”, extrayendo los datos enviados que en este caso son “HELLO”.

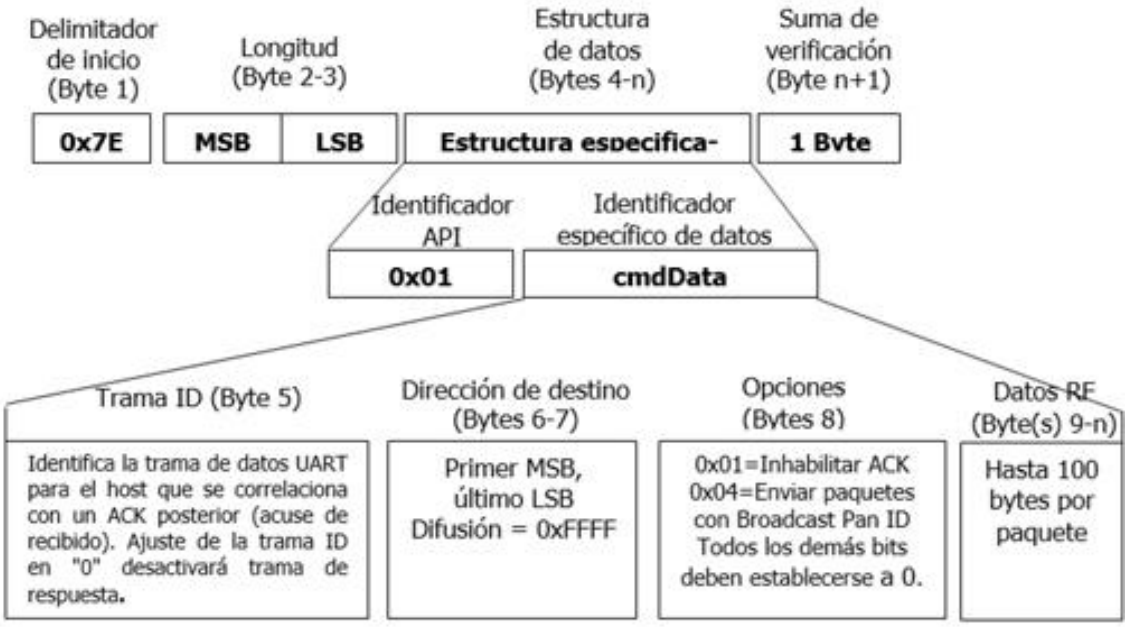


Figura 10. Estructura de trama para transmitir datos (Vera Romero et al., 2015)

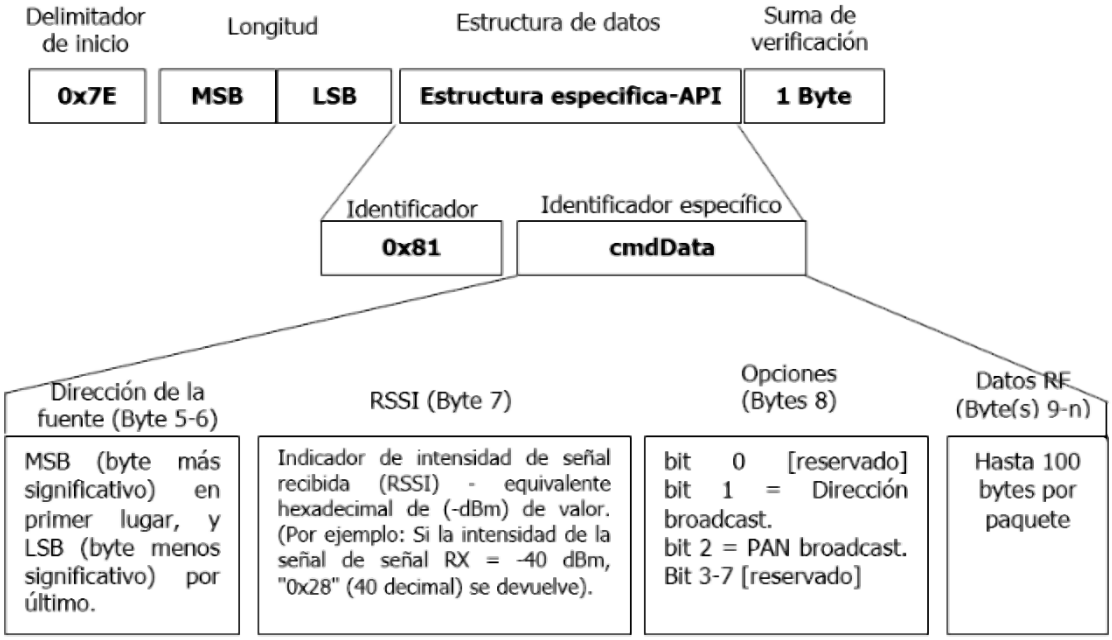


Figura 11. Estructura de trama para recibir datos (Vera Romero et al., 2015)

Módulos Xbee

Para la selección de los módulos que se empleará para este trabajo, se partió de la tabla 2.

Tabla 2

Familias de módulos por radiofrecuencia Xbee de DIGI (Hubbard, 2017)

Producto	Frecuencia	Factor de forma	Protocolo	Multipunto	Mesh	Programable	Puerto de comunicación	Modem
Xbee Zigbee	2.4 GHz	TH/SMT	Zigbee		x	x	x	x
Xbee 802.15.4	2.4 GHz	TH/SMT	802.15.4	x			x	x
Xbee DigiMesh 2.4	2.4 GHz	TH	DigiMesh		x		x	x
Xbee Wi-Fi	2.4 GHz	TH/SMT	802.11 b/g/n	x				
Xbee-PRO 900HP	900 GHz	TH	P2P/DigiMesh	x	x	x	x	x
Xbee-PRO 868LP	868 GHz	SMT	P2P/DigiMesh	x	x	x	x	
Xbee SX	900 GHz	SMT	DigiMesh	x	x			x

Se decide tomar a la familia de los módulo que se rigen bajo el protocolo Zigbee por las características que se presentan en las tablas 1 y 3. A continuación se contrastará a los módulos Xbee de serie 1 y serie 2 de forma general.

- **Xbee Serie 1:** permiten la conexión de redes punto a punto (Par) o punto-multipunto (Estrella) según las vistas en la figura 7. Se caracterizan por su facilidad de programación y uso. (Cardador, 2017)
- **Xbee Serie 2:** estos módulos permiten crear redes de todo tipo de configuración, desde las punto a punto más simples hasta las Mesh más complejas con varios repetidores de señal o routers. De esta forma otorgan mayor rango de distancia para la cobertura y menor consumo energético por nodo de red. (Cardador, 2017)

Tabla 3

Características de Xbee S1 y S2 (Hussein et al., 2020)

	Series 1	**Series 2**
Distancia (m)	30	40
Rango Óptimo (m)	100	120
Corriente de transmisión/recepción (mA)	45/50	40/40
Firmware	802.15.4	Zigbee
Entradas digitales	8	11
Entradas análogas	7	4
PWM	2	No
Topología punto a punto	Si	Si
Topologías Mesh y Cluster Tree	No	Si

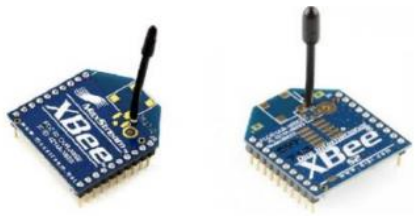


Figura 12. Xbee S1 y S2 (Hussein et al., 2020)

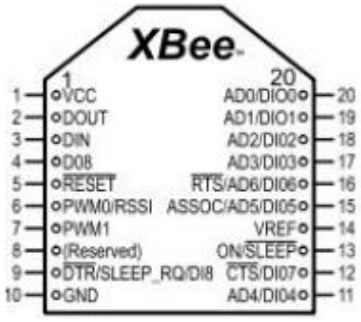
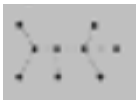


Figura 13. Diagrama de pines del Xbee S2 (Crespo, 2016)

Tabla 4

Características de las familias Xbee (Hubbard, 2017)

Familia	Frecuencia	Protocolo	Descripción	Rango de visión	Factor de forma	Costo	Tasa de datos	Consumo de corriente Tx/Rx	Referencia de Hardware
Xbee Wi-Fi		IEEE 802.11 	Conectividad punto-multipunto con capacidad de guardar datos en la nube.	N/A		$ 29	1 – 72 Mbps	309mA/ 200mA	S4B
Xbee S2C DigiMesh 2.4	2,4 GHz	DigiMesh 	Red DigiMesh, bajo costo, bajo consumo	1200 m	 Antena	$ 17,5	250 Kbps	33mA/ 28mA	S2C
Xbee-PRO S2C DigiMesh 2.4			DigiMesh de rango extendido	3200 m		$ 28,5		120mA/ 32mA	
Xbee 802.15.4		802.15.4 	Conectividad punto-multipunto por dispositivo.	1,2 km	 Montaje superficial	$ 17,5	250 Kbps	33mA/ 28mA	
Xbee PRO 802.15.4			Versión de rango extendido punto – multipunto	1,2 km		$ 28,5		120mA/ 32mA	
Xbee Zigbee			Red de mallado Zigbee, bajo costo, bajo consumo.	1,2 km		$ 17,5	250 Kbps	33mA/ 28mA	
Xbee-PRO Zigbee		Zigbee Pro 	Zigbee de rango extendido	3,2 km		$ 28,5		120mA/ 32mA	
Xbee Zigbee-Thread Ready			Actualizable a protocolo de hilo (bajo costo, bajo consumo)	1,2 km	 Montaje superficial	$ 19,25	250 Kbps	33mA/ 28mA	S2D

Finalmente, de los módulos que se rigen bajo el protocolo Zigbee, se escoge al módulo Xbee S2C sombreado en amarillo en la tabla 4, puesto que en comparación con su versión Xbee S2C PRO, éste consume menos energía y es de menor costo, además de que su obtención es muy asequible.

Programa XCTU

La programación de las antenas Xbee puede manejarse a través del software XCTU creado por Digi, la compañía fabricante de los módulos de ambas series. Este programa permitirá otorgar el papel de "coordinador", "router" o "end-device" a cada módulo según como se desee armar la red. XCTU es una aplicación gratuita multiplataforma diseñada para permitir a los desarrolladores interactuar con los módulos Digi RF a través de una interfaz gráfica fácil de usar. (Digi, 2020)

Arduino

Las placas de Arduino son microcontroladores que detectan señales de sensores que afectan su entorno y/o envían instrucciones a actuadores para controlar ciertos procesos. Estas placas trabajan en conjunto con el programa de Arduino GENUINO que les otorga el código específico para que desarrollen distintas funciones en base a lo que se desee obtener o al tipo de circuito que se conecte. (Arduino, 2015)

Para este proyecto se empleará al Arduino UNO y NANO:

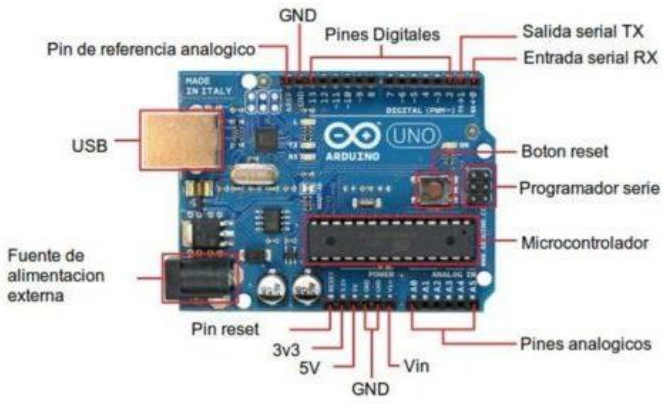


Figura 14. Partes de Arduino UNO (Ingeniería Mecafenix, 2017)

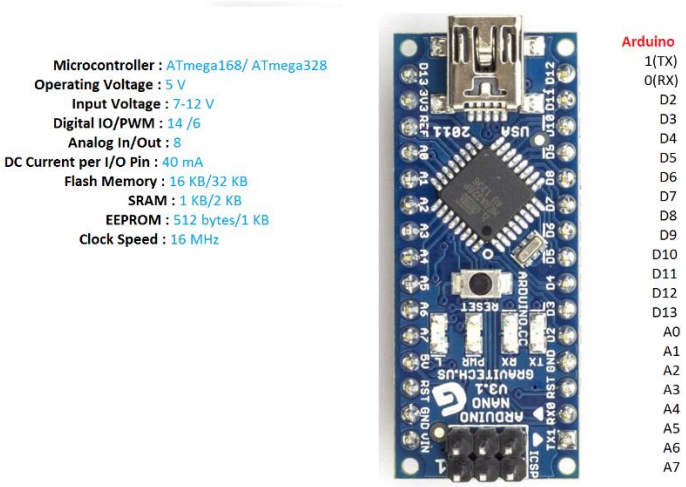


Figura 15. Partes de Arduino NANO (Cheppali, 2014)

Tabla 5

Consumo energético de distintos modelos de Arduino (Prometec, n.d.)

Modelo	Consumo en mA	Duración de una batería de 1200mAh
Arduino UNO	46	26 horas
Arduino MEGA	93	Casi 13 horas
Arduino DUE	75	16 horas
Arduino NANO	15	80 horas

La tabla 5 muestra los consumos energéticos de 4 diferentes modelos de Arduino. El consumo de Arduino NANO de 15 mA permitirá determinar el consumo total del circuito de las antenas.

Sensor Ultrasónico HC-sr04

Para la medición del nivel de sólidos dentro de contenedores se ha decidido emplear sensores de ultrasonido, específicamente para que sean compatibles con Arduino, se emplearan sensores HC-sr04 (figura 16)

Figura 16. Sensor de HC-sr04 (Llamas, 2015)

Figura 17. Diagrama de pines del sensor ultrasónico HC-sr04 (Patagoniatec, 2015)

El funcionamiento de este sensor de distancia es muy sencillo, se envía una onda ultrasónica mediante un disparador, la cual rebota contra un objeto y posteriormente es recibida por el pin Echo. De esta manera, el sensor calcula el tiempo que le tomó a la onda volver y en base a la velocidad del sonido, detecta su distancia. (Hernández, 2016)

$$velocidad\ (v) = \frac{distancia\ (s)}{tiempo(t)} \quad (1)$$

$$s = v * t \quad (2)$$

Dado que la velocidad del sonido viaja a 343 metros por segundo, entonces: (Hernández, 2016)

$$s\ [m] = 343\ \left[\frac{m}{s}\right] * t[s] \quad (3)$$

Figura 18. Diagrama de funcionamiento de sensor HC-sr04 (Components 101, 2017)

Tabla 6

Características técnicas del sensor HC-sr04 (Diosdado, 2015)

Características de sensores HC-sr04	
Alimentación	+5 V DC
Frecuencia de trabajo	40 KHz
Consumo (suspendido)	< 2mA
Consumo (trabajando)	15 mA
Ángulo efectivo	< 15°
Distancia	2cm a 400cm
Resolución	0,3 cm

*A partir de 250cm la resolución no es buena

Processing

Processing es un conjunto de un lenguaje de programación que se basa en Java. (Quijado, 2016) Una de sus principales funciones y para lo que se usará en este proyecto es la recepción de datos de forma serial y la creación de una interfaz de usuario que grafique los datos mencionados.

Metodología de Diseño Iterativo

El diseño iterativo se enfoca en constantemente ir perfeccionando un producto, proyecto o servicio. Inicialmente parte desde la identificación del problema que se

enfrentará mediante varias investigaciones que lo aborden. Posteriormente se procede a iniciar una fase de generación de ideas para generar una solución, ideas que se van analizando y paralelamente se va construyendo el diseño para una posible solución al problema. Se procede a realizar pruebas que permitan observar el comportamiento de la solución y se las evalúa. (Jerez, 2019)

La base de esta metodología yace en la repetición, permanecer en un constante análisis del sistema o producto propuesto para que cada vez se acerque más a su óptima funcionalidad. Incluso cuando se lanza al mercado el producto final, se debe seguir monitoreando la interacción de los usuarios con el sistema. De esta manera se sigue realizando mejoras para que con el paso del tiempo sea mucho más amigable con el usuario.

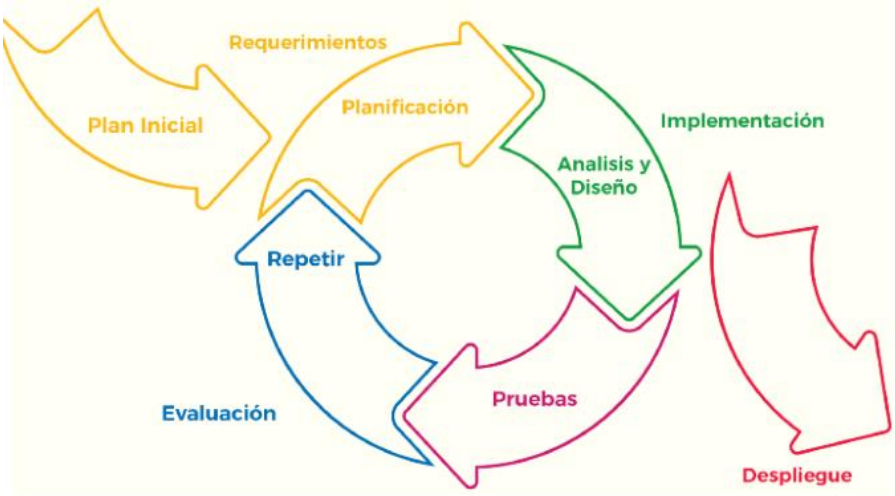


Figura 19. Ciclo de metodología iterativa (Jerez, 2019)

Entre algunas de las empresas más importantes que defienden y aplican a rigor esta metodología son Facebook, Amazon y Wordpress. Continuamente se encuentran en contacto con sus clientes o usuarios retroalimentando sus servicios, aumentando siempre su calidad.

Actualmente, existen varios proyectos que emplean esta metodología, tales como el de Catalín en 2016, que desarrolló un aplicativo móvil para apuestas deportivas. Menciona que se lo ha ido corrigiendo mientras se desarrollaba para darle mayor fuerza a su aplicación. (Catalín, 2016) De igual manera, Guaya en 2020 se basó en el método iterativo para desarrollar su trabajo acerca de la reducción de espacio de soluciones en algoritmos heurísticos.

Método Taguchi

Genichi Taguchi fue un alto exponente en el desarrollo de técnicas estadísticas para optimizar procesos de manufactura. Su principal aporte al área tiene como objetivo reducir la variabilidad en las salidas de los procesos y es conocida como Ingeniería Robusta. Específicamente, el método Taguchi se basa en minimizar el efecto de los ruidos que intervienen en los procesos. (Sejzer, 2016)

Cabe destacar que dentro de sus aportes se encuentra una técnica estadística para diseño de experimentos (DOE), en la cual se consigue identificar y cuantificar las causas de un efecto en las experimentaciones. Este proceso permite evaluar los factores controlables y no controlables que inciden en un experimento, para así, poder definir qué tan acertado es el resultado final (saliente) del sistema.

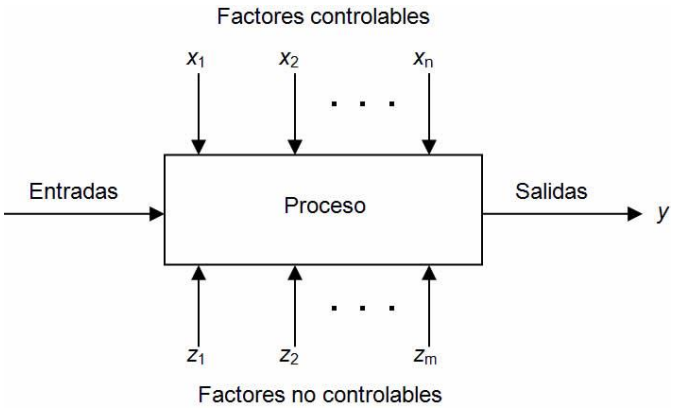


Figura 20. Diagrama para DOE

Método

Metodología Empleada en el Trabajo

La metodología empleada en este trabajo será la de "Diseño Iterativo", ya que se pretende iniciar a desarrollar un sistema que optimice las actividades o procesos dentro de la gestión de residuos sólidos en las ciudades. Específicamente en este trabajo, se efectuará un ciclo completo de esta metodología, como se observa en la figura 19, ya que se limitaran los resultados a un primer prototipo que demuestre la funcionalidad del sistema.

Se pretende que siguiendo con esta metodología, en futuros trabajos, se siga el desarrollo de este sistema repitiendo el ciclo de diseño iterativo, perfeccionando cada vez más el sistema hasta que se considere lo suficientemente robusto y se lance a su aplicación a gran escala.

Diseño Conceptual

Se plantea diseñar una red inalámbrica bajo el estándar Zigbee de topología "Mesh" (figura 7), con un gran número de nodos, que permita monitorear el nivel de llenado en los contenedores de basura dentro de la ciudad de Quito. Lamentablemente, debido a la difícil situación que actualmente afronta el mundo, el siguiente trabajo se limita a ejecutar sus experimentos con una red de 1 coordinador y 2 routers. De esta manera, la topología empleada tendrá que ser "Cluster Tree". Este tipo de topología permitirá demostrar el funcionamiento del sistema, a pesar de no ser la indicada para aplicar en ciudades enteras como Quito.

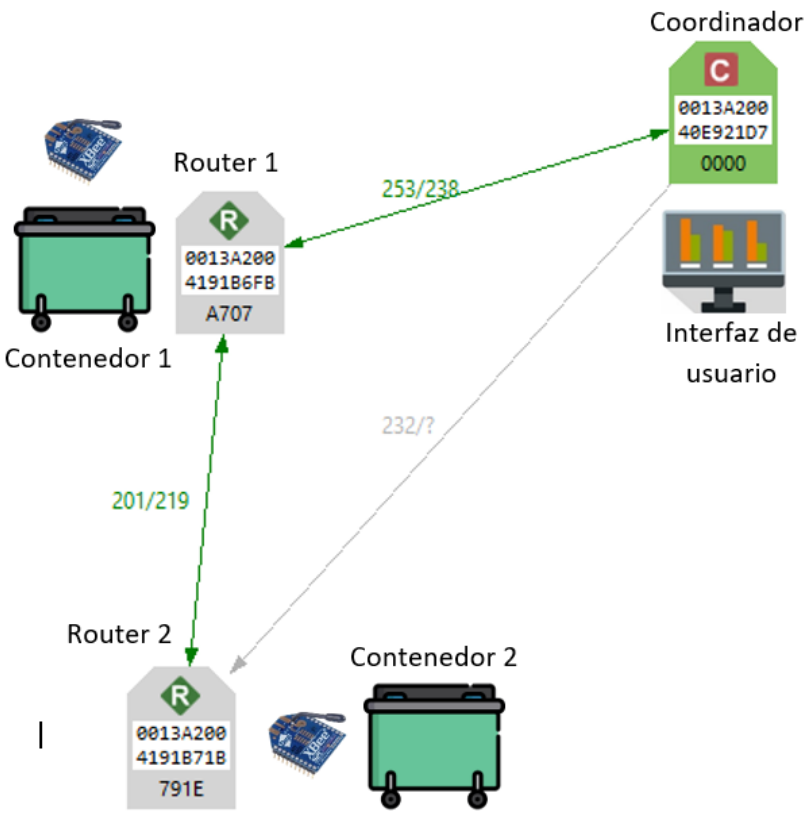


Figura 21. Diseño conceptual de la red "Cluster Tree"

El ciclo de trabajo inicia en el router, donde se detecta el nivel de llenado mediante un sensor HC-sr04. Los datos del sensor son procesados por un Arduino NANO, el cual también se encarga de convertir en tramas (figura 10) a los datos leídos. Las tramas son captadas a través del puerto serial del Arduino NANO hacia un módulo Xbee S2c configurado como "Router".

Se envían las tramas a través de la red hacia otro módulo configurado como "Coordinador", el cual las transmite a un puerto serial nuevo del Arduino UNO. Dentro de este Arduino, se interpreta las tramas byte por byte para extraer los datos leídos por los sensores y determinar a qué router pertenece la trama leída.

Finalmente, los datos del sensor son enviados por el serial original del Arduino UNO hacia Processing, donde son ilustrados en una interfaz gráfica para que un usuario

pueda monitorear remotamente el nivel de llenado en los contenedores que contengan los routers de la red.

Configuración de Módulos Xbee

Coordinador

Todos los módulos son configurados mediante el programa XCTU, ya sea para "Coordinador", "Routers" o "Dispositivos finales (End Device)". Para el coordinador se utilizó un Xbee S2, una versión muy similar al Xbee S2c, al cual se alteró únicamente los siguientes datos en su configuración de fábrica:

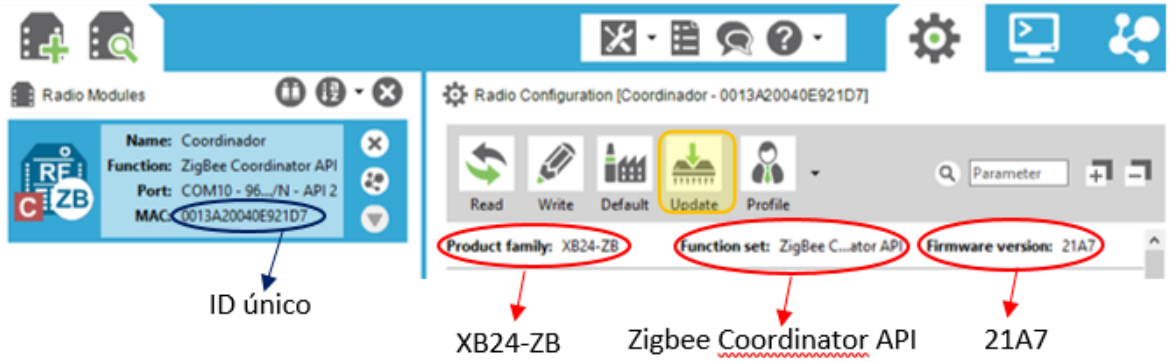


Figura 22. Configuración de firmware Coordinador API, Xbee S2

Para otorgar el tipo de nodo que tendrá un Xbee S2 se debe actualizar el firmware. Oprimiendo el botón "Update" sombreado amarillo en la figura 22 se puede realizar los cambios respectivos a las 3 categorías encerradas en rojo de la misma figura. Al haber realizado esto, también se debe configurar los siguientes campos en el firmware:

Tabla 7

Configuración de campos para Coordinador API, Xbee S2

Comando	Campo	Descripción
ID PAN ID	12344	Este campo puede contener cualquier # que se desee, pero debe ser el mismo en todos los nodos de la red.
DH Destination Adress High	0	Es la 1era fracción del ID a la que se direccionará el mensaje. Debido a que el coordinador no transmitirá mensajes, se configura como “0”.
DH Destination Adress Low	0	Es la 2da fracción del ID a la que se direccionará el mensaje. Dado que el coordinador no transmitirá mensajes, se configura como “0”.
AP API Enable	2	Se lo configura a “2” ya que esta opción permite asegurar que la estructura de la trama no se distorsione.

Routers

Ambos routers tienen la misma configuración para este trabajo. Los routers serán Xbee S2c con la siguiente configuración del firmware:

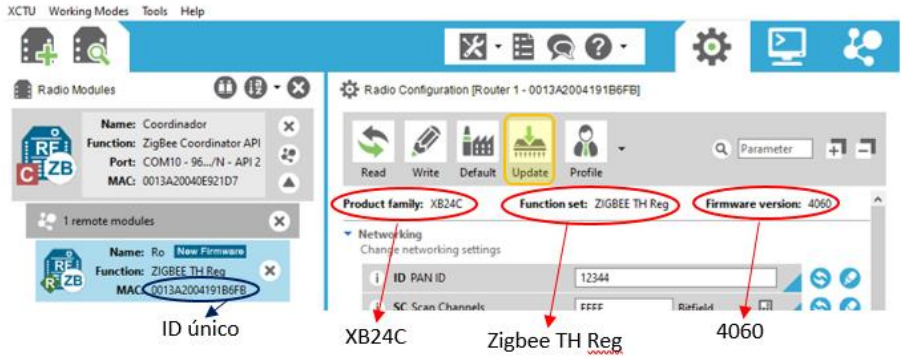


Figura 23. Configuración del firmware para Routers, Xbee S2c

Al actualizar el firmware como en la figura 23, se procede a configurar los siguientes campos:

Tabla 8

Configuración de campos para Routers API, Xbee S2c

Comando	Campo	Descripción
ID PAN ID	12344	El mismo que el Coordinador.
JV Channel Verification	Enable [1]	El router verifica el canal de comunicación que tendrá con el coordinador, asegurando una comunicación efectiva.
CE Coordinator Enable	Disable [0]	Hay que asegurarse que este parámetro este desactivado en los Xbee S2c, puesto que **aquí se determina si el módulo es router o coordinador.**
DH Destination Adress High	FFFF	Se coloca "FFFF" para que las tramas sean enviadas a toda la red (Broadcast) y se asegure que llegue al Coordinador sin importar la distancia.
DH Destination Adress Low	FFFF	Se coloca "FFFF" para que las tramas sean enviadas a toda la red (Broadcast) y se asegure que llegue al Coordinador sin importar la distancia.
AP API Enable	2	Se lo configura a "2" ya que esta opción permite asegurar que la estructura de la trama no se distorsione.
IR IO Sampling Rate	3E8	En hexadecimal quiere decir 1000 ms, es decir 1 s. Este es el tiempo de muestreo que tendrá el router al mandar sus tramas.

Software

El software de este sistema se basa únicamente en las plataformas de Arduino (versión 1.8.13) para manejo de las tramas enviadas y recibidas por los módulos Xbee tanto en los routers como el coordinador; y Processing (versión 3.5.3) para generación de una interfaz gráfica que permita una interpretación amistosa de los datos.

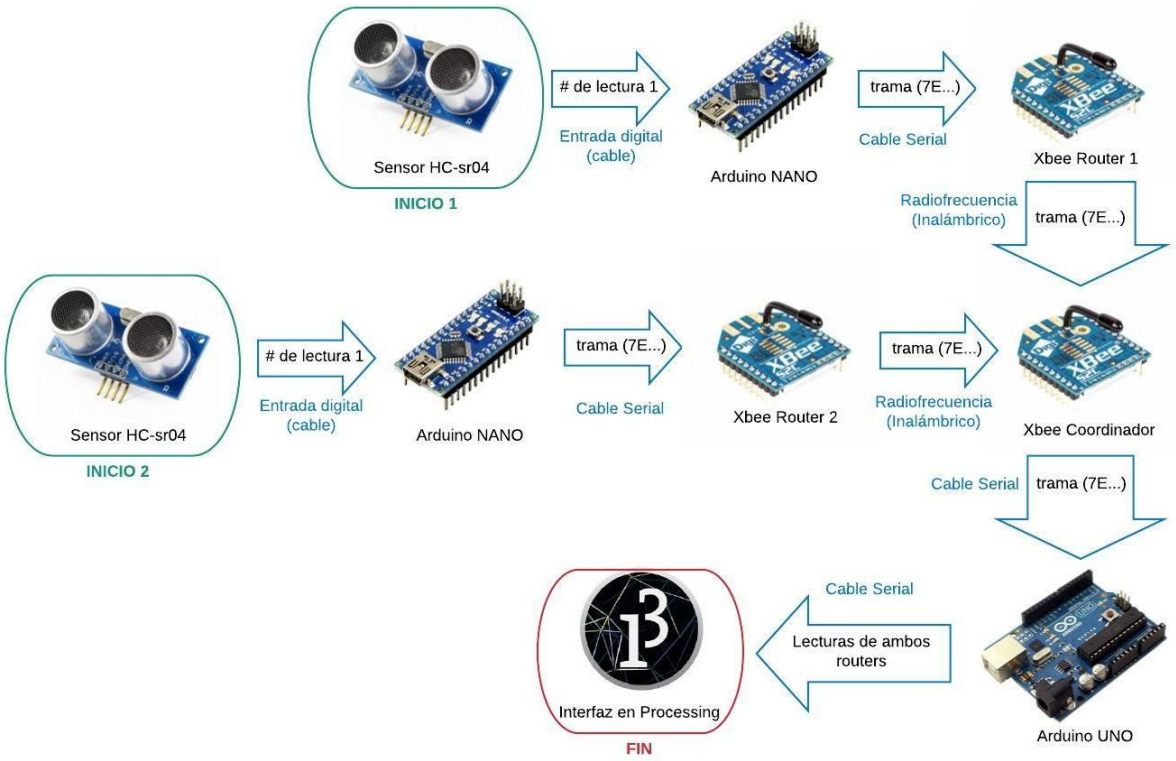


Figura 24. Flujo de envío de datos

La figura 24 muestra el flujo de los datos desde los sensores ubicados en cada router, hasta que sus lecturas llegan a la interfaz. Se muestra en qué estado se encuentra la información dentro de las flechas y a través de qué se la envía en los textos azules.

Routers (Envío de Datos)

Dado que la red está configurada en modo API (figura 9), los routers envían los datos mediante tramas. Para formular una trama específica para el envío de datos leídos por el sensor ultrasónico, se empleó el código del Anexo A. Ya que las lecturas varían entre números de 1, 2 y 3 cifras, es necesario configurar en el código 3 diferentes tramas para enviar por el Xbee. El siguiente diagrama de flujo (figura 25) otorga un mejor entendimiento del funcionamiento del código para los routers.

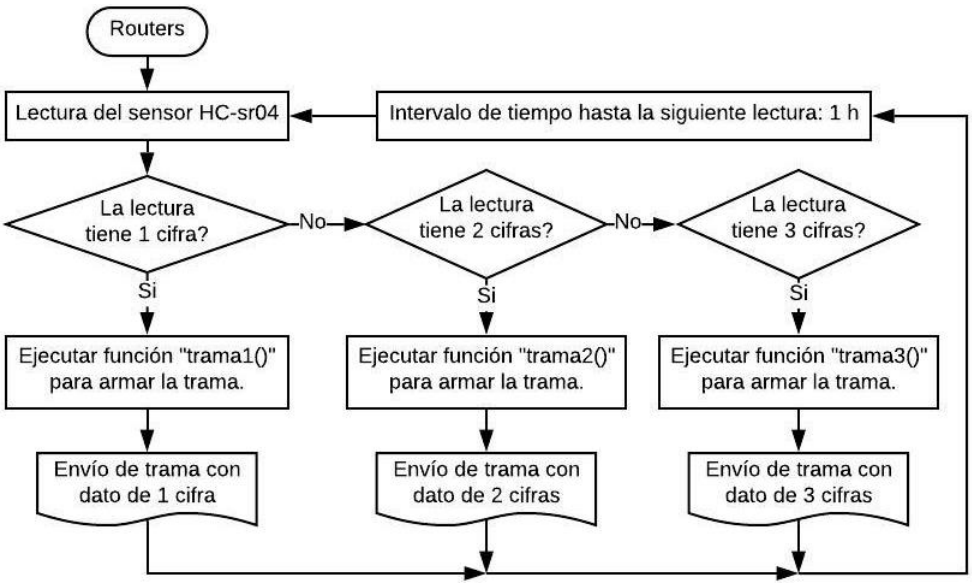


Figura 25. Diagrama de flujo de Routers

La longitud de las tramas varía en base al número de cifras que la lectura da, y también varía ciertos parámetros en su estructura. En la siguiente tabla se observa la estructura de las tramas que tendrían los calores de las lecturas. Cabe recalcar que todos los bytes en tramas enviadas por API se encuentran en idioma hexadecimal.

Tabla 9

Estructura de tramas de envío de datos

Byte	**Estructura de los 3 tipos de trama**			**Descripción**
	3 cifras	**2 cifras**	**1 cifra**	
0	0x7E	0x7E	0x7E	**Inicia Capa 1:** Delimitador inicial de la trama
1	0x00	0x00	0x00	**Inicia Capa 2:** "Longitud de trama" Byte inicial de capa "Length". (Normalmente es 0)
2	0x11	0x10	0x0F	2do byte de capa "Length". Este byte varía dependiendo del número de bytes en la capa 3 (data). Aumentando el # de cifras de la lectura, aumenta en 1 byte la capa 3. Por esta razón varían en las 3 tramas.
3	0x10	0x10	0x10	**Inicio de Capa 3:** "Data de la trama"

				Tipo de data que se enviará. 0x10 significa “Transmit Request”, de forma que los último bytes de esta capa son datos convertidos a hexadecimal y pertenecen a la lectura del sensor.
4	0x01	0x01	0x01	ID de la trama. Para este trabajo será siempre igual, puesto que cada router solamente enviará 1 tipo de dato.
5	0x00	0x00	0x00	
6	0x00	0x00	0x00	
7	0x00	0x00	0x00	
8	0x00	0x00	0x00	En estos bytes se ingresa la dirección de destino del mensaje, es decir, el ID del Xbee al que se desea mandar la data.
9	0x00	0x00	0x00	
10	0x00	0x00	0x00	Se lo configura de esta manera debido a que se desea enviar los datos en “Broadcast” a todos los nodos de la red. De esta forma se asegura que la trama llegue al Coordinador.
11	0xFF	0xFF	0xFF	
12	0xFF	0xFF	0xFF	
13	0xFF	0xFF	0xFF	
14	0xFE	0xFE	0xFE	
15	0x00	0x00	0x00	Radio del Broadcast: No se lo define para que la trama viaje por toda la red sin limitaciones.
16	0x00	0x00	0x00	Opciones: Byte para configurar posibles características extras que tenga la red. Se lo desactiva puesto que no es necesario.
17	(centena, HEX)	-	-	Centena de la lectura traducida a hexadecimal.
18	(decena, HEX)	(decena, HEX)	-	Decena de la lectura traducida a hexadecimal.
19	(unidad, HEX)	(unidad, HEX)	(unidad, HEX)	Unidad de la lectura traducida a hexadecimal.
20	(sumatoria)	(sumatoria)	(sumatoria)	**Inicio de Capa 4:** “Sumatoria” Este byte es la sumatoria de todos los bytes dentro de la capa 3 (Data).

Coordinador (Recepción de Datos)

El coordinador se encarga de recibir las tramas enviadas por los routers. Estas tramas son interpretadas en el código de Arduino del Anexo B, donde se las analiza, se determina desde qué router llegó la trama leída y se extrae los valores de las lecturas dadas por los sensores de cada router. El siguiente diagrama de flujo (figura 26) demuestra el funcionamiento del código para el microcontrolador del coordinador.

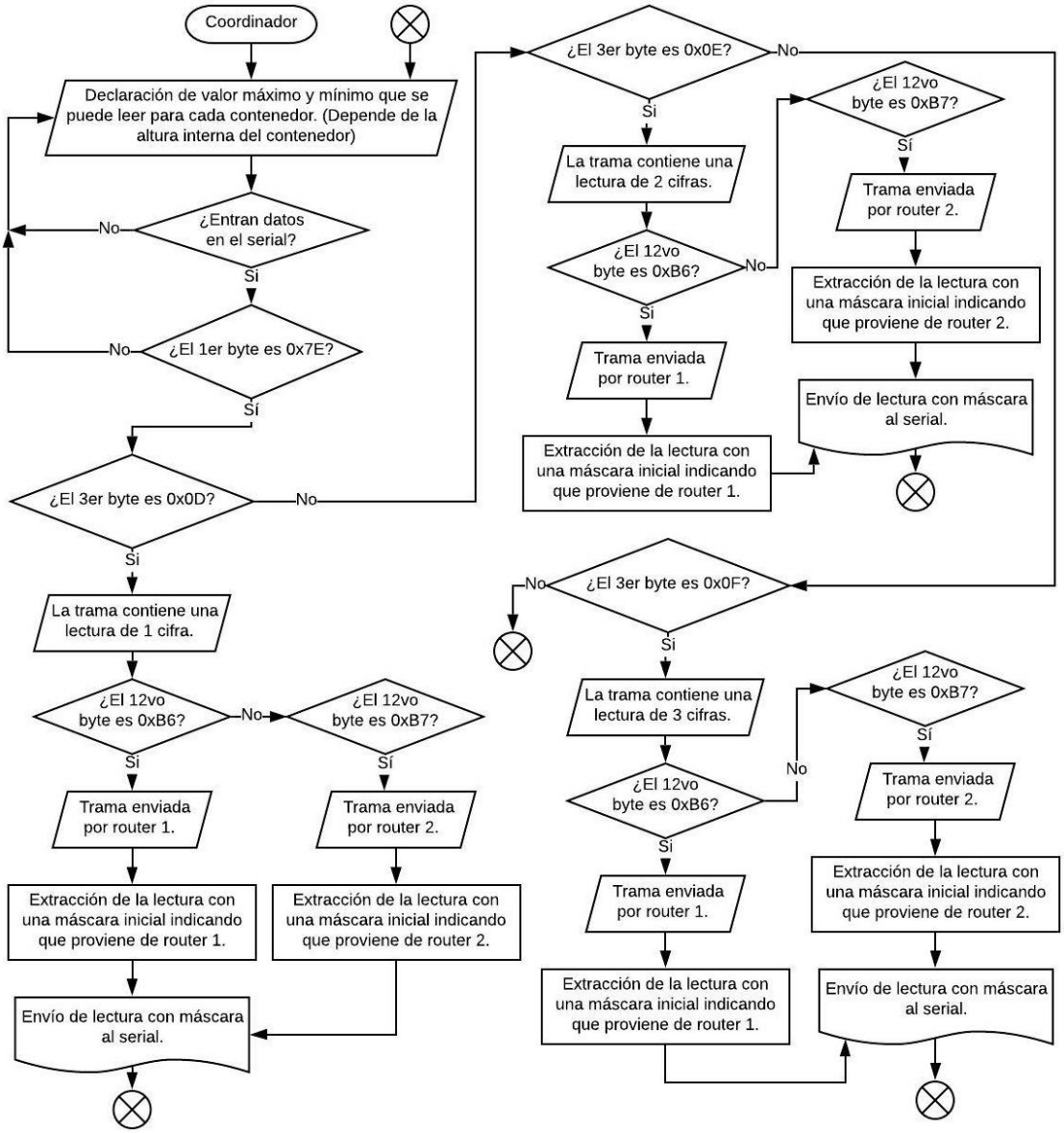


Figura 26. Diagrama de flujo del Coordinador

El diagrama de la figura 26 muestra los bytes que se toman en cuenta para la lectura de las tramas entrantes al coordinador. Primeramente, se verifica un inicio de trama con el byte 0x7E, para que las siguientes operaciones se hagan en orden y no se lea tramas desordenadas.

En segundo lugar, se determina si la trama contiene una lectura de 1, 2 o 3 cifras mediante el tercer byte de la trama captada. Como se explica en la tabla 9, este byte determina la extensión de la data, de forma que, hexadecimalmente permite conocer las cifras de la lectura.

Finalmente, con el doceavo byte, se determina de qué router viene la trama, puesto que este byte se encuentra en el conjunto de bytes que muestran la dirección de origen, es decir, el ID del router de origen. Cabe recalcar que los ID´s de los módulos Xbee empleados en este trabajo son:

- ID de Router 1: 0013A2004191B6FB
    - En bytes: 00, 13, A2, 00, 41, 91, B6, FB.
- ID de Router 2: 0013A2004191B71B
    - En bytes: 00, 13, A2, 00, 41, 91, B7, 1B.

Interfaz Gráfica

Es programada mediante Processing (versión 3.5.3) otorgando el nivel de llenado, en porcentajes, de cada contenedor que esté conectado en la red. Se empleó el código del Anexo C para el diseño de este cuadro de monitoreo.

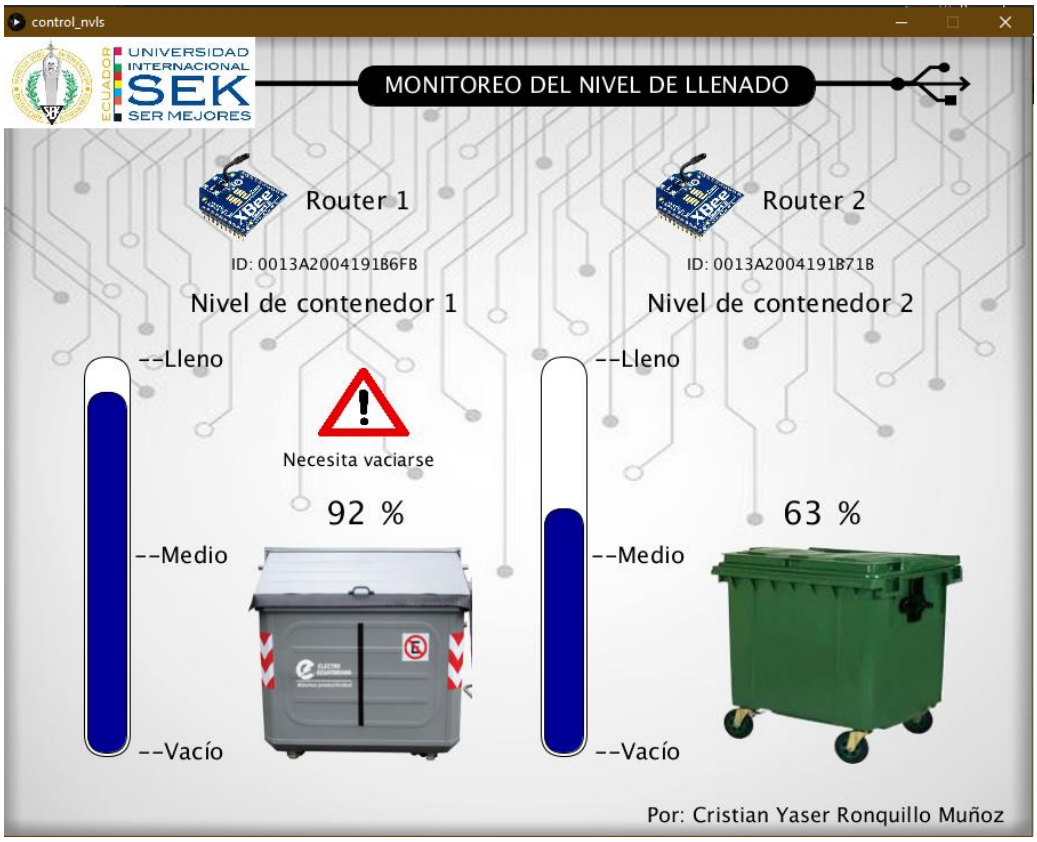


Figura 27. Interfaz gráfica para monitoreo del nivel de llenado

Primeramente, se toma los datos leídos por el serial del Arduino UNO conectado al módulo Coordinador y se los convierte en porcentajes tomando en cuenta las lecturas máximas que son aceptables de cada router. Las lecturas máximas son la altura interna del contenedor, que son medidas en cm por los sensores. Teniendo estos datos se crean barras que grafiquen el nivel de llenado de cada contenedor. También, la interfaz lanza una alerta cada vez que un contenedor sobrepase su 80 % de capacidad, mencionando que debe vaciarse. Este dato puede fácilmente ser alterado en la lista inicial de variables del código (Anexo C), según como lo desee el usuario.

Como se observa en la figura 27, la interfaz también ilustra el tipo de contenedor al que se instaló cada router. Cabe recalcar que para este trabajo no se emplearon los

contenedores ilustrados, debido a que se limita al diseño y funcionalidad del programa, mas no a la aplicación en contenedores municipales.

Hardware

Dentro del diseño del hardware se engloba el diseño eléctrico del coordinador y router, además de la construcción del prototipo para 2 antenas routers que permitan demostrar la funcionalidad del sistema y una propuesta de diseño para contenedores municipales que tengan integrado en su cuerpo las antenas.

Diseño para Coordinador

A nivel de hardware, el diseño del coordinador es simple, únicamente son necesarios 2 elementos mencionados en la tabla 10, además de jumpers y cables de datos USB para la computadora.

Tabla 10

Materiales empleados en Coordinador

Cant.	Nombre	Descripción	Costo	Imagen
1	Módulo Xbee S2	Recibe las tramas enviadas por los routers de toda la red.	$ 42	
1	Arduino UNO	Procesa las tramas del módulo y extrae las lecturas de sensores.	$ 10	
	COSTO TOTAL APROXIMADO		**$ 52**	

Dentro de este diseño no es necesario tomar en cuenta alguna fuente de alimentación ya que el Arduino UNO está continuamente alimentado por la computadora con 5V. En la figura 28 se puede observar el diagrama del circuito que tendrá y en la figura 29 un diagrama esquemático, ambos realizados mediante el programa Fritzing. El costo aproximado de los materiales se lo estima en base a costos en el mercado ecuatoriano.

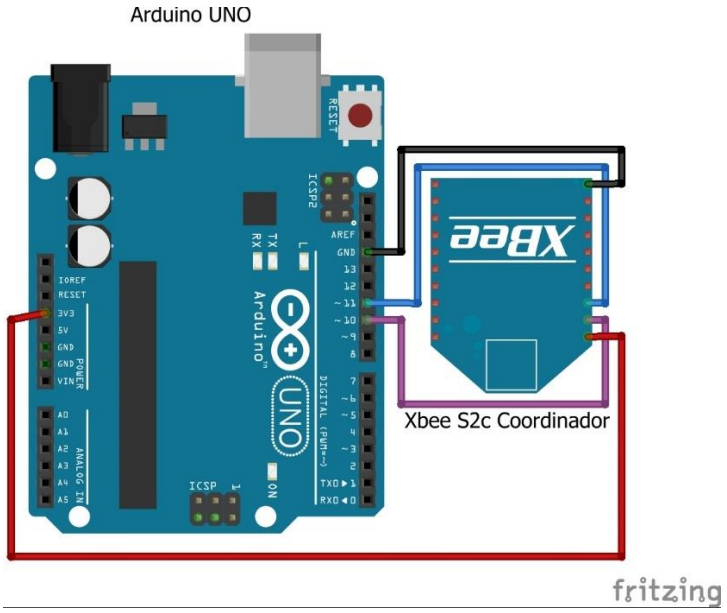


Figura 28. Diagrama de circuito para Coordinador

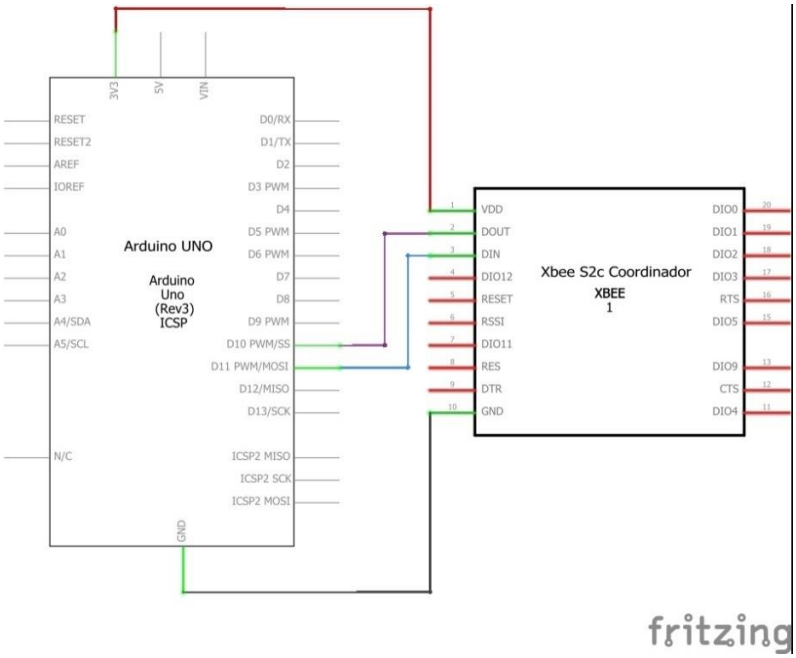


Figura 29. Diagrama esquemático del circuito para Coordinador

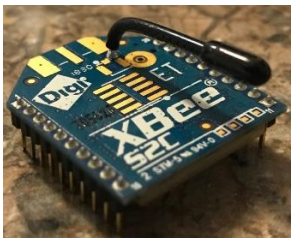
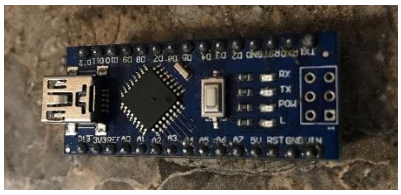
Para alimentar al Xbee se conectó a la fuente de 3.3 V del Arduino como se ve en el diagrama de circuito, dado que este es el voltaje en el que trabajan los módulos Xbee. Para la comunicación serial se conectó los pines 2 (DOUT) y 3 (DIN) del módulo a los pines 10 y 11 respectivamente, los cuales están programados como entradas y salidas seriales en el código del Anexo B. Cabe recalcar que no se conectó a los pines RX y TX del Arduino debido a que la comunicación serial que se da en éstos es la que se comunica con la interfaz. Aplicando este diseño con los materiales mencionados en la tabla 10, se consigue la construcción mostrada en el Anexo D.

Diseño para Router

El diseño de hardware para router es el mismo para todos los routers. En la siguiente tabla se puede observar los materiales que se emplearon para generar un prototipo de antena router en este trabajo.

Tabla 11

Materiales empleados en Routers

Cant.	Nombre	Función	Costo	Imagen
1	Módulo Xbee S2c	Módulo que toma las tramas con las lecturas y las envía al coordinador.	$ 42	
1	Arduino NANO	Microcontrolador que genera las tramas a partir de las lecturas del sensor.	$ 7	
1	Sensor de ultrasonido HC-sr04	Mide la distancia entre el tope superior interno de un contenedor de basura y el nivel en que se encuentra acumulada la basura. Al aumentar el nivel de basura, la lectura de distancia reduce al quedar menos espacio disponible.	$ 2,5	
1	Interruptor	Permite encender o apagar el circuito.	$ 0,4	
1	Fusible de vidrio (0,5 A)	Protege el circuito de posibles cortos que aumenten la corriente.	$ 0,1	

1	Portafusibles	Permite colocar y extraer fusibles con facilidad, sin tener que fijarlos.	$ 2,8	
2	Batería tipo 18650 3,7V 8800mAh	Conectando 2 baterías en serie se consigue 7,4V. Alimentan al circuito.	$ 5	
1	Porta baterías doble para baterías 18650.	Permite retirar y conectar las baterías con facilidad.	$ 1,5	
1	NANO IO Shield	Placa que acopla Arduino NANO con Módulo Xbee.	$ 13	
	TOTAL COSTO APROXIMADO		**$ 74,3**	

Para integrar los elementos ilustrados en la tabla 11 se empleó nuevamente el programa Fritzing que permite generar diagramas de circuitos, obteniendo los siguientes diagramas.

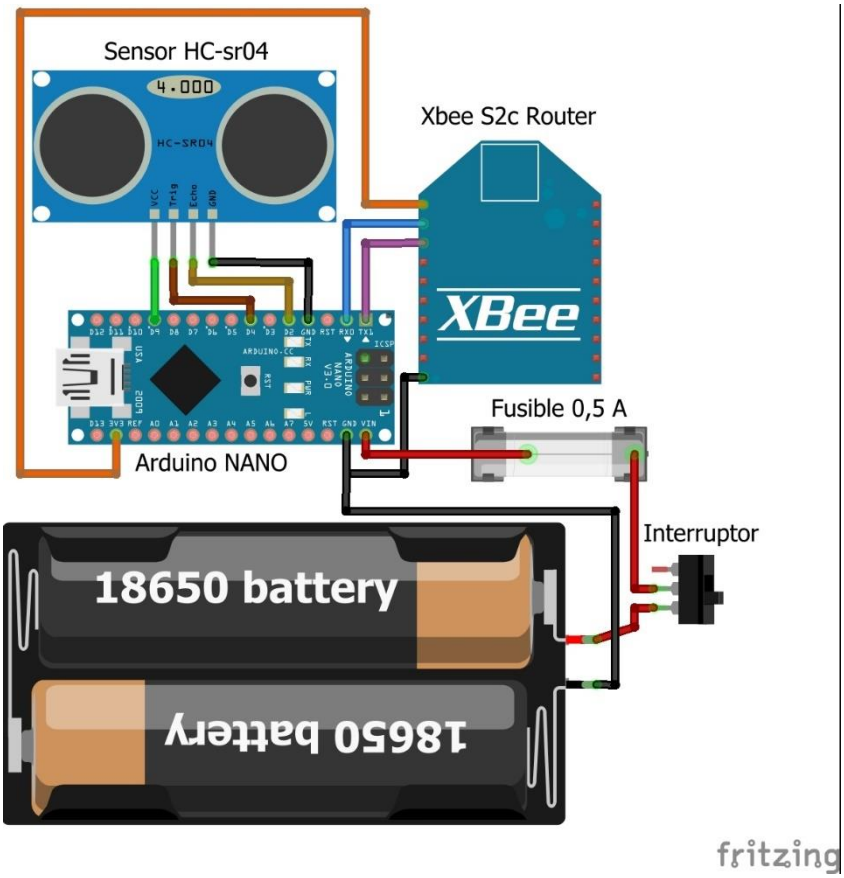


Figura 30. Diagrama de circuito para Routers

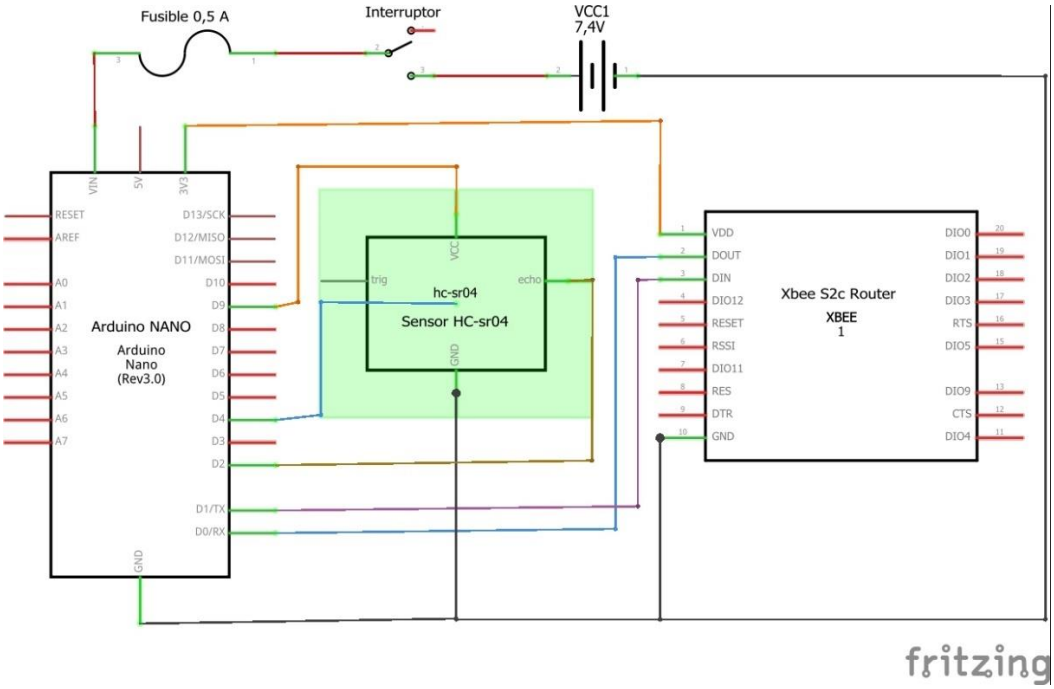


Figura 31. Diagrama esquemático de circuito para Routers

Fuente de Alimentación.

En la figura 15 (Cheppali, 2014) se observa entre las características del Arduino NANO que su voltaje de entrada por el pin VIN debe encontrarse entre 7-12 V, de forma que las baterías 18650 conectadas en serie otorgando 7,4 V son precisas para este dispositivo.

Para poder determinar la duración de nuestra fuente, es necesario conocer el consumo de corriente por hora del circuito, y tomar en cuenta en qué lapsos de tiempo se propone que estará en funcionamiento. La variable del voltaje se considera como estable por lo que no será necesario incluirla en los cálculos.

$$Consumo\ de\ corriente\ [mAh] = C.Xbee + C.Arduino\ NANO + C.Sensor \quad (4)$$

Se puede rescatar en la tabla 4 (Hubbard, 2017) que el consumo energético de los módulos Xbee S2c en pleno funcionamiento, es decir, al transmitir datos, es de 33mAh. También, en la tabla 3 (Hussein et al., 2020) se menciona un consumo en transmisión y recepción de datos, de 40mAh.

Sin embargo, Córdova en 2018, evaluó en su trabajo el consumo de sus módulos Xbee S2 transmitiendo datos, consiguiendo medir 22mAh. (Córdova, 2018) Dado que la estructura de su red es muy similar a la empleada en este trabajo, se decide tomar en cuenta un consumo para los módulos empleados de 22mAh.

Se propone que el router tome lecturas cada hora durante 30 segundos (modo ON). De esta manera, el sensor se encenderán solamente por los 30 segundos de medición, y permanecerá apagado (modo OFF) el resto del tiempo reduciendo el consumo de corriente. Ahora bien, se calculará el consumo promedio por hora:

$$C.de\ corriente\ por\ hora\ [mAh] = Consumo\ \ ON + Consumo\ \ OFF \quad (5)$$

$$Consumo\ ON\ [mAh] = (22mA + 15mA + 15mA) * \frac{1}{120} h = 0{,}43\ mAh \quad (6)$$

$$Consumo\ OFF\ [mAh] = (22mAh + 15mAh) * \frac{119}{120} h = 35{,}99\ mAh \quad (7)$$

$$C.de\ corriente\ por\ hora\ [mAh] = 0{,}43mAh + 35{,}99mAh = 36{,}42mAh \quad (8)$$

Ahora bien, se calculará la duración de las baterías empleadas para el prototipo diseñado:

$$Duración\ de\ baterías\ [días] = \frac{\frac{Carga\ de\ batería\ [mAh]}{Consumo\ de\ corriente\ por\ hora\ [mAh]}}{\#\ horas\ diarias\ en\ funcionamiento\ [h/día]} \quad (9)$$

$$Duración\ de\ baterías\ [días] = \frac{\frac{8800\ [mAh]}{36{,}42\ [mAh]}}{24\ [h/día]} \quad (10)$$

$$Duración\ de\ baterías\ [días] = 9{,}15\ días \quad (11)$$

Prototipo de antenas

Durante este trabajo también se construyó, a base de madera triplex, un prototipo para las antenas que proteja el circuito de golpes, permita una instalación compacta del circuito, y un correcto funcionamiento del sistema. El diseño se lo realizó mediante un 3D empleando el programa Inventor de Autodesk, del cual también se extrajo los planos detallados que se pueden observar en el anexo E. El resultado final de estos prototipos se pueden observar en la figura 48, o bien en el anexo F más detalladamente.

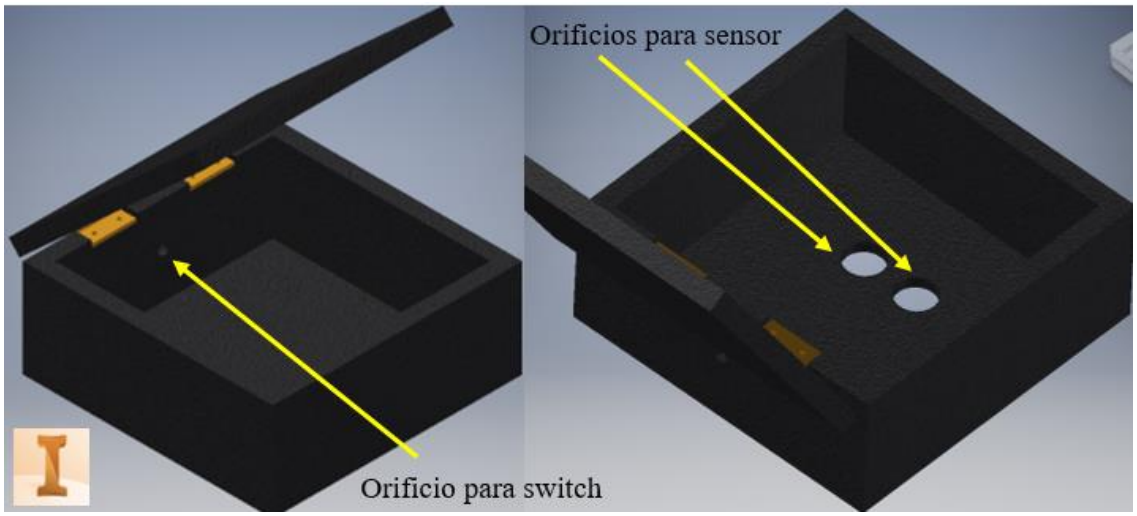


Figura 32. Diseño 3D del cuerpo para prototipo de Routers

Propuesta para Contenedores Municipales en Quito

Además de la producción de prototipos que demuestren el funcionamiento del sistema, se realiza una propuesta para el diseño de los contenedores municipales en Quito. El objetivo de esta propuesta es integrar el circuito de los routers en los contenedores de forma que cumplan con su funcionamiento y se protejan de factores externos que puedan alterar su rendimiento.

Para esta propuesta se empleó el programa Inventor de Autodesk, el cual permitió realizar un 3D de un contenedor municipal, y alterar ligeramente su diseño para poder integrar el circuito.

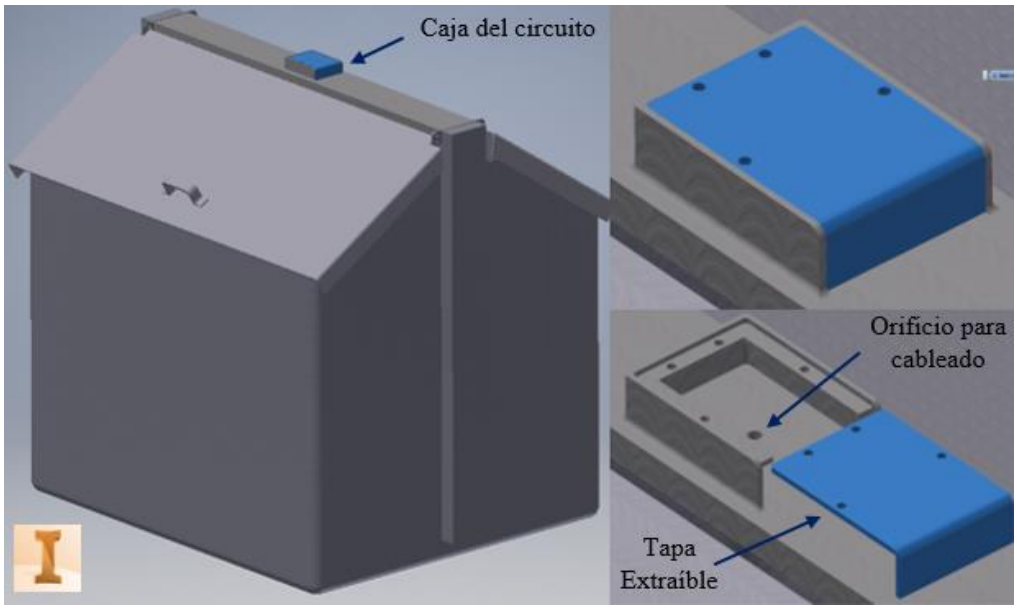


Figura 33. Propuesta 3D de integración de antenas a contenedores municipales de Quito

En la figura 33 se observa que el compartimento donde se alojaría el circuito de las antenas se ubicaría sobre la barra superior del contenedor, con lo que sería una tapa extraíble (elemento azul) que permita a agentes municipales intervenir al circuito cuando sea necesario. En medio del compartimento se ubicó un orificio que permita pasar el cableado a los sensores que se encontrarían en la parte interna.

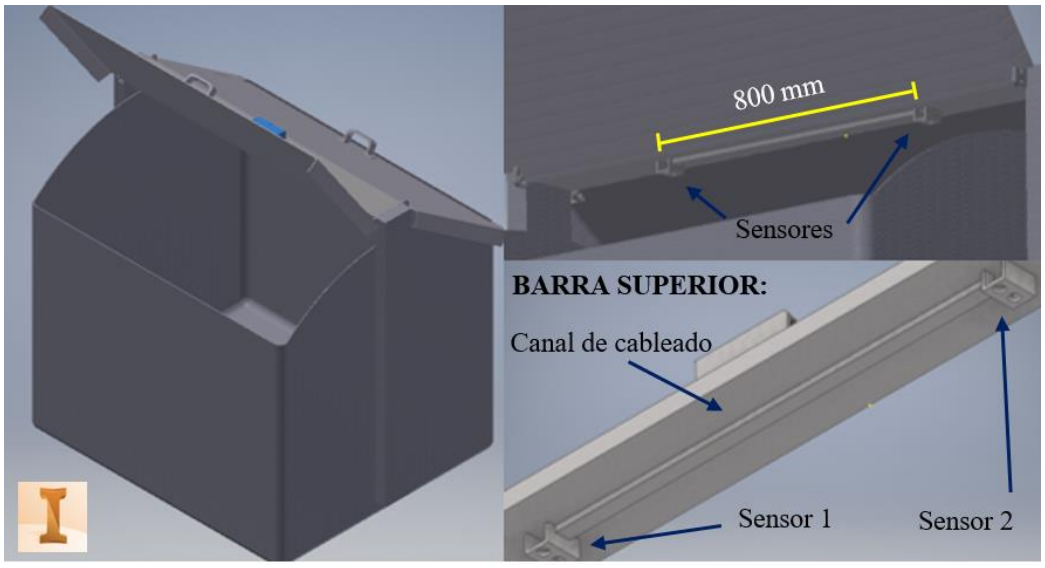


Figura 34. Ubicación de sensores en contenedor municipal

Para esta propuesta, se decidió emplear 2 sensores HC-sr04 debido al extenso largo del contenedor, y la limitación de medida que tienen los sensores en linea de vista que es de 15°. (tabla 6) Empleando 2 sensores ubicados como se muestra en la figura 34, se puede obtener un promedio del nivel de llenado que capte cada uno y tener un estimado del llenado total que lleve el contenedor.

Como se observa en las figuras 33 y 34, la única alteración al contenedor se daría a nivel de la barra superior, siendo este el único elemento que sufriría cambios. En el Anexo G se encuentran los planos para la propuesta de cambio en la barra superior, para observar más detalladamente cómo sería el diseño.

Experimento 1

Objetivo:

- Verificar que el envío de tramas por modo API desde ambos routers hasta el coordinador sea correcto y que los datos de las lecturas obtenidas por los sensores no se ven distorsionados, empleando el monitor serial del programa Arduino IDE.

DOE:

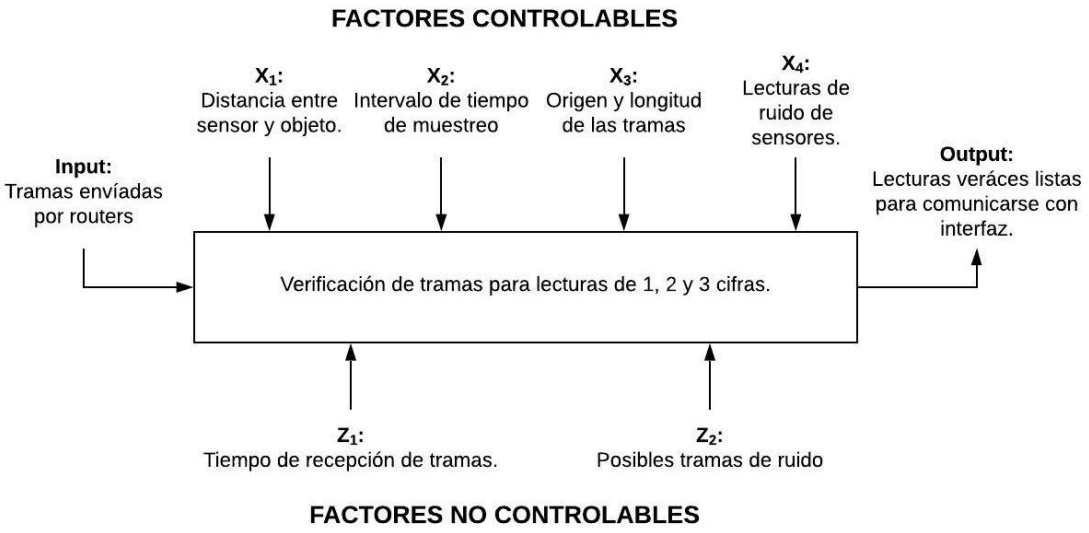


Figura 35. DOE para experimento 1

Variable principal:

- Lectura en distancia de los sensores

Materiales:

1. 2 Routers API (tabla 8) con circuito y cuerpo armado (figura 48)
2. 1 Xbee Coordinador API (figura 47 y tabla 7)
3. 1 cable de datos USB
4. 1 cinta métrica de 1,5m
5. Programa Arduino IDE

Proceso:

Para iniciar este experimento, tanto el coordinador como los routers, ya deben encontrarse construidos y programados según como se detalló a lo largo de la metodología en este trabajo. En resumen:

1. Se debe configurar los módulos empleando el programa XCTU para delegar a cada dispositivo el papel de coordinador y routers. Para el coordinador se programa al módulo según los parámetros presentados en la tabla 7, y para los routers, se programa a los módulos según los parámetros presentados en la tabla 8.
2. Después, se efectúan el diseño de software y hardware paralelamente, ya que comparten muchas características que los complementan y aceleran el proceso de diseño.
3. En software, se emplearon los códigos de los anexos A y B, alterando el código del coordinador únicamente para que imp rima las tramas y lecturas en vez de escribirlas en el serial.
4. Para eliminar lecturas de ruido en los sensores, se toma 10 muestras y se las promedia, aumentando la precisión en las medidas de los sensores. (Anexo A)
5. Finalmente, se construye el cuerpo de los routers (figura 48), de forma que el circuito no se encuentre vulnerable a posibles factores externos que lo afecten.

Una vez que se tienen los routers y coordinador preparados, se prepara un ambiente controlado (figura 36) para tomar las lecturas de cada sensor. Se emplea una cinta métrica que permita verificar la veracidad de las lecturas en las distintas posiciones que se colocarán a los routers.

Figura 36. Ambiente controlado para toma de lecturas

Posteriormente, se coloca a los routers en las 3 posiciones que se emplearán para la toma de medidas (tabla 12). Se está tomando medidas en estas tres posiciones ya que cada una es a una distancia con diferente número de cifras que las demás, lo que permitirá verificar una buena recepción de las diferentes tramas que se envían de acuerdo con el número de cifras de las lecturas del sensor, como se explicó en la tabla 9.

Tabla 12

Posiciones para testeo de routers

	Distancia router-objeto [cm]		**Número de cifras de las distancias**
	Router 1	**Router 2**	
1era posición	20	25	2
2da posición	120	120	3
3era posición	8	8	1

Figura 37. 1era posición de lecturas, 25cm y 20cm

Figura 38. 2da posición de lectura, 120cm

Figura 39. 3ra posición de lecturas, 8cm

Experimento 2

Objetivo:

- Verificar que la topología programada para los módulos funciona correctamente y permite un aumento en el rango de comunicación entre los nodos más distantes.

DOE:

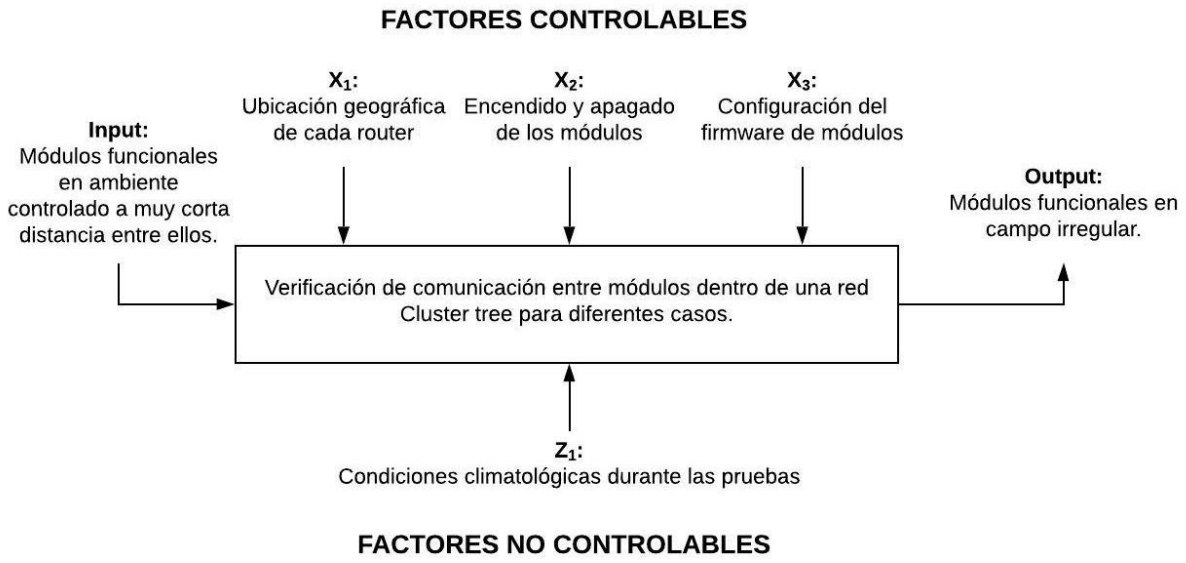


Figura 40. DOE para experimento 2

Variable principal:

- Funcionamiento o caída de algún router.
    - 1er caso: ambos routers funcionales
    - 2do caso: cae el router 1.
    - 3er caso: cae el router 2.

Materiales:

1. 2 Xbee S2c Routers API (table 8)
2. 1 Xbee S2 Coordinador API (tabla 7)
3. 1 adaptador Xbee Explorer USB.

4. Alimentación para los routers. (en este experimento se empleó la fuente del circuito completo de los routers)
5. Programa XCTU

Proceso:

En este experimento se verificó si hay o no comunicación entren los diferentes nodos de la red sometiéndolos a diferentes casos que podrían darse. El programa XCTU permite observar el estatus de la red en tiempo real, es decir, permite observar gráficamente si todos los nodos se encuentran conectados o no a la red.

Para iniciar, se llevó a los módulos al exterior y se los ubicó de forma que únicamente el router 1 se encuentre en el rango de comunicación de 40 metros con obstáculos (tabla 3) del coordinador, como se observa en la figura 41. El router 2 logra comunicarse con el router 1 dado que se encuentran viéndose uno al otro, es decir, sin obstáculos, y estando así el rango aumenta a 1,2km.

Figura 41. Ubicación geográfica de los módulos

Experimento 3

Objetivo:

- Verificar la funcionalidad del sistema monitoreando el nivel de llenado de 2 contenedores de basura de distintas dimensiones, en la interfaz creada mediante Processing.

DOE:

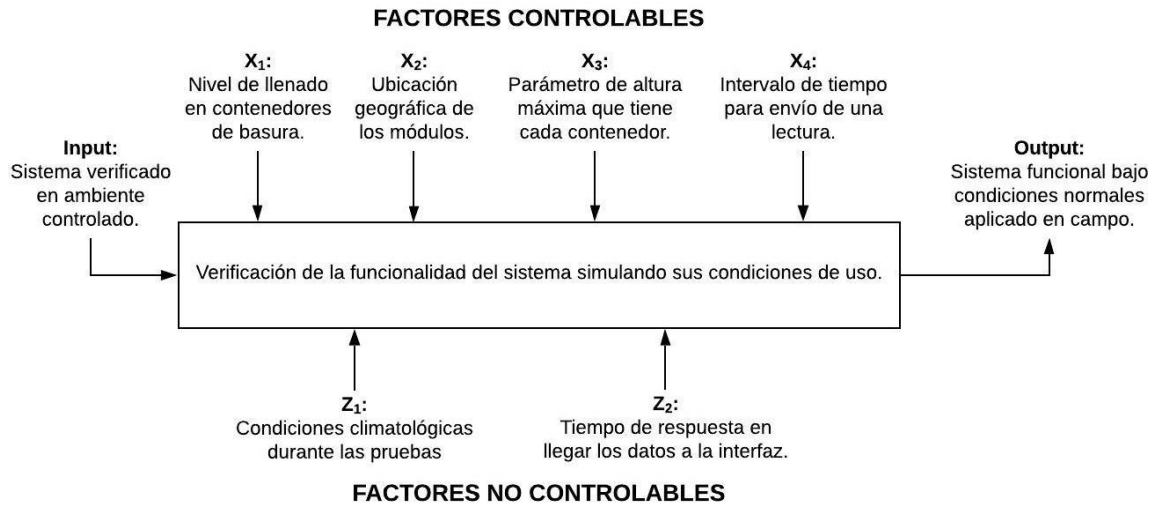


Figura 42. DOE para experimento 3

Variable principal:

- Nivel de basura en 2 contenedores distintos.

Materiales:

1. 2 Routers API elaborados como en la figura 48 y tabla 8
2. 1 Xbee programado para Coordinador API (tabla 7)
3. 1 adaptador USB explorer para Xbee
4. 2 contenedores de basura distintos
5. Algunas fundas de basura o que simulen serlo
6. Programa Processing (versión 3.5.3.)

Proceso:

Este experimento parte del anterior, se emplean las mismas ubicaciones ilustradas en la figura 41, integrando contenedores de basura a los prototipos generados de los routers. A cada contenedor se le instaló provisionalmente una antena en la parte interna de su tapa, como se muestra en la figura 43.

Figura 43. Ubicación de prototipos en contenedores de basura

Posteriormente, se configuró a los algoritmos del coordinador y de la interfaz gráfica con las alturas internas de cada contenedor, es decir, la distancia que hay entre el sensor hasta la superficie inferior de cada contenedor. Se realiza esto para que el porcentaje de llenado en la interfaz sea acorde con las diferentes alturas de los contenedores, pues el 20% del contenedor 1 no es el mismo que el 20% del contenedor 2 por ejemplo. La altura del primer contenedor fue de 45cm, y para el segundo de 30cm. Esto se puede observar en las variables sombradas de amarillo en la figura 44.

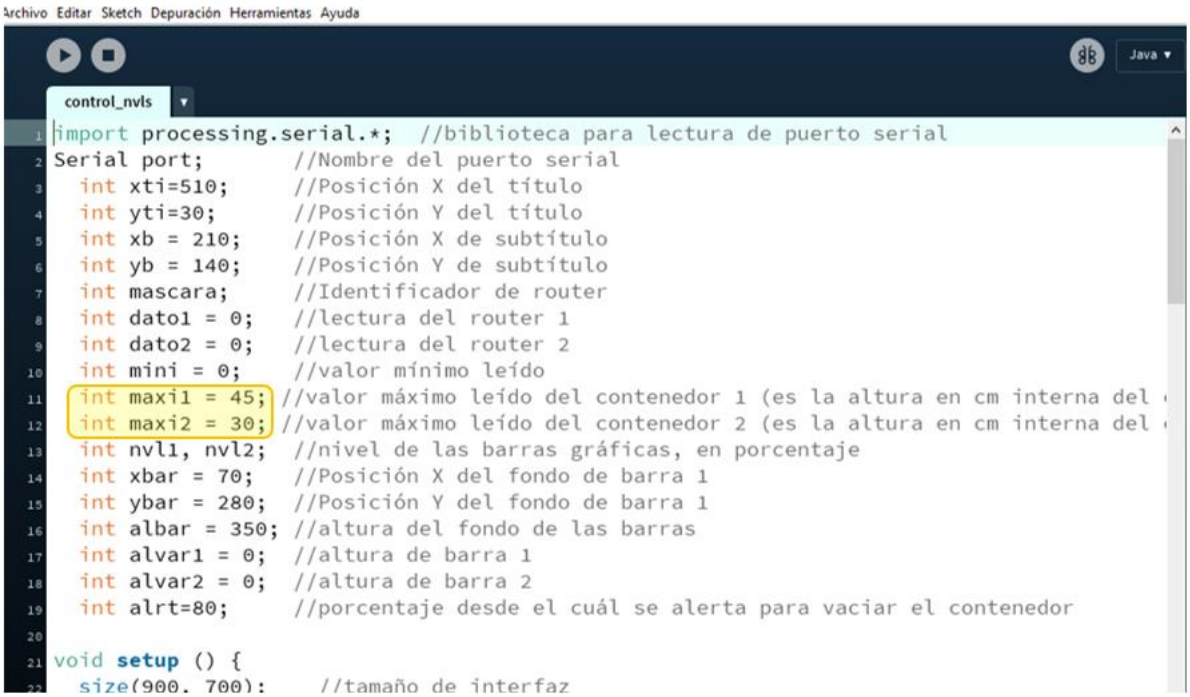


Figura 44. Listado de variables en código de interfaz, Processing

También se alteró la interfaz gráfica para que las imágenes del tipo de contenedor que cada router está interviniendo sean las respectivas con los contenedores que se está usando en estas pruebas y no los municipales como en la figura 27. Como se mencionó al diseñar el software, esta característica es sencilla de reprogramar en dependencia del tipo de contenedor que se esté monitoreando.

Ahora bien, se procede a iniciar las pruebas. Para la primera prueba, se toma lecturas con los contenedores vacíos para determinar si las alturas seteadas en los algoritmos son las correctas al observar que la interfaz marca 0% en ambos contenedores.

Figura 45. Contenedores vacíos para 1era prueba

Posteriormente, en la segunda prueba se procede a depositar cierta cantidad de basura en cada contenedor, otorgando un aproximado visual del 50% para el contenedor 1 y un 35% para el contenedor 2.

Figura 46. Contenedores con niveles medios de llenado

Finalmente, para probar el sistema de alerta programado en el algoritmo de la interfaz (Anexo C), se procede a aumentar el nivel de llenado del contenedor 1 a un aproximado superior al 80%.

Resultados

Una vez que se realizó las operaciones e ingresó los códigos generados en la sección de “Software”, junto con los diseños de circuitos presentados en la sección de “Hardware” en la realidad, se obtuvieron los siguientes resultados:

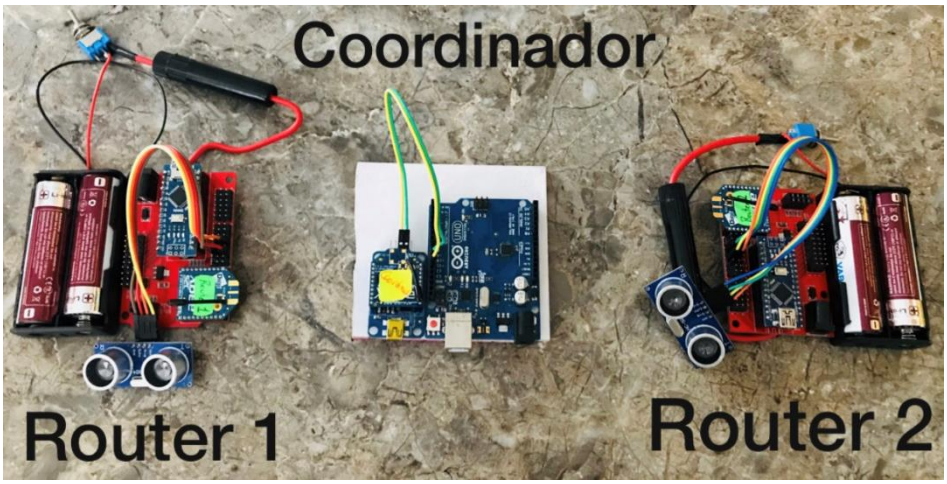


Figura 47. Circuitos de Routers y Coordinador

Posterior a la obtención de los circuitos en su totalidad y habiendo demostrado que funcionan sin la necesidad de otros elementos electrónicos, se procedió a la fabricación del cuerpo de las antenas obteniendo el siguiente resultado:

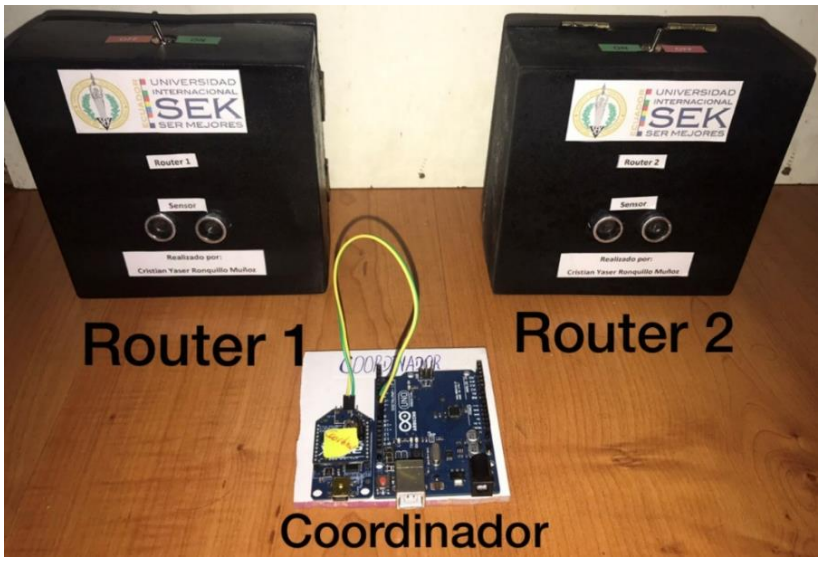


Figura 48. Prototipo de Routers y Coordinador listos

Experimento 1

Empleando el monitor serial del programa Arduino IDE configurado a una velocidad de 9600 baudios, se consiguió verificar los resultados del primer experimento para los 3 casos.

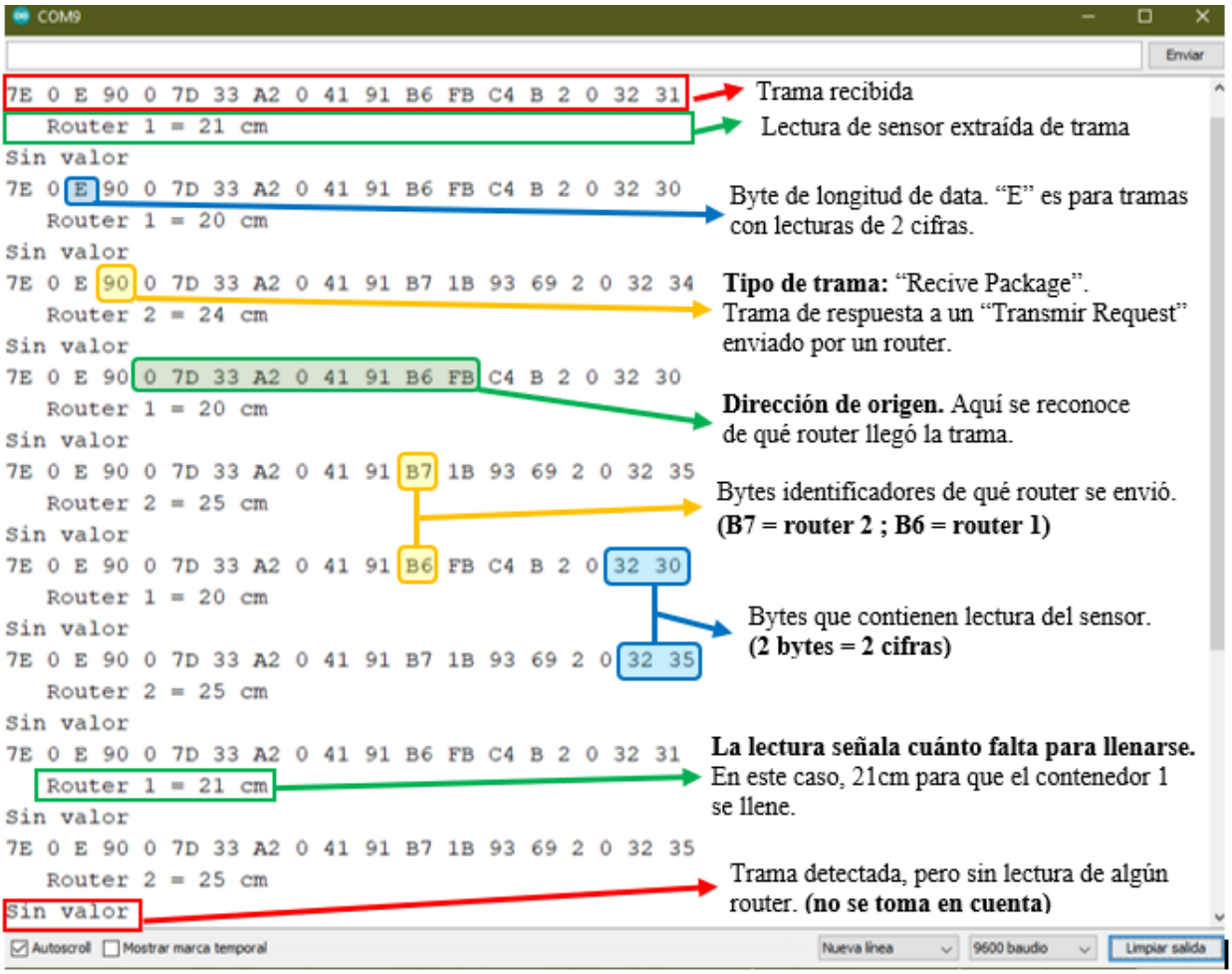


Figura 49. Tramas para 1era posición, lecturas de 2 cifras.

En la figura 49 se observa la impresión de las tramas recibidas por el coordinador para la primera posición del experimento, y debajo de ellas se encuentra la lectura del sensor del router que la envió. Cabe aclarar que en las tramas impresas no se imprime el último byte, que es el de la sumatoria (tabla 9), debido a que no se lo consideró importante a analizar. A continuación, se mostrarán las recepciones para las demás posiciones de los

routers detallando únicamente los bytes que cambian en comparación con las tramas de la figura 49.

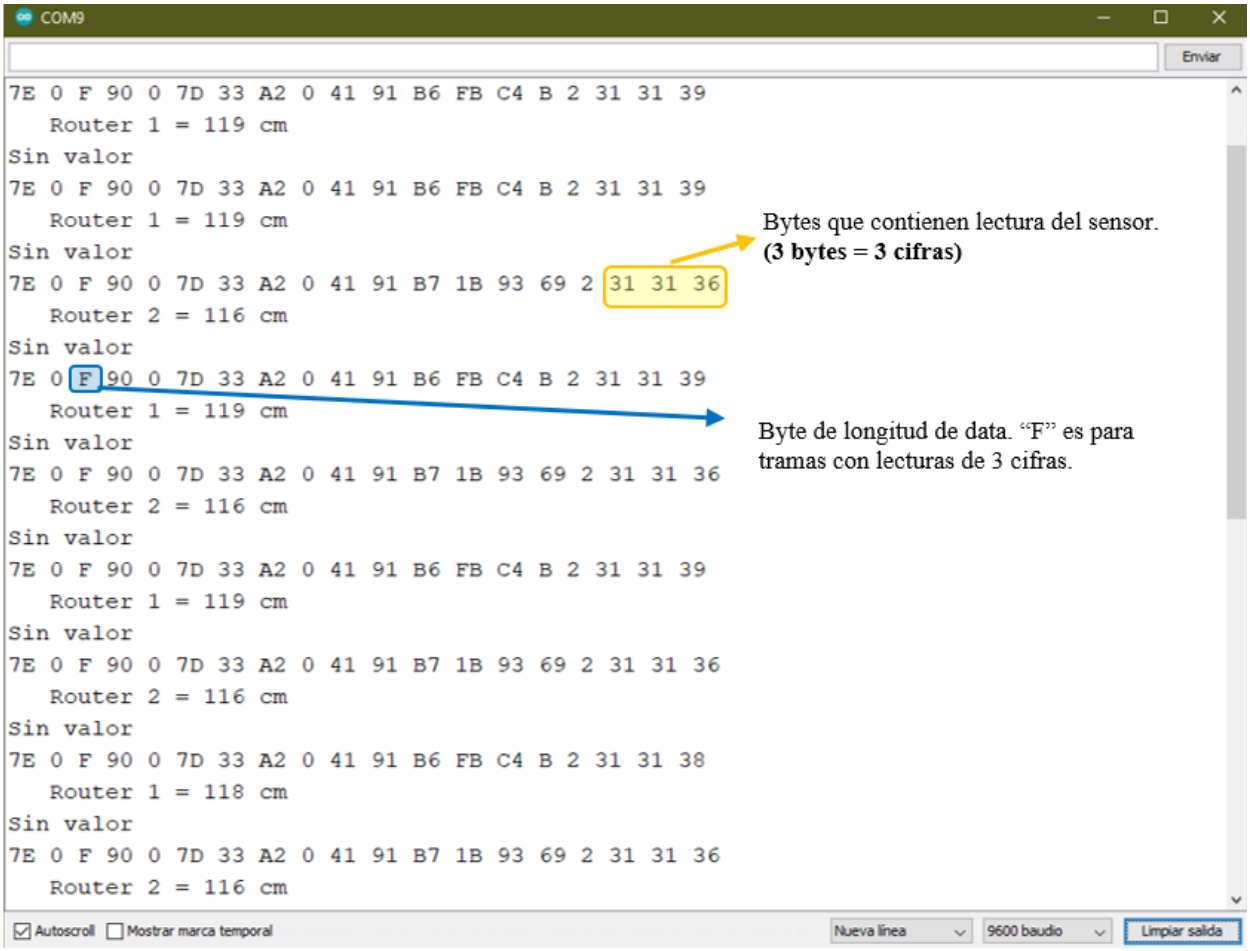


Figura 50. Tramas para 2da posición, lecturas de 3 cifras

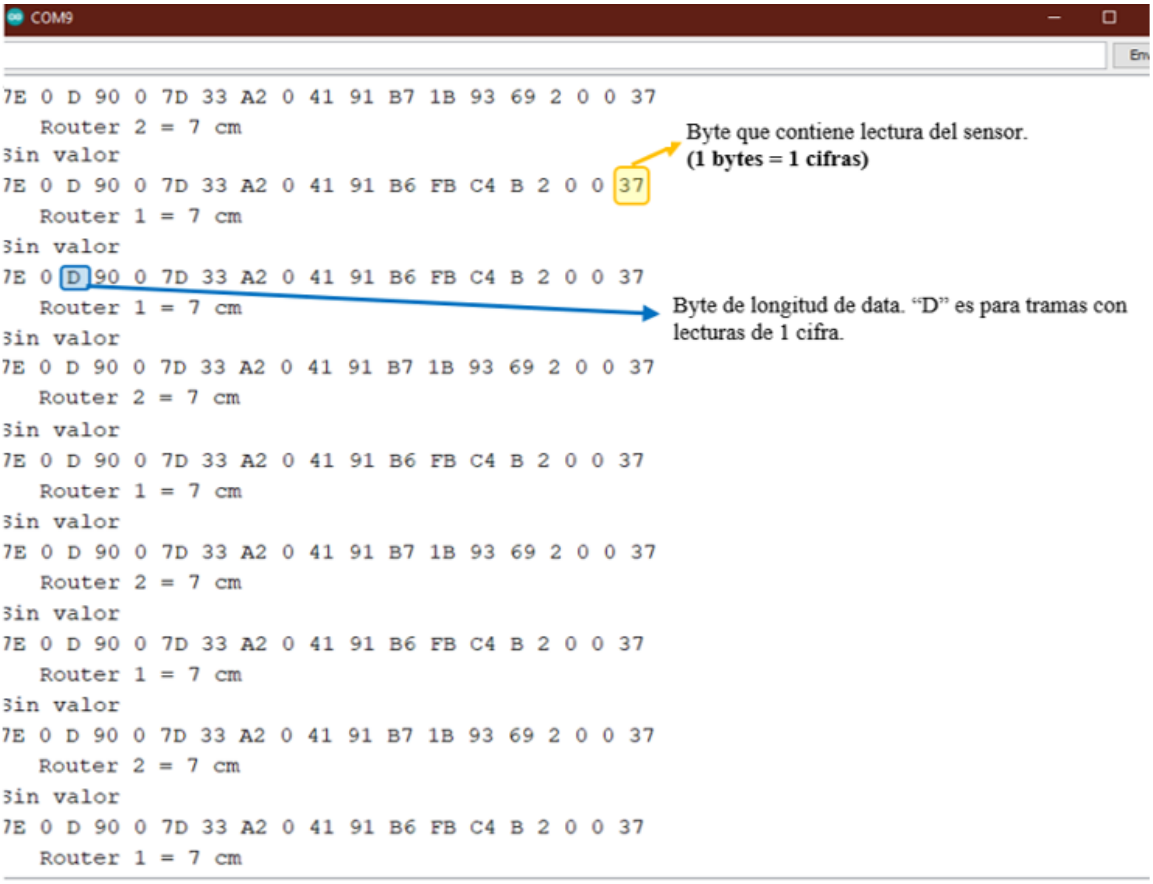


Figura 51. Tramas para 3ra posición, lecturas de 1 cifra

Experimento 2

Para el primer caso, ubicando el router 1 a una distancia de 29,3m del coordinador, y al router 2 a 37,6m del router 1, se consiguió el siguiente diagrama de red con todos los nodos activos:

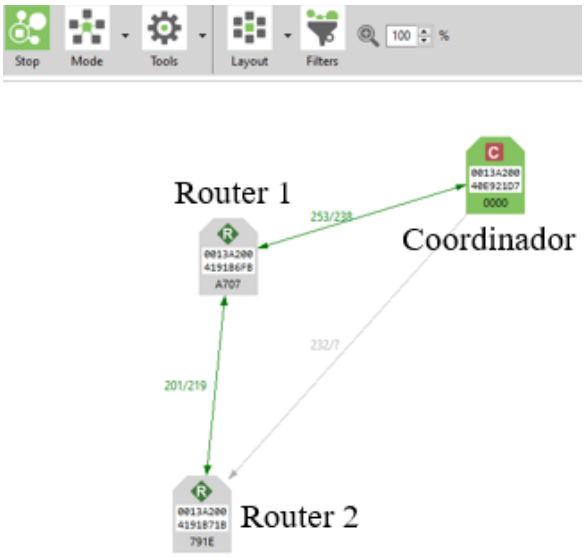


Figura 52. Prueba de topología, 1er caso

Para el segundo caso, en el que se apagó únicamente al router 1, manteniendo sus ubicaciones anteriores, se obtuvo el siguiente diagrama de red:

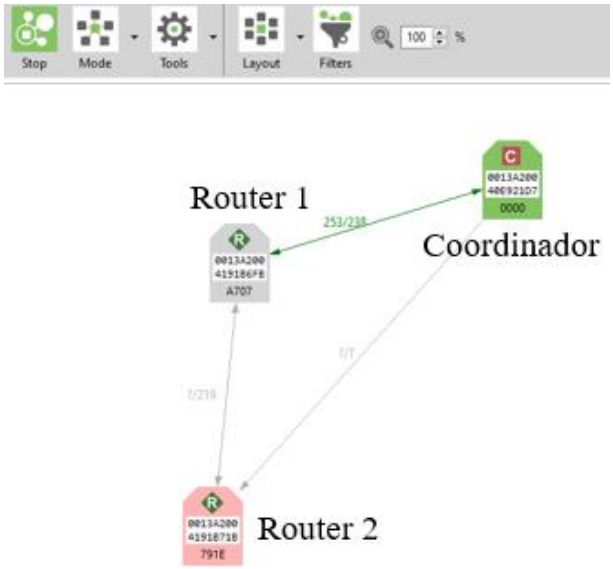


Figura 53. Prueba de topología, 2do caso

Finalmente, para el tercer caso se apagó únicamente al router 2 y se obtuvo el siguiente diagrama de red:

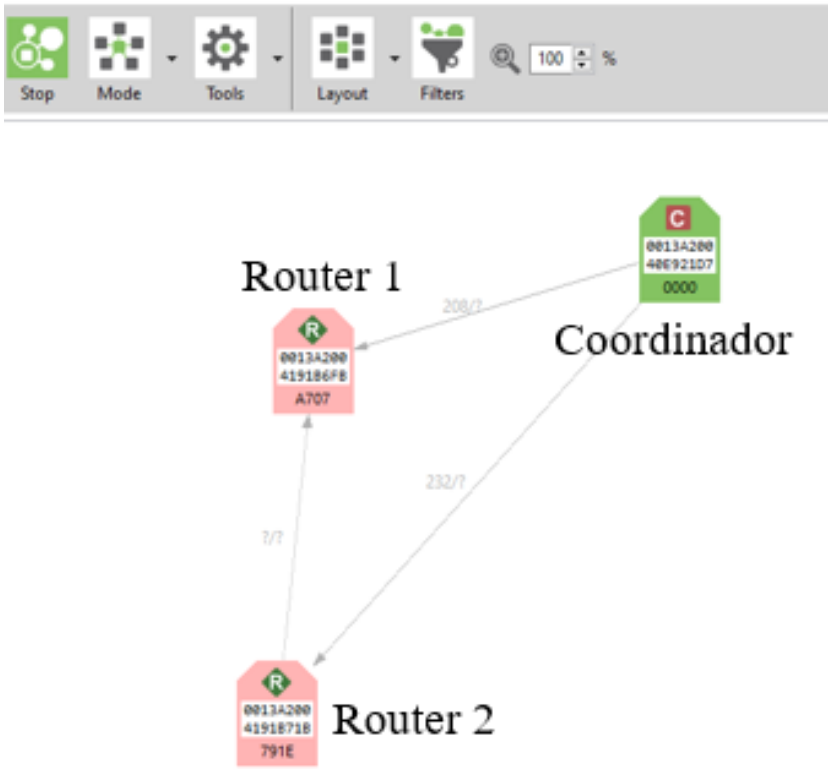


Figura 54. Prueba de topología, 3er caso

Se observa que todos los routers de la red han caído, es decir, que no se recibe señal de ninguno al coordinador.

Experimento 3

A lo largo de la primera prueba, en la cual los contenedores se encontraban vacíos, se obtuvo el siguiente resultado reflejado en la interfaz de usuario:

Figura 55. Resultados en interfaz para prueba 1 del sistema en campo

En la figura 55 se puede observar que los routers en los contenedores señalan que éstos se encuentran al 0% de su capacidad total, debido a que se encuentran vacíos. Cabe recalcar que al momento de iniciar el sistema, demoró un aproximado de 10 segundos en obtener los datos de los routers.

En el segundo caso, depositando cierta cantidad de basura en los contenedores, que visualmente se aproxima a que son 50% y 35% para los routers 1 y 2 respectivamente, se obtuvo los siguientes resultados:

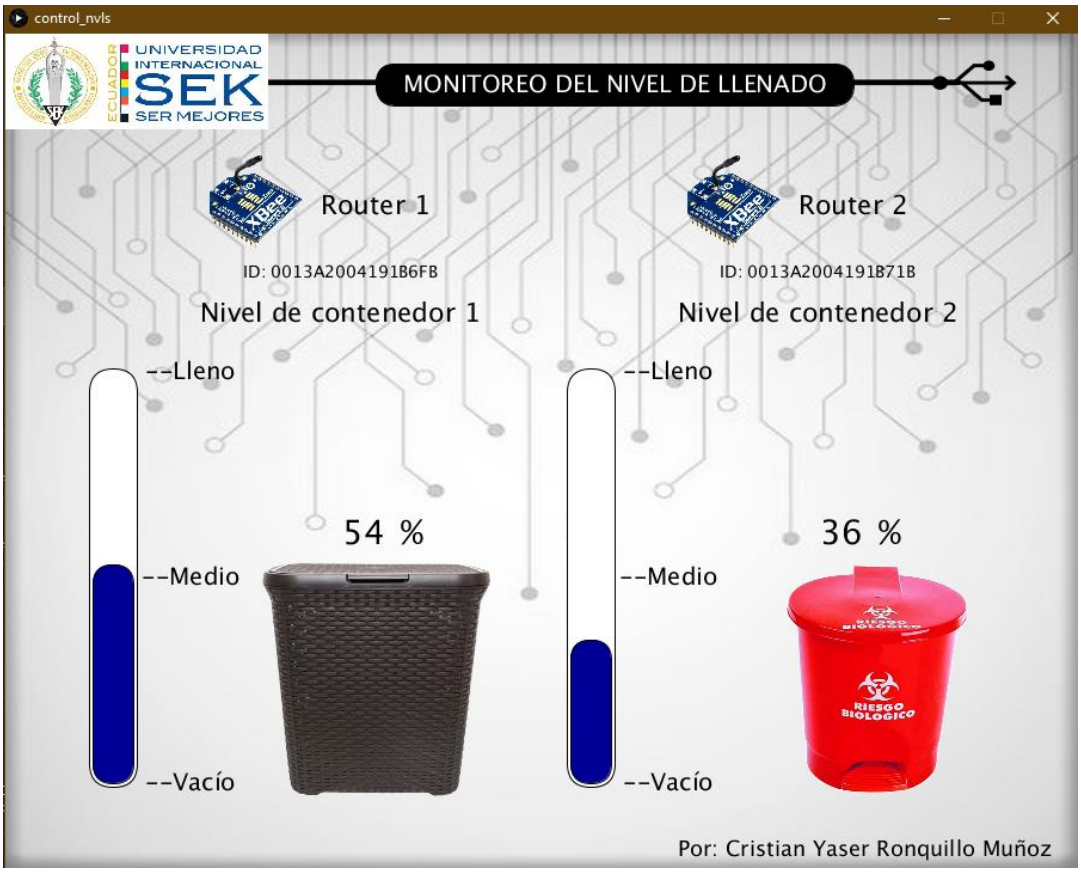


Figura 56. Resultados en interfaz para prueba 2 del sistema en campo

En la figura 56 se muestran los porcentajes de llenado detectados por los sensores de cada router. El tiempo aproximado en obtención de los datos para esta prueba fue de 10 segundos entre el cambio físico en el nivel de llenado, hasta que los datos llegaron a la interfaz.

La siguiente y última prueba en este experimento, se la realizó únicamente influyendo sobre el contenedor 1, agregando un poco más de basura para aumentar su nivel

y a su vez permita verificar la alerta para vaciarlo en la interfaz. Los resultados fueron los siguientes:

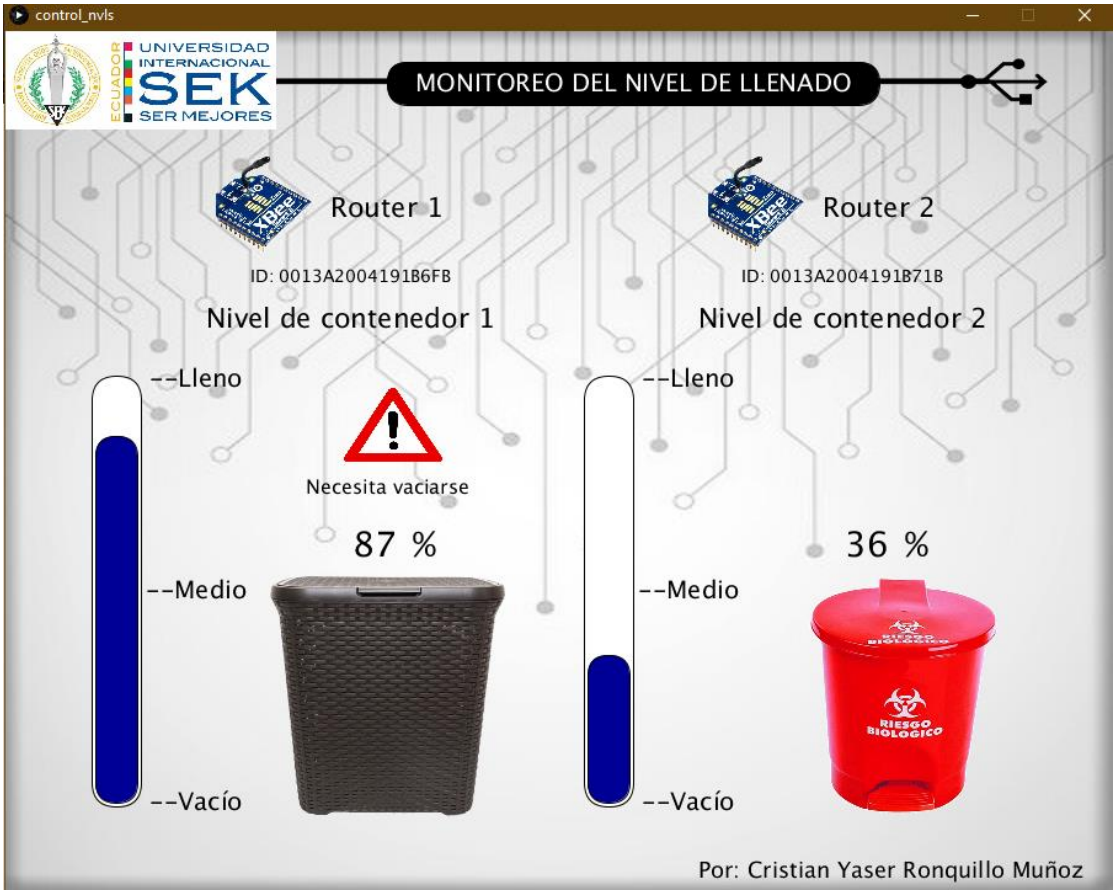


Figura 57. Resultados en interfaz para prueba 3 del sistema en campo

En la figura 57 se observa cómo al aumentar en más del 80% al nivel de llenado en el contenedor 1, se visualizó una alerta por encima de la cifra, señalando que éste debe vaciarse.

Discusión de Resultados

Dentro de la etapa de diseño para los routers, se consideró no emplear un Arduino y que las lecturas entregadas por el sensor vayan directamente al Xbee. Los módulos Xbee tienen la capacidad de programar su firmware mediante el programa XCTU para que sus pines digitales sean entradas o salidas, recibiendo así señales de sensores o dando órdenes para actuadores. Sin embargo, no se consiguió realizar esto debido a que los sensores HC-sr04 funcionan de forma que es necesaria una codificación específica para obtener las distancias detectadas en sus lecturas, y esta codificación se la tuvo que hacer mediante un Arduino.

Los circuitos para los routers construidos son exactamente iguales. Lo único que los diferencia es el ID de los módulos Xbee que emplean, siendo esta la forma de reconocer cada router. En la figura 47 se observa el resultado del circuito para el coordinador. Si se lo compara con el diseño en la figura 28, se puede observar que la alimentación del Xbee no está conectada al Arduino UNO, sino que se la conecto a un adaptador USB Explorer para Xbee que permite conectar el módulo a la computadora. Se realizó de esta forma debido a que no es posible identificar el Xbee en XCTU a través del Arduino, solamente es posible empleando un USB Explorer.

La fabricación para el cuerpo de los routers de realizó de acuerdo a los planos presentados en el anexo E. Cada elemento del circuito entro perfectamente como se observa en el anexo F. Los elementos fueron fijados empleando silicona sobre la madera. Los sensores fueron el único elemento que no se fijó con el objetivo de poder extraerlo con facilidad si existía algún fallo en la etapa de pruebas.

Los resultados experimento 1 resultaron satisfactorios, pues se consiguió imprimir las tramas que recibía el coordinador en el monitor serial del Arduino UNO. En la figura 49 se observa detalladamente los bytes de mayor importancia que se tomó en cuenta para la extracción de las lecturas en las tramas. En las pruebas iniciales del experimento se leía tramas de ruido, es decir, sin ningún contenido de lecturas. A todas las tramas de ruido se las filtró empleando funciones “if” en el código del coordinador (Anexo B). Por ejemplo, solamente se captaba tramas con 23 bytes o más, y tramas que inicien con el byte 7E. Dentro de estas pruebas también se observó que los sensores enviaban algunas lecturas fallidas, de forma que se aplicó un sistema de muestreo para que los sensores tomen 20 lecturas y las promedien para obtener datos más precisos. Para esto se empleó una función “for” en el código del coordinador, dentro de la función encargada de tomar lecturas de los sensores llamada “lectura()”.

En el experimento 2 se consiguió captar la estructura de la red empleada en tiempo real. Para el primer caso (figura 52) se verificó que existía una comunicación estable entre los dos routers y entre el router 1 con el coordinador. Dado que el router 2 se encontraba fuera del rango de comunicación con el coordinador (figura 41), no hubo una comunicación directa entre ambos. Sin embargo, se comprobó la eficiencia de la red ya que el coordinador se apoyó en el router 1 para recibir las tramas del router 2. Una red con gran variedad de routers incrementa en gran manera el rango de comunicación entre ellos y con el coordinador. En el segundo caso (figura 53) seguía existiendo una comunicación estable con el router 1, incluso si el router 2 se apagaba. Posteriormente, en el tercer caso (figura 54), no se obtuvo lectura de ningún router, aunque el router 2 seguía encendido.

Obteniendo este resultado se comprobó la importancia del router 1 en esta red, y así, el funcionamiento de la topología Cluster Tree.

Finalmente, en el experimento 3 los resultados fueron satisfactorios al observar que en cada caso al que se sometió al sistema, éste respondía muy bien. Se llegó a esta conclusión dado que el tiempo que demoró entre el cambio físico en el nivel de llenado y la obtención de los datos en la interfaz, fue de aproximadamente 10 segundos. En la tercera prueba, al depositar lo que el sistema detectó como el 87% del nivel de llenado en el contenedor 1, la alerta para vaciar este contenedor saltó inmediatamente se obtuvo la lectura. Cabe recalcar que se debe especificar en qué COM se encuentra conectado el Arduino del coordinador, para así configurarlo en el programa de Processing para la interfaz. En este trabajo se empleó el COM9 para la lectura del serial. El programa de Arduino IDE permite detectar en qué COM se encuentra conectado el Arduino.

Conclusiones

Observando los resultados de los experimentos, se concluye que la topología de Cluster Tree permitió un correcto funcionamiento del sistema en este trabajo, tomando en cuenta que se consiguió comunicar al coordinador de la red con el router 2, estando a una distancia fuera del rango de alcance. (figura 41) Sin embargo, este sistema apunta a una aplicación empleando una red mucho más grande que la testeada, una red capaz de llegar a un máximo de 65 535 nodos (tabla 1). De esta forma, se aplicaría una topología del tipo Mesh (figura 7) otorgando mucha más robustez al sistema. Aplicando esta topología, los routers serían independientes en la red, en el sentido de que si alguno deja de funcionar, la red nunca caería como sucedió en la figura 54.

El diseño del prototipo para los routers permitió la correcta instalación de los elementos electrónicos dentro de éste, protegiéndolos de factores externos que puedan afectar el circuito, además de un mejor manejo del interruptor y el sensor. No se contempló la posibilidad de una fijación permanente en los contenedores mediante este primer prototipo, pues para las pruebas que se planearon realizar era necesario retirarlo varias veces. Por este motivo, su instalación en los contenedores de prueba fue provisional.

La propuesta para el rediseño de los contenedores municipales presentada en las figuras 33 y 34 se basó en el diseño del circuito empleado en este trabajo (figura 30), con todos sus elementos. Únicamente, se propone emplear otro sensor debido al tamaño que tienen los contenedores municipales en Quito.

El diseño de la interfaz gráfica consiguió cumplir con lo esperado, pues mediante ésta se consiguió observar el nivel de llenado de los contenedores testeados en este trabajo, de una forma muy amigable con el usuario. Esta interfaz permitió observar el tipo de

contenedor sobre el que cada router se instaló de forma gráfica, además de la ID de cada módulo en la red.

Adicionalmente, se evidenció cómo la interfaz lanzaba una alerta para vaciar determinado contenedor, lo que podría ayudar a los gestores de la red a programar las rutas óptimas para sus camiones recolectores, y así se eviten rutas en las que haya contenedores casi vacíos. Esto reduciría los tramos de las rutas, reduciendo también el nivel de contaminación por parte de las flotas recolectoras. También, este sistema de alerta permite detectar qué contenedores a lo largo de la red se encuentran próximos a llenarse. De esta manera, se reduciría en gran medida la presencia de contenedores que desborden basura en las ciudades, mejorando la calidad de vida de los moradores que viven cerca de éstos focos de concentración de basura, y la imagen que se tiene como ciudad.

En conclusión, se consiguió diseñar un sistema que detecte el nivel de llenado en contenedores de basura mediante comunicación inalámbrica bajo el protocolo Zigbee. Cabe recalcar que este diseño está sujeto a una continua mejora, aplicando nuevos métodos que permitan un ahorro de energía en los routers, efectuar más testeos con una mayor cantidad de módulos Xbee formando una red del tipo Mesh, etc. La importancia de la aplicación de este sistema en ciudades recae en sus ventajas de ahorro en recursos humanos, económicos y ambientales, además de que forma parte del proceso de mejora formando a las ciudades como ciudades inteligentes. Habría un ahorro en recursos humanos y económicos reduciendo la cantidad de camiones recolectores circulando por la ciudad. En el ámbito de lo ambiental, se reducirían las emisiones de CO_2 por parte de los camiones y la aglomeración de basura en varios puntos de las ciudades.

Recomendaciones

1. Verificar la funcionalidad de los módulos Xbee mediante el programa XCTU previo a cualquier tipo de aplicación que se les vaya a dar. Es importante programar su firmware en base al diseño conceptual de red que se desee, y a las actividades que vayan a realizar en cada nodo.
2. Adquirir un adaptador USB Explorer para Xbee, ya que permite la conexión del módulo con la computadora. Si se conecta el módulo a un Arduino, la computadora no lo reconocerá.
3. Si un módulo Xbee está conectado a los pines TX y RX de un Arduino, no se podrá subir ningún código al Arduino, debido a que su entrada de datos serial está ocupada por el módulo Xbee. Se debe desconectar a la antena, subir el código al Arduino, y volver a conectar el módulo.
4. Siempre colocar un fusible entre la fuente y el positivo del circuito para proteger a los elementos de un posible cortocircuito. En este trabajo se utilizó uno de 500mA dado que la corriente máxima que puede entregar un Arduino NANO a sus pines es de 500mA, sobre ese valor se quemaría.
5. Si se va a trabajar en modo API, se debe determinar la estructura de la trama que se desea enviar y recibir. Para lograr esto se debe emplear el programa XCTU y en su monitor crear un tipo de trama o “frame” con las características que se desee. El programa genera automáticamente la trama que se enviará. Una vez que se envíe la trama, se puede observar la trama recibida en el otro módulo igualmente por el monitor de XCTU.

Trabajos Futuros

1ro: Dispositivos Finales sobre Routers.

Este trabajo se ha realizado en base a que cada contenedor tendrá un router que le permita comunicarse con otros contenedores, y a su vez, con el coordinador de red. Se lo realizó así debido a que actualmente la ciudad de Quito, en donde se realizaron las pruebas, no se cuenta con routers que activen la red, y sin ellos es imposible generar una red de topología "Cluster Tree" como se desearía.

Sin embargo, la meta óptima para la aplicación de este sistema en ciudades es que el dispositivo de cada contenedor sea un "Dispositivo final (End Device)", para que así se aproveche la ventaja de este tipo de nodo que es el modo "Sleep" o dormido del módulo. De esta manera, el Xbee únicamente consumiría corriente cuando sea necesario enviar datos o recibirlos, y no tendría que estar activo todo el tiempo. Aplicando esto, sería imperativo contar con routers en la red, que podrían estar ubicados en postes de alumbrado público, y de esta manera no se tomaría en cuenta su consumo, pues se alimentarían directamente del poste.

Si hubiese dispositivos finales en los contenedores, la duración de las baterías sería la siguiente:

$$C.de\ corriente\ por\ hora\ [mAh] = Consumo\ \ ON + Consumo\ \ OFF \quad (12)$$

$$Consumo\ \ ON = \ (22mA + 15mA + 15mA) * \frac{1}{120} h = 0{,}43\ mAh \quad (13)$$

Cabe destacar que el Arduino NANO es el único elemento que permanecería encendido en modo OFF del circuito, consumiendo 15mA como se observa en la tabla 5.

$$Consumo\ OFF = 15mA * \frac{119}{120} h = 14{,}87\ mAh \quad (14)$$

$$C.de\ corriente\ por\ hora\ [mAh] = 0{,}43mAh + 14{,}87mAh = 15{,}3mAh \quad (15)$$

$$Duración\ de\ baterías\ [días] = \frac{Carga\ de\ batería\ [mAh]}{Consumo\ de\ corriente\ por\ hora\ [mAh]} \over \frac{}{\#\ horas\ diarias\ en\ funcionamiento\ [h/día]} \quad (16)$$

$$Duración\ de\ baterías\ [días] = \frac{\frac{8800\ [mAh]}{15{,}3\ [mAh]}}{24\ [h/día]} \quad (17)$$

$$Duración\ de\ baterías\ [días] = 24\ días \quad (18)$$

Como se observa en la ecuación 18, el tiempo de duración de las baterías empleando dispositivos finales que se duerman en modo OFF, es de 24 días. Este cambio en la red aumentaría el rendimiento de las baterías en un 262,3% frente al rendimiento de 9,15 días (ecuación 11) empleando routers en contenedores.

2do: Arduino en Modo "Sleep".

Otra forma efectiva para aumentar la duración de baterías es apagando al Arduino en los lapsos de tiempo que no se toman lecturas, modo OFF. De esta manera, sólo se consumiría energía en modo ON siendo el módulo del contenedor un dispositivo final.

3ro: Panel Solar para Carga de Baterías.

El sistema de alimentación en los routers puede ser mejorando de gran manera empleando un panel solar que cargue continuamente las baterías durante el día. De esta forma, su duración aumentaría. En las figuras 58 y 59 se puede observar cómo variaría el diagrama de circuito aplicando una celda fotovoltaica o panel solar.

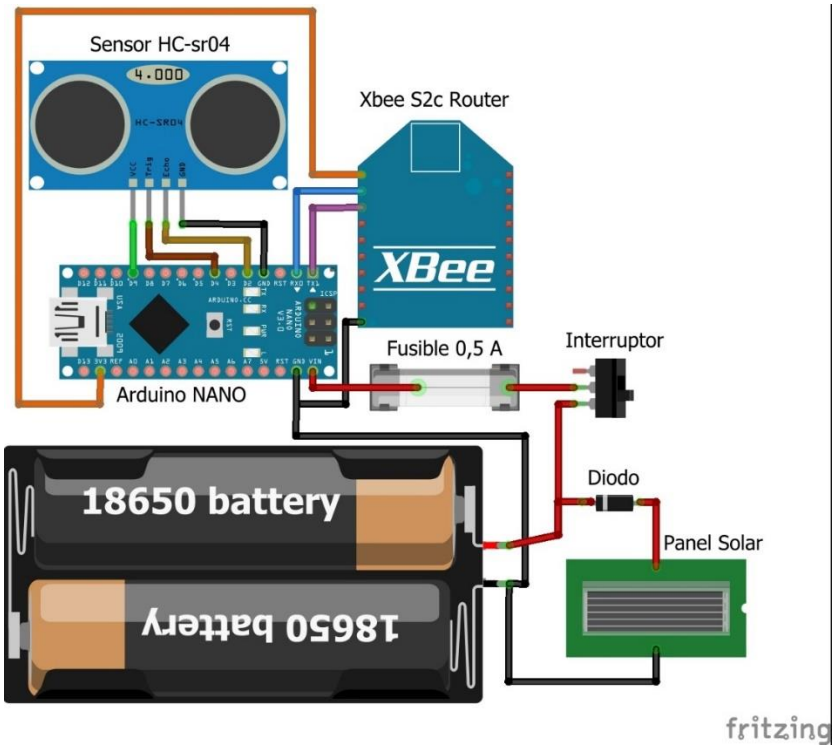


Figura 58. Circuito de Routers con panel solar

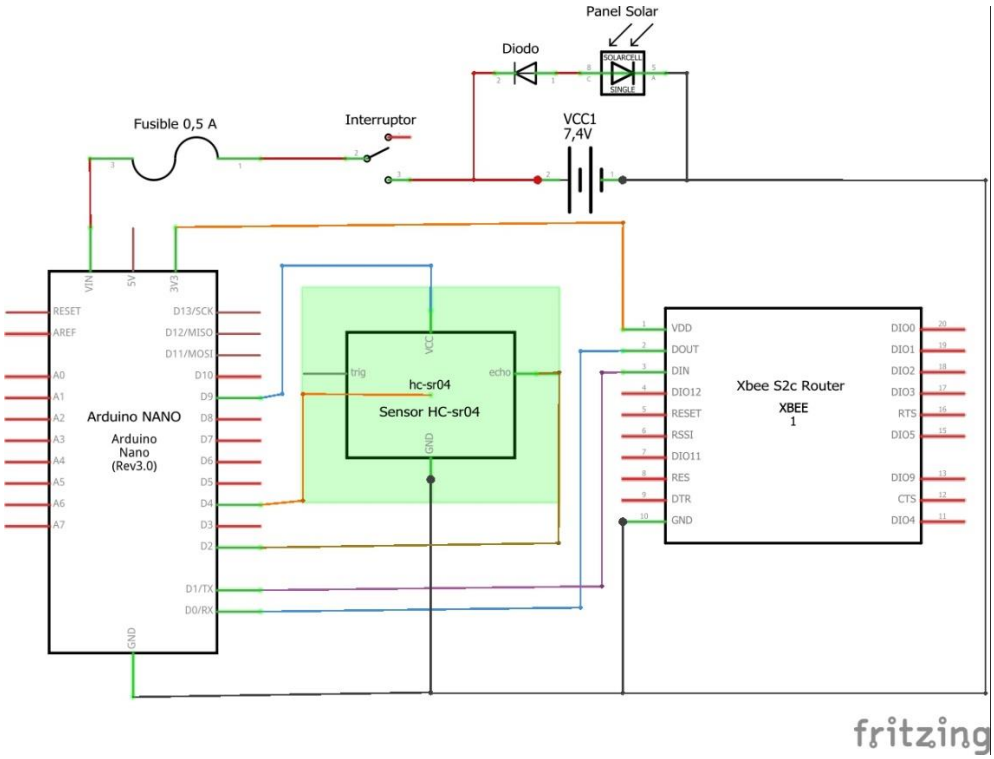


Figura 59. Diagrama esquemático de circuito para Routers con panel solar

4to: Circuito sin Arduino.

Otra forma de ahorrar energía podría ser empleando un circuito sin Arduino y que las lecturas del sensor caigan directamente con el módulo Xbee. Esta opción se contempló al diseñar el circuito de los routers, sin embargo no se la ejecutó ya que los sensores empleados (HC-sr04) necesitan que su lectura sea procesada por un microcontrolador para interpretarla como distancia. (Hernández, 2016)

Si se consigue desarrollar una biblioteca o actualización del firmware de los módulos Xbee para que procesen las lecturas que captan los sensores ultrasónicos HC-sr04, se conseguiría obviar el uso de Arduino en el circuito y en consecuencia reducir el consumo energético.

Referencias

Agudelo, N., Amezquita, J., & Silva, J. (2018). *IMPLEMENTACION DE SISTEMA DE INTEROPERABILIDAD IOT PARA TECNOLOGÍAS INALÁMBRICAS ESTABLECIDAS EN LOS ESTÁNDARES IEEE 802.15.1, IEEE 802.15.4, IEEE 802.11*.

Albornoz, R., & Soto, E. (2018). *Redes de Computadores I - Estudio del Estándar Zigbee*.

Amaya., C. (s.f). ´´Aplicaciones de las tecnicas y tecnlogias asociadas con la ingenierria simultanea en el sector manufacturero de Barranquilla´´. Revista cientifica de ingenieria y desarrollo.(N°8), pp. 1-24. 2000.

Arduino. (2015). *Arduino - BoardAnatomy*. https://www.arduino.cc/en/guide/BoardAnatomy

Berrone, P., & Ricart, J. (2018). *Índice IESE Cities in Motion*. https://doi.org/10.15581/018.ST-471

Cardador, R. (2017). *REDES DE SENSORES CON XBEE*.

Components 101. (2017). *Sensor ultrasónico HC-SR04: funcionamiento, diagrama de pin, descripción y hoja de datos*. https://components101.com/ultrasonic-sensor-working-pinout-datasheet

Crespo, E. (2016). *IEEE 802.15.4 | Aprendiendo Arduino*. https://aprendiendoarduino.wordpress.com/tag/ieee-802-15-4/

Cruz, J. C. (2018). *ZigBee para IoT - Análisis de la Tecnología ZigBee para su uso en el Internet de las Cosas - Medium*. https://medium.com/análisis-de-la-tecnología-zigbee-para-su-uso-en-el/zigbee-para-iot-12666b636821

Digi. (2020). *XCTU - Next Gen Configuration Platform for XBee/RF Solutions | Digi*

International. https://www.digi.com/products/embedded-systems/digi-xbee/digi-xbee-tools/xctu

Diosdado, R. (2015). *Zona Maker - Ultrasonido HC-SR04*. https://www.zonamaker.com/arduino/modulos-sensores-y-shields/ultrasonido-hc-sr04

El Telégrafo. (2017, December 29). *El Telégrafo - La contenerización de basura registra problemas en Quito*. 2017-12-29. https://www.eltelegrafo.com.ec/noticias/quito/1/la-contenerizacion-de-basura-registra-problemas-en-quito

Elinoff, G. (2017). *Bluetooth vs Wi-Fi vs ZigBee - Productos electrónicos*. https://www.electronicproducts.com/Computer_Peripherals/Communication_Peripherals/Bluetooth_vs_Wi_Fi_vs_ZigBee.aspx

EMASEO EP. (2016). *Proceso de Instalación - Emaseo EP*. Proceso de Instalación. http://www.emaseo.gob.ec/servicios/recoleccion-mecanizada/proceso-de-instalacion/

EMASEO EP. (2017). *Servicios de Emaseo EP: Sistema de Recolección no Mecanizada - Emaseo EP*. Servicios de Emaseo EP: Sistema de Recolección No Mecanizada. http://www.emaseo.gob.ec/servicios-emaseo-ep-sistema-recoleccion-no-mecanizada/

Fontuño, A. G. (2012). *Desarrollo e implementación de una red de sensores Zigbee mediante el dispositivo Xbee de Digi* [Universidad ETSE]. http://deeea.urv.cat/public/PROPOSTES/pub/pdf/1957pub.pdf

Hernández, L. (2016). *Sensor ultrasónico, mideindo la distancia en un sistema de aparcamiento*. https://programarfacil.com/blog/arduino-blog/sensor-ultrasonico-arduino-medir-distancia/

Horner, J. (2019). *WiFi VS Bluetooth VS ZigBee: ¿Cómo elegir el protocolo de conexión apropiado? El | Blog de GearBest*. https://www.gearbest.com/blog/how-to/wifi-vs-

bluetooth-vs-zigbeehow-to-choose-the-appropriate-connection-protocol-6905

Hubbard, L. (2017). *CONNECT WITH CONFIDENCE*. https://docplayer.net/48225255-Connect-with-confidence.html

Hussein, Z. K., Hadi, H. J., Abdul-Mutaleb, M. R., & Mezaal, Y. S. (2020). Low cost smart weather station using Arduino and ZigBee. *Telkomnika (Telecommunication Computing Electronics and Control)*, *18*(1), 282–288. https://doi.org/10.12928/TELKOMNIKA.v18i1.12784

Ingeniería Mecafenix. (2017). *Arduino ¿Que es, como funciona? y sus partes - Ingeniería Mecafenix*. https://www.ingmecafenix.com/electronica/arduino/

La Cámara. (2020). *LA CÁMARA- Quito, el cantón más poblado del Ecuador en el 2020 – CAMICON*. Quito, El Cantón Más Poblado Del Ecuador En El 2020. https://www.camicon.ec/la-camara-quito-el-canton-mas-poblado-del-ecuador-en-el-2020/

La Hora. (2019, October 17). Contenedores saturados de basura : Noticias Carchi : La Hora Noticias de Ecuador, sus provincias y el mundo. *Contenedores Saturados de Basura*. https://www.lahora.com.ec/noticia/1102279811/contenedores-saturados-de-basura

La Nota Positiva. (2019). *El colombiano que creó EcoBox*. https://lanotapositiva.com/actualidad/el-colombiano-que-creo-ecobox-las-maquinas-que-pagan-por-reciclar-en-transmilenio_9315

Linero, R., Camargo, L., & Medina, B. (2015). *Vista de Análisis del rendimiento de redes basadas en el estándar IEEE 802.15.4*. https://revistas.uis.edu.co/index.php/revistauisingenierias/article/view/71-79/4982

Llamas, L. (2015). *Medir distancia con Arduino y sensor de ultrasonidos HC-SR04*.

https://www.luisllamas.es/medir-distancia-con-arduino-y-sensor-de-ultrasonidos-hc-sr04/

Manaure, A. (2019). *ECO-BITS, el contenedor de residuos con tecnología IoT*. http://www.digitaltoo.com/2017/12/18/eco-bits-contenedor-residuos-tecnologia-iot/

Microsoft. (2018). *Estándares de protocolo definidos por proyecto IEEE 802 y FDDI*. https://support.microsoft.com/es-ec/help/103954/protocol-standards-defined-by-ieee-project-802-and-fddi

Montiel Salazar, Y. N. (2019). *Análisis de la ciudad de Guayaquil hacía un nuevo modelo como ciudad inteligente.* [Universidad de Guayaquil. Facultad de Ingeniería Industrial. Carrera de Ingeniería en Teleinformática.]. http://repositorio.ug.edu.ec/handle/redug/46701

Patagoniatec. (2015). *≫ SR04 ▷ Medir Distancia Por Ultrasonido Con Arduino | PatagoniaTec*. https://saber.patagoniatec.com/2014/10/tutorial-modulo-ultrasonico-arduino-argentina-ptec-sr04-srf05-us020/

Pinos, L., Becerra, M., & Alfonso, P. (2015). *Manual para la aplicación de la tecnología ZigBee para edificios inteligentes en la ciudad de Cuenca*.

Pye, A. (2020). *El aguijón del zigbee*. https://www.environmentalengineering.org.uk/news/the-sting-of-the-zigbee-1835/

Ruggeri, P. (2018). *ECO-BITS, Recolección Inteligente de Residuos | Espacio Sustentable*. https://espaciosustentable.com/recoleccion-de-residuos/

Travel + Leisure. (2018). *Las 10 ciudades más inteligentes del mundo*. https://travelandleisure.mx/destinos/2018/08/20/las-10-ciudades-inteligentes-del-mundo/

UIT. (2020). *Ciudades inteligentes y sostenibles*.

https://www.itu.int/es/mediacentre/backgrounders/Pages/smart-sustainable-cities.aspx

Anexo A: Código Arduino de Routers

```
//Pines de sensor y # de muestras para promediar
int const trig = 4;
int const echo = 2;
int const vcc = 9;
int muestra = 20; //# muestras tomadas por el sensor para promediar
//Variables para detección de #cifras en valor leído
int unidad, decena, centena,a,b,c;
int x; //bandera para salir de if
int d; //Valor leído (disntancia entre tapa de contenedor y nivel de basura)
void setup(){
  Serial.begin(9600);
  pinMode(trig, OUTPUT);
  pinMode(echo, INPUT);
  pinMode(vcc, OUTPUT);
}
void loop() {
  x=1;
  lectura();
  //----- Para una lectura de 1 cifra ---------
  if (0<=d && d<10 && x==1){
    trama1();
    x=2;
```

```
  }
  //----- Para una lectura de 2 cifras ---------
  if (10<=d && d<100 && x==1){
    trama2();
    x=2;
  }
  //----- Para una lectura de 3 cifras ---------
  if (100<=d && d<1000 && x==1){
    trama3();
    x=2;
  }
  delay(1000);
}
void lectura(){ //lectura del sensor ultrasonido
  digitalWrite(vcc, HIGH); //se alimenta al sensor
  for (int j = 0; j<muestra; j++){
    digitalWrite(trig,LOW);
    delayMicroseconds(5);
    digitalWrite(trig, HIGH);
    delayMicroseconds(10);
    float t = pulseIn(echo, HIGH);
    digitalWrite(trig,LOW);
    delayMicroseconds(5);
```

```
    d += int(0.017 * t);
  }
  digitalWrite(vcc,LOW); //Se desactiva al sensor para reducir consumo energético
  d = d/muestra; //promedio de muestras tomadas para mayor exactitud
  return d;
}

void trama1(){   //Trama para lectura de 1 cifra
  Serial.write(0x7E);
  Serial.write((byte)0x0);
  Serial.write(0x0F);
  Serial.write(0x10);
  Serial.write(0x01);
  Serial.write((byte)0x0);
  Serial.write((byte)0x0);
  Serial.write((byte)0x0);
  Serial.write((byte)0x0);
  Serial.write((byte)0x0);
  Serial.write((byte)0x0);
  Serial.write(0xFF);
  Serial.write(0xFF);
  Serial.write(0xFF);
  Serial.write(0xFE);
```

```
  Serial.write((byte)0x0);
  Serial.write((byte)0x0);
  unidad = d;
  a = unidad + 30 + 18;
  Serial.write(a);
  long chexsum1 = (0x10 + 0x01 + 0x00 + 0x00 + 0x00 + 0x00 + 0x00 + 0x00 + 0xFF +
0xFF + 0xFF + 0xFE + 0x00 + 0x00 + a);
  Serial.write( 0xFF - (chexsum1 & 0xFF));
}

void trama2(){  //Trama para lectura de 2 cifras
  Serial.write(0x7E);
  Serial.write((byte)0x0);
  Serial.write(0x10);
  Serial.write(0x10);
  Serial.write(0x01);
  Serial.write((byte)0x0);
  Serial.write((byte)0x0);
  Serial.write((byte)0x0);
  Serial.write((byte)0x0);
  Serial.write((byte)0x0);
  Serial.write((byte)0x0);
  Serial.write(0xFF);
```

```
  Serial.write(0xFF);
  Serial.write(0xFF);
  Serial.write(0xFE);
  Serial.write((byte)0x0);
  Serial.write((byte)0x0);
  decena = d/10;
  unidad = d - (decena*10);
  a = unidad + 30 + 18;
  b = decena + 30 + 18;
  Serial.write(b);
  Serial.write(a);
  long chexsum1 = (0x10 + 0x01 + 0x00 + 0x00 + 0x00 + 0x00 + 0x00 + 0x00 + 0xFF +
0xFF + 0xFF + 0xFE + 0x00 + 0x00 + a + b);
  Serial.write( 0xFF - (chexsum1 & 0xFF));
}

void trama3(){  //Trama para lectura de 3 cifras
  //----- Capa 1: Inicio de la trama -------------------
  Serial.write(0x7E);
  //----- Capa 2: Longitud de la data ------------------
  Serial.write((byte)0x0);
  Serial.write(0x11);
  //----- Capa 3: Inicio de la data --------------
```

```
Serial.write(0x10); //Tipo de la data
Serial.write(0x01); //ID de la data
//Dirección de destino (este router envía por broadcast a toda la red)
Serial.write((byte)0x0);
Serial.write((byte)0x0);
Serial.write((byte)0x0);
Serial.write((byte)0x0);
Serial.write((byte)0x0);
Serial.write((byte)0x0);
Serial.write(0xFF);
Serial.write(0xFF);
Serial.write(0xFF);
Serial.write(0xFE);
Serial.write((byte)0x0);
Serial.write((byte)0x0);
//Data leída por el sensor, se la transforma a HEX para enviar en la trama
centena = d/100;
decena = (d - (centena*100))/10;
unidad= d -((centena*100)+(decena*10));
a = unidad + 30 + 18;
b = decena + 30 + 18;
c = centena + 30 + 18;
Serial.write(c); //centena del valor leído por el sensor
```

```
  Serial.write(b); //decena del valor leído por el sensor
  Serial.write(a); //unidad del valor leído por el sensor
  //----- Fin de la data ----------------------
  //----- Capa 4: Sumatoria de la data ------------------------
  long chexsum1 = (0x10 + 0x01 + 0x00 + 0x00 + 0x00 + 0x00 + 0x00 + 0x00 + 0xFF +
0xFF + 0xFF + 0xFE + 0x00 + 0x00 + c + b + a);
  Serial.write( 0xFF - (chexsum1 & 0xFF));
}
```

Anexo B: Código Arduino de Coordinador

```
#include <SoftwareSerial.h>

SoftwareSerial Xbee(10, 11); // pines 10(RX) y 11(TX) que conecta a Xbee Coordinador

int valor=0, centena=0, decena=0, unidad=0; //variables para lectura del valor dado por el sensor


const int mini = 0;    //valor mínimo que se leerá

const int maxi1 = 200; //valor máximo que se leerá, altura interna en cm del contenedor 1

const int maxi2 = 200; //valor máximo que se leerá, altura interna en cm del contenedor 2


void setup() {

  Serial.begin(9600); //Iniciar comunicación serial con Processing desde el COM

  Xbee.begin(9600);   //Iniciar comunicación serial con Xbee Coordinador en pines 10 y 11

}


void loop() {

  int x=1;

  if (Xbee.available() >= 23) {  //1er filtro de tramas leidas

    byte inicio = Xbee.read();//byte inicial de trama captada

    if (inicio == 0x7E) {  //2do filtro de tramas (solo tramas que inicien con 0x7E se toman en cuenta)

      byte long1 = Xbee.read();//1er byte de capa length (normalmente es 0)
```

```
byte long2 = Xbee.read();//2do byte de capa length (muestra longitud de la capa de data)
for (int j = 0; j<8; j++) {  //salto de bytes hasta el byte identificador de cada router
  byte salto1 = Xbee.read();
}
byte mascara = Xbee.read(); //Byte para identificar de qué router viene la trama
for (int i = 1; i<5; i++) { // Salto a los valores del sensor
  byte salto2 = Xbee.read();
}
if (long2 == 0x0D && x==1){ //Para valores leidos del sensor que tengan 1 cifra
  cifras1();
  x=2;        //salida del if
}
if (long2 == 0x0E && x==1){  //Para valores leidos del sensor que tengan 2 cifras
  cifras2();
  x=2;        //salida del if
}
if (long2 == 0x0F && x==1){  //Para valores leidos del sensor que tengan 3 cifras
  cifras3();
  x=2;        //salida del if
}
// ----------Máscara de Router 1----------
if (mascara == 0xB6){
```

```
        if (valor>= mini && valor<=maxi1){
          Serial.write('A');
          Serial.write(valor);
        }
      }
      // ---------------Máscara de Router 2-----------------
      if (mascara == 0xB7){
        if (valor>= mini && valor<=maxi2){
          Serial.write('B');
          Serial.write(valor);
        }
      }
    }else{
      Serial.println("Sin valor");
     }
  }
}
//--------- Funciones para valores de 1, 2 y 3 cifras ------------
void cifras1() {
  unidad = Xbee.read();
  valor = unidad-48; //Conversión del valor leído en decimal
}
```

```
void cifras2() {
  decena = Xbee.read();
  unidad = Xbee.read();
  valor = (((decena-48)*10)+(unidad-48)); //Unificación de las cifras en 1 valor
}

void cifras3() {
  centena = Xbee.read();
  decena = Xbee.read();
  unidad = Xbee.read();
  valor = (((centena-48)*100)+((decena-48)*10)+(unidad-48)); //Unificación de las cifras en
1 valor
}
```

Anexo C: Código Processing para Interfaz

```
import processing.serial.*;  //biblioteca para lectura de puerto serial
Serial port;       //Nombre del puerto serial
  int xti=510;     //Posición X del título
  int yti=30;      //Posición Y del título
  int xb = 210;    //Posición X de subtítulo
  int yb = 140;    //Posición Y de subtítulo
  int mascara;     //Identificador de router
  int dato1 = 0;   //lectura del router 1
  int dato2 = 0;   //lectura del router 2
  int mini = 0;    //valor mínimo leído
  int maxi1 = 200; //valor máximo leído del contenedor 1 (es la altura en cm interna del
contenedor 1)
  int maxi2 = 200; //valor máximo leído del contenedor 2 (es la altura en cm interna del
contenedor 2)
  int nvl1, nvl2;  //nivel de las barras gráficas, en porcentaje
  int xbar = 70;   //Posición X del fondo de barra 1
  int ybar = 280;  //Posición Y del fondo de barra 1
  int albar = 350; //altura del fondo de las barras
  int alvar1 = 0;  //altura de barra 1
  int alvar2 = 0;  //altura de barra 2
  int alrt=80;     //porcentaje desde el cuál se alerta para vaciar el contenedor
```

```
void setup () {
  size(900, 700);    //tamaño de interfaz
  port = new Serial(this,"COM9", 9600);    //puerto serial de lectura
  port.bufferUntil('\n');
  background(0);
}

void draw () {
  // Detalles iniciales de la parte superior----------
  imageMode(CENTER);//Esta función hace que las coordenadas de la imagen sean el
centro de esta y no la esquina izquierda arriba
  PImage imagen=loadImage("circ.jpg");  // Fondo de interfáz
  image(imagen,450,350,900,700);
  PImage imagen1=loadImage("Lizq.png");
  image(imagen1,xti-280,yti+11,200,35);
   PImage imagen2=loadImage("logo.jpg");
  image(imagen2,xti-400,yti+10,220,80);
  PImage imagen3=loadImage("Lizq.png");
  image(imagen3,xti+200,yti+11,200,35);
  PImage imagen4=loadImage("digi.png");
  image(imagen4,xti+300,yti+11,80,50);

  //-------- Título ------------
```

```
 fill(0); //relleno cuadro fondo de título
 rect(xti-200,yti-5,400,35,90);   // (A,B,C,D,E)
// A = Posición en X ~~ B = Posición en Y
// C = Ancho del rectángulo ~~ D = Alto del rectángulo
// E = Borde del rectángulo
 fill(255);              //relleno de título
 textSize(20);           //tamaño del texto
 textAlign(CENTER, TOP); //alineación del texto
 text("MONITOREO DEL NIVEL DE LLENADO",xti,yti);
 // Recepción de datos por serial ----------------------
 if(port.available() > 0){ // si hay algún dato disponible en el puerto
   mascara = port.read();//Lee el primer byte para reconocer de qué router viene
   if (mascara == 'A'){  //Máscara de router 1
     dato1 = port.read();
   }
   if (mascara == 'B'){  //Máscara de router 2
     dato2 = port.read();
    }
 }
 //-------- Router 1 ------------
 PImage imagen5=loadImage("xb.png"); //imagen de Xbee
 image(imagen5,xb,yb,80,80);
 PImage imagen6=loadImage("cont.png"); //imagen de contenedor 1
```

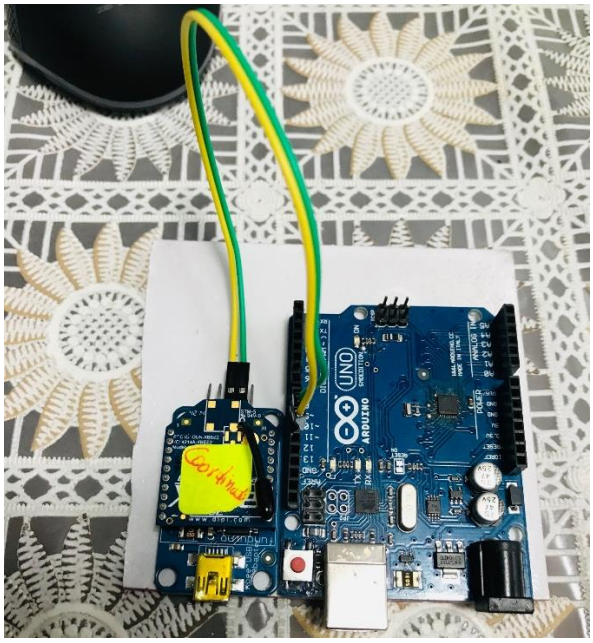
```
 image(imagen6,xbar+240,ybar+260,200,200);
 fill(0);
 textSize(22);   // tamaño de subtítulos
 text("Router 1",xb+100,yb-10);
 text("Nivel de contenedor 1",xb+70,yb+80);
 textSize(14);          // tamaño del texto ID
 text("ID: 0013A2004191B6FB",xb+70,yb+50);
 nvl1 = 100-((dato1*100)/maxi1);  //porcentaje de llenado en base a lectura de sensores
 alvar1= (nvl1*albar)/100;       //altura de barra 1 en base a porcentaje de llenado
 if (dato1>=mini && dato1<=maxi1){  //límites para lectura de datos
  textSize(28);
  text(nvl1, xbar+230, ybar+120);
  text("%", xbar+270, ybar+120);
  if(nvl1>=alrt){    //Alerta para vaciar contenedor
    PImage imagen7=loadImage("alerta.png");
    image(imagen7,xbar+245,ybar+40,80,60);
    textSize(16);
    text("Necesita vaciarse", xbar+240, ybar+80);
  }
 }
 fill(255);      // relleno del fondo de barra 1
 rect(xbar,ybar,40,albar,90); //fondo de barra 1
 fill(0,0,150);      // relleno de barra 1
```

```
 rect(xbar+3,ybar+3+(albar-alvar1),34,alvar1-6,90); // barra 1
 fill(0);    //relleno de indicadores
 textSize(20);    //tamaño de indicadores
 text("--Lleno",xbar+85,ybar-10);
 text("--Medio",xbar+85,ybar-13+(albar/2));
 text("--Vacío",xbar+85,ybar-16+albar);

 //-------- Router 2 ------------
 PImage imagen8=loadImage("xb.png");    //Imagen de Xbee
 image(imagen8,xb+400,yb,80,80);
 PImage imagen9=loadImage("cont2.png"); //Imagen de contenedor 2
 image(imagen9,xbar+660,ybar+260,230,200);
 fill(0);              // relleno de subtítulos
 textSize(22);         // tamaño de subtítulos
 text("Router 2",xb+500,yb-10);
 text("Nivel de contenedor 2",xb+470,yb+80);
 textSize(14);         // tamaño del texto ID
 text("ID: 0013A2004191B71B",xb+470,yb+50);
 nvl2 = 100-((dato2*100)/maxi2);
 alvar2= (nvl2*albar)/100;
 if (dato2>=mini && dato2<=maxi2){    //límites de lectura de datos
   textSize(28);
   text(nvl2, xbar+630, ybar+120);
```

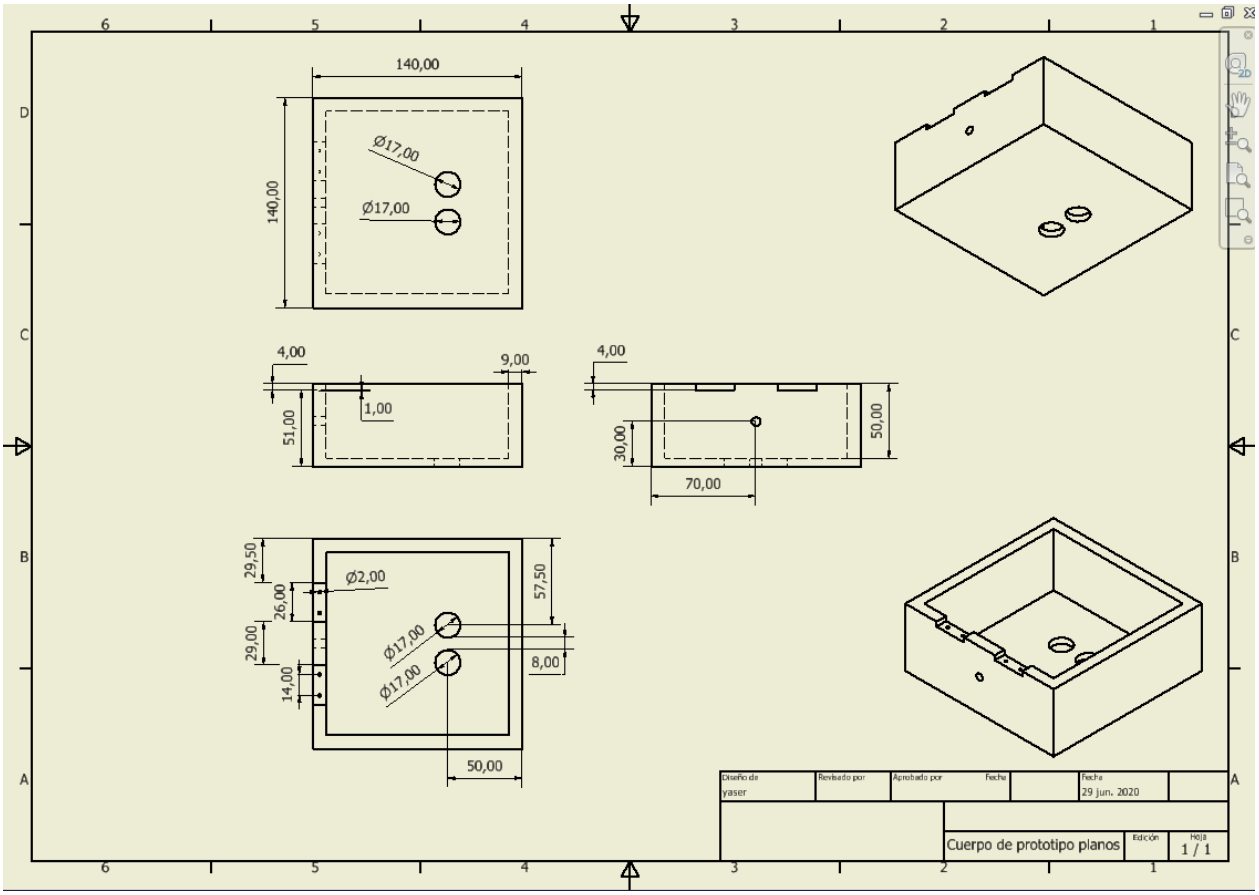
```
   text("%", xbar+670, ybar+120);
   if(nvl2>=alrt){    //Alerta para vaciar contenedor
     PImage imagen10=loadImage("alerta.png");
     image(imagen10,xbar+645,ybar+40,80,60);
     textSize(16);
     text("Necesita vaciarse", xbar+640, ybar+80);
   }
 }
 fill(255);      // relleno del fondo de barra 2
 rect(xbar+400,ybar,40,albar,90); //fondo de barra 2
 fill(0,0,150);      // relleno de barra 2
 rect(xbar+403,ybar+3+(albar-alvar2),34,alvar2-6,90); // barra 2
 fill(0);    //relleno de indicadores
 textSize(20);    //tamaño de indicadores
 text("--Lleno",xbar+485,ybar-10);
 text("--Medio",xbar+485,ybar-13+(albar/2));
 text("--Vacío",xbar+485,ybar-16+albar);
 //---------- Firma -----------------------
 fill(0);    //relleno de firma
 textSize(18);          // tamaño del texto
 text("Por: Cristian Yaser Ronquillo Muñoz",720,670);
}
```

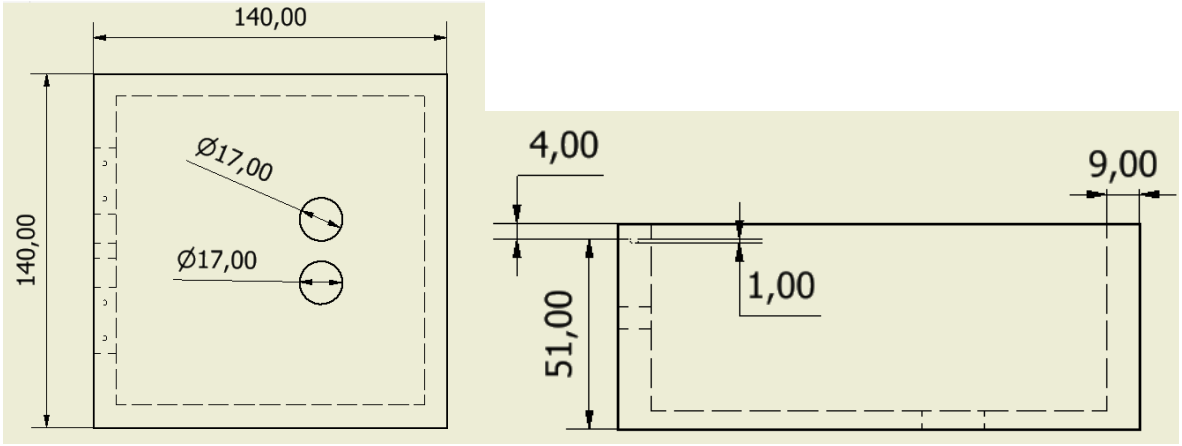
Anexo D: Prototipo del Coordinador

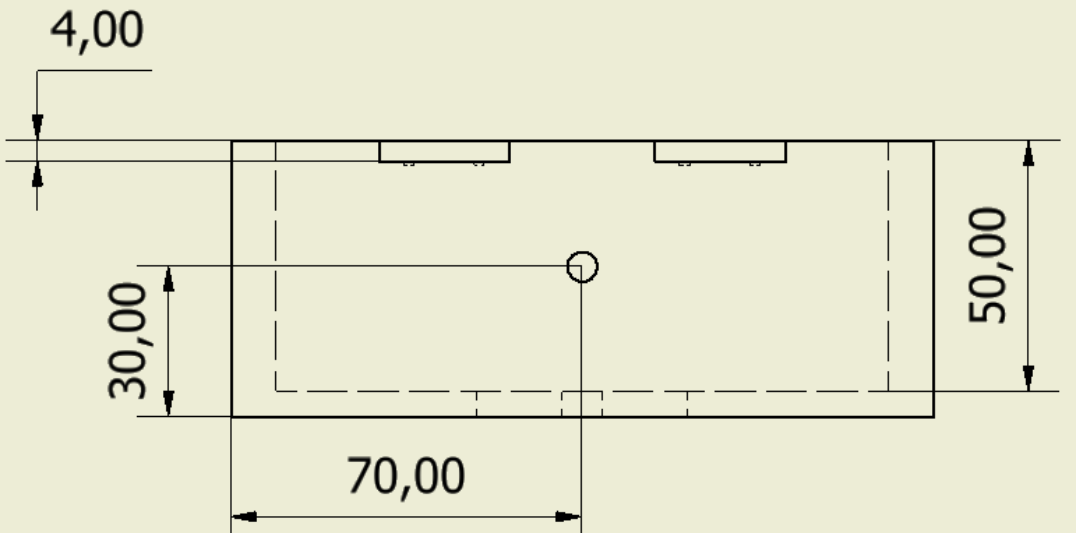


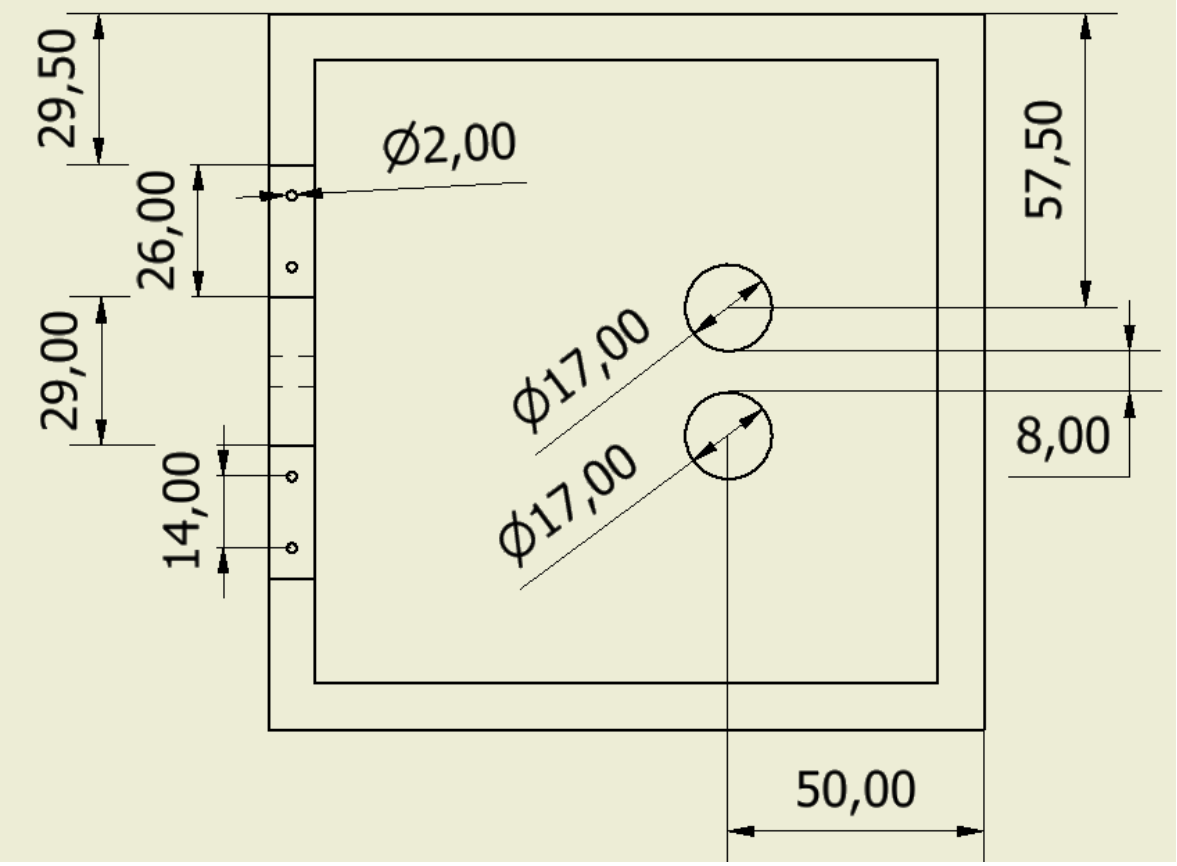
Anexo E: Planos para cuerpo del Prototipo del Router

Planos del cuerpo:

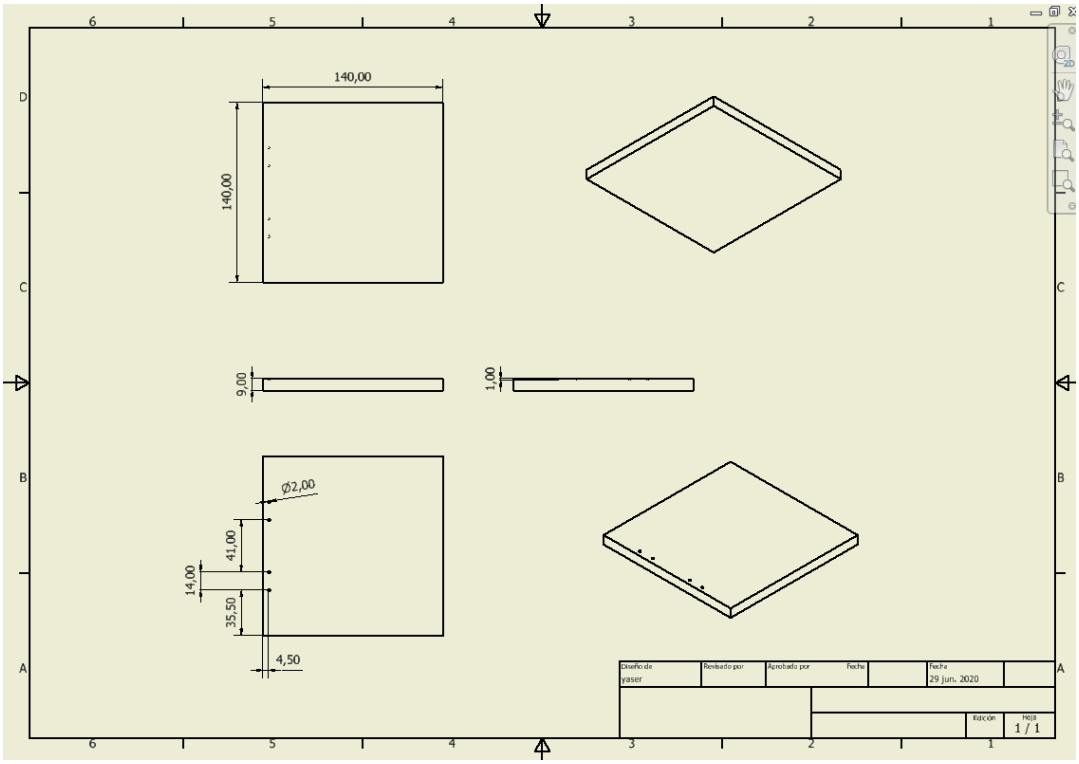


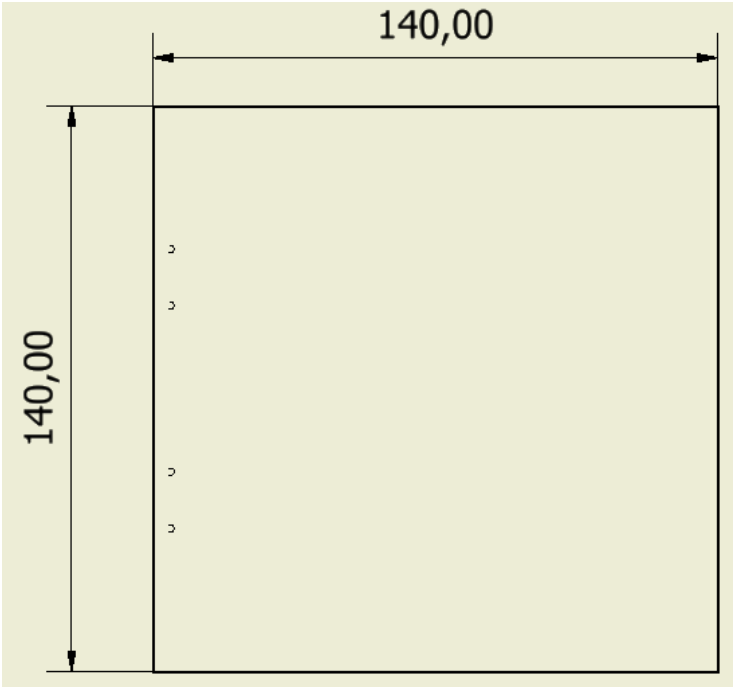


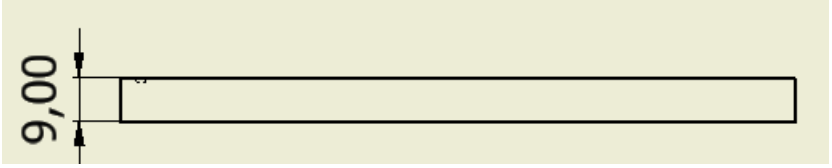

4,00
30,00
50,00
70,00

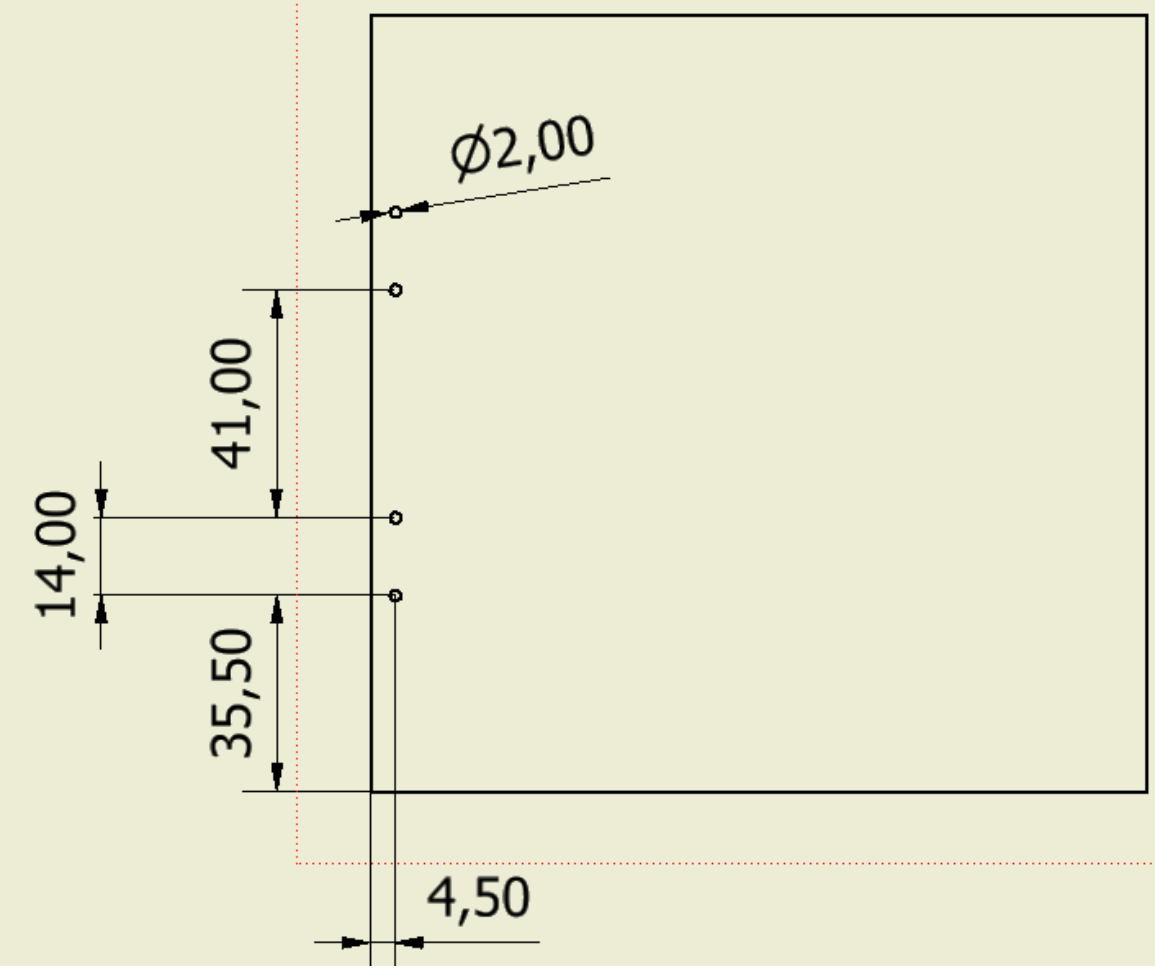

29,50
Ø2,00
26,00
57,50
29,00
Ø17,00
8,00
14,00
Ø17,00
50,00

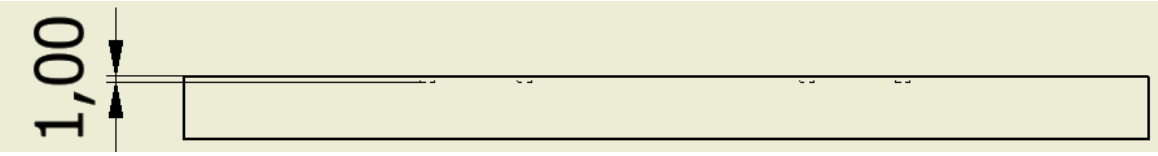
Planos de la tapa:



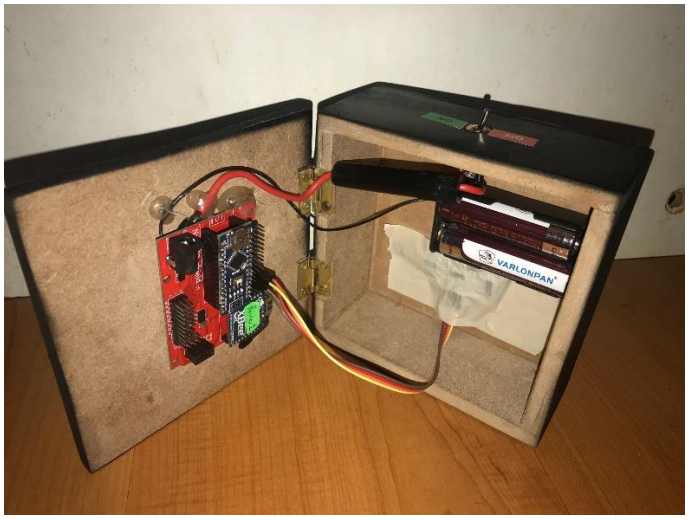


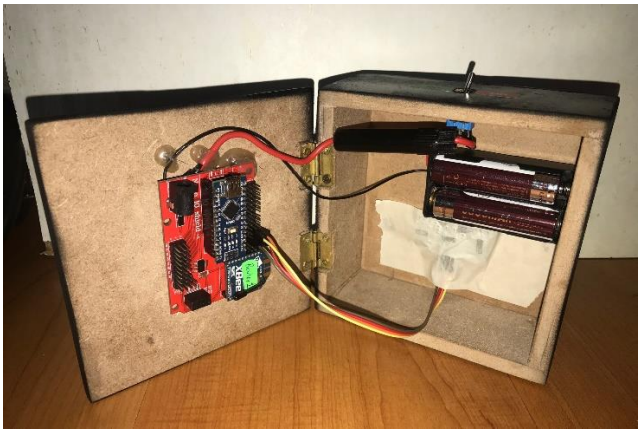



Ø2,00
41,00
14,00
35,50
4,50

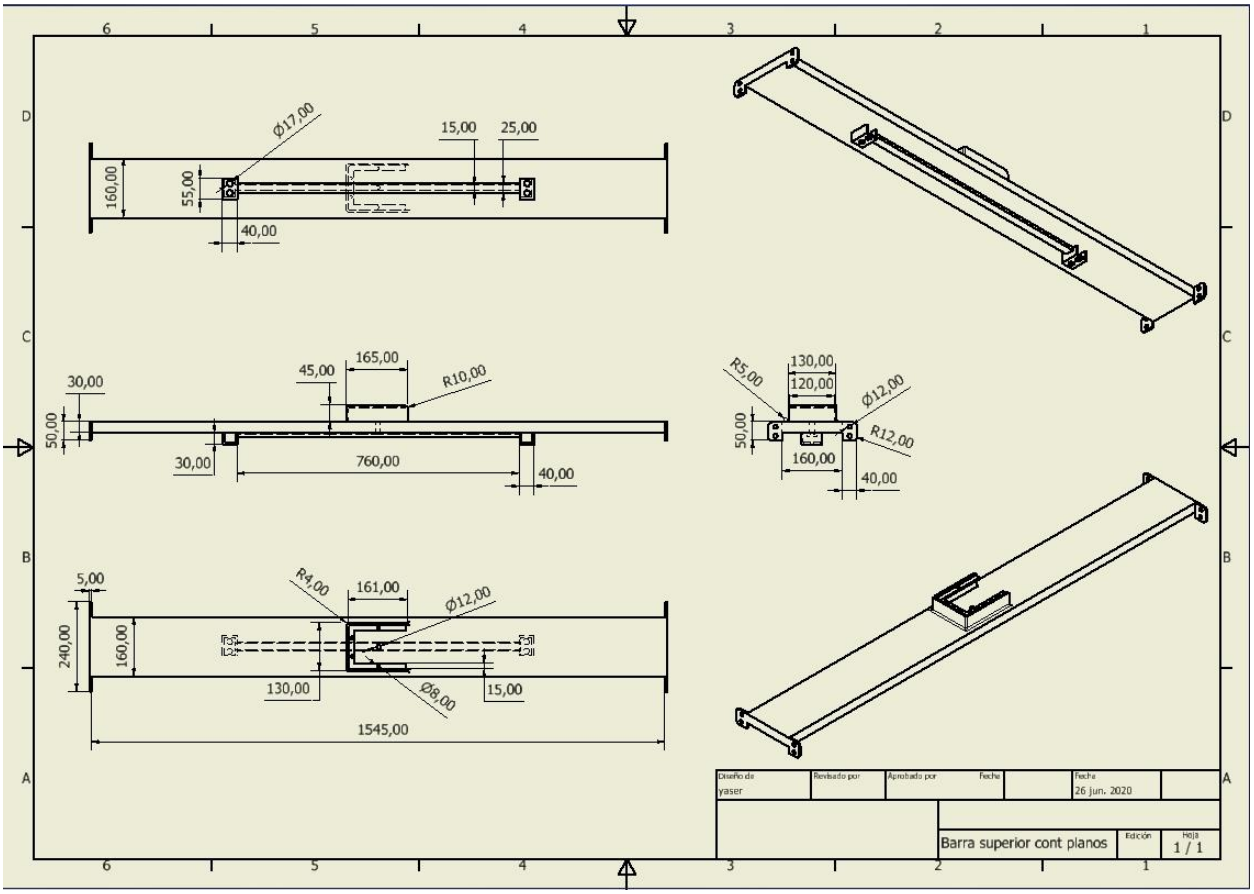

1,00

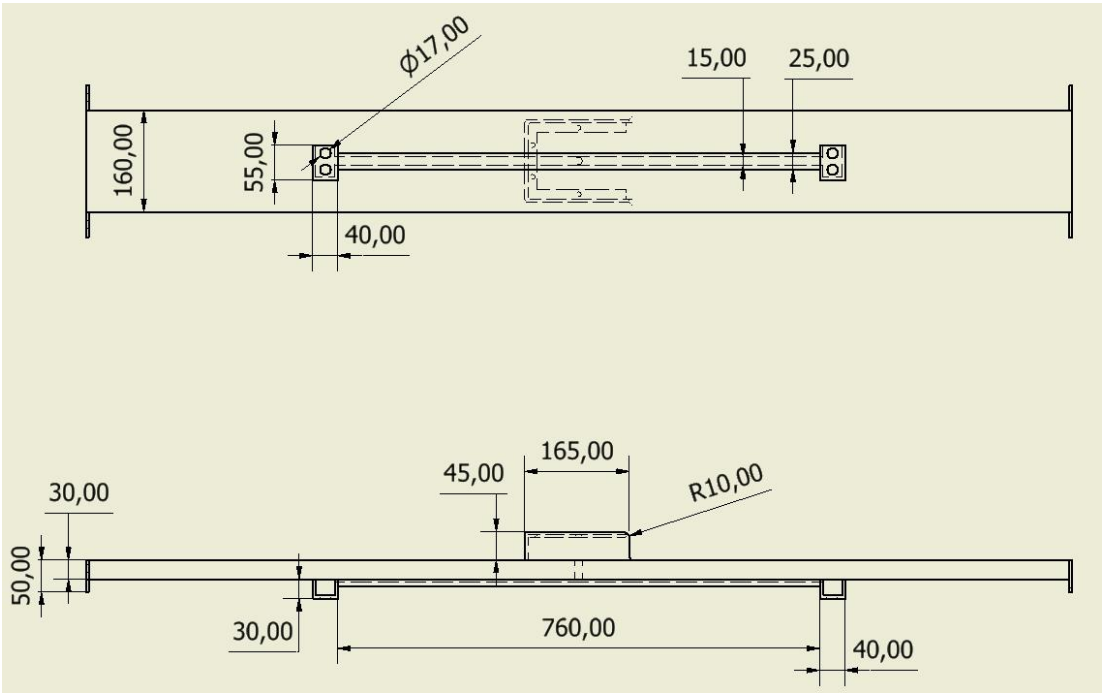
Anexo F: Prototipo del Router

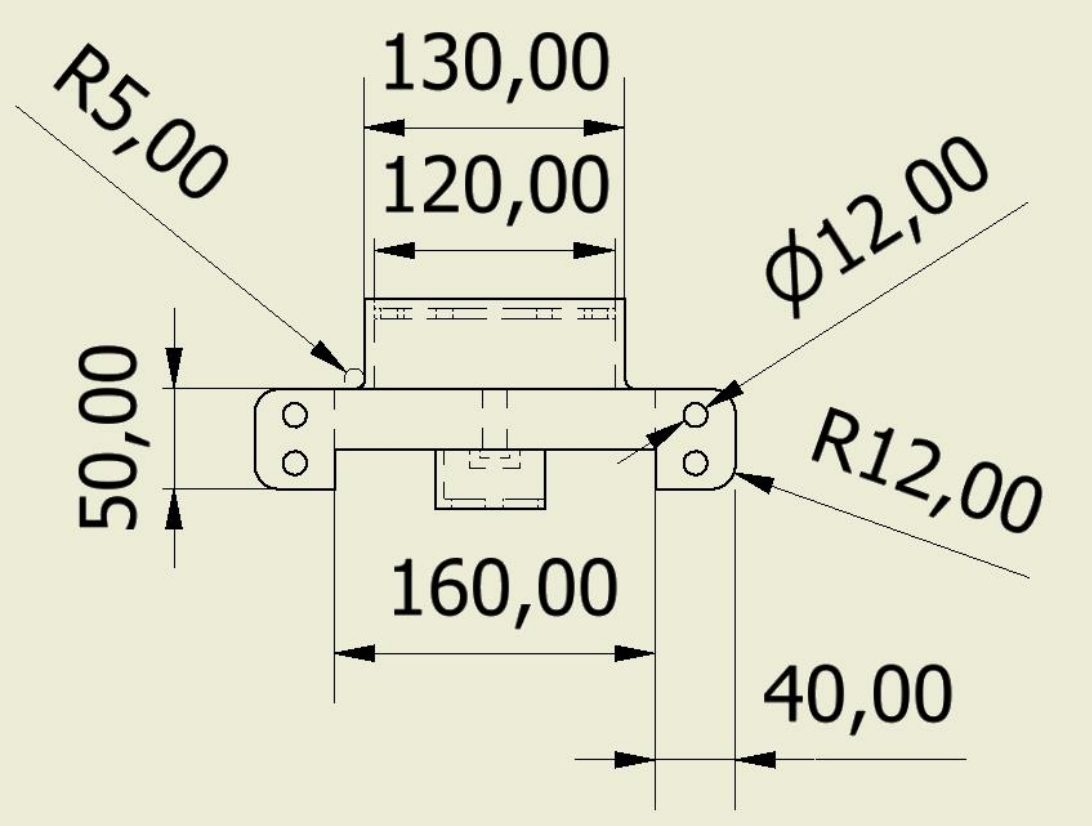


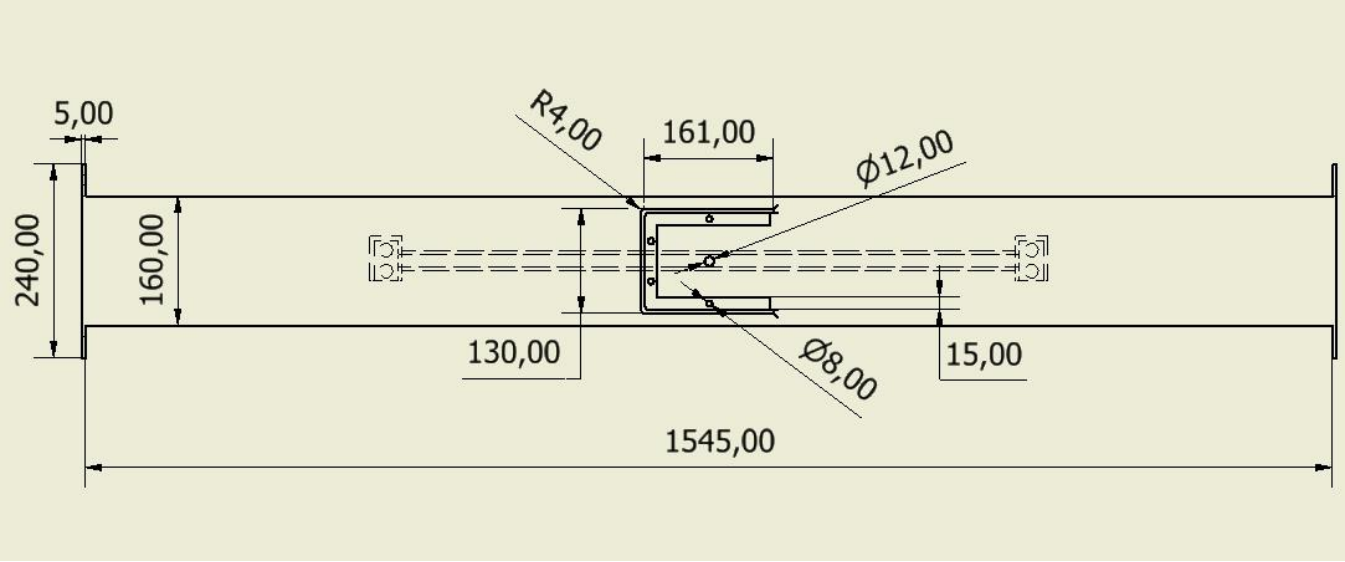


Anexo G: Planos-Rediseño de barra superior en contenedores municipales






R5,00
130,00
120,00
Ø12,00
50,00
R12,00
160,00
40,00


5,00
R4,00
161,00
Ø12,00
240,00
160,00
130,00
Ø8,00
15,00
1545,00