UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería

"Diseño, construcción e implementación de un dispositivo de internet de las cosas para eliminar errores en el proceso de despacho en un centro de distribución"

Trabajo de graduación presentado por Antonio Javier Rivera Urruela
para optar al grado académico de Licenciado en Mecánica Industrial

Guatemala,
2023

UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería

"Diseño, construcción e implementación de un dispositivo de internet de las cosas para eliminar errores en el proceso de despacho en un centro de distribución"

Trabajo de graduación presentado por Antonio Javier Rivera Urruela
para optar al grado académico de Licenciado en Mecánica Industrial

Guatemala,
2023

Vo.Bo.:

(f) _____

Ing. Víctor Hugo Ayerdi Bardales

Tribunal Examinador:

(f) _____

Ing. Víctor Hugo Ayerdi Bardales

(f) _____

Ing. José Antonio Bagur Najera

(f) _____

Ing. Andrés Rodrigo Viau Najarro

Fecha de aprobación: Guatemala 3 de octubre de 2023.

Prefacio

Este trabajo lo dedico a dos mujeres extraordinarias, mi mamá y mi abuela Mimi, cuya generosidad, aliento, cariño y sacrificio me permitieron alcanzar mis metas académicas. Su influencia perdurará en cada logro que alcance. No transcurre un día en el que no te extrañe mamá, siempre estarás en mi mente y corazón.

Índice

Prefacio v

Lista de figuras x

Lista de cuadros xi

Resumen xiii

Abstract xv

1. Introducción 1

2. Justificación 3

3. Objetivos 5
3.1. Objetivo general . . . 5
3.2. Objetivos específicos . . . 5

4. Marco teórico 7
4.1. Sistemas . . . 7
4.1.1. Sistema como un servicio (SaaS): . . . 7
4.1.2. Sistema de planificación de recursos (ERP): . . . 7
4.1.3. ERP Cin7 Core / Dear Inventory: . . . 8
4.1.4. Formatos de intercambio de información . . . 8
4.1.5. Métodos HTTP . . . 9
4.1.6. Interfaz de programación de aplicaciones: . . . 11
4.1.7. NoCode: . . . 12
4.1.8. Zapier . . . 13
4.1.9. Servicios de computación en la nube . . . 14
4.2. Componentes electrónicos . . . 15
4.2.1. Placa de desarrollo ESP2866 . . . 15
4.2.2. Placa de desarollo ESP32 . . . 15
4.2.3. Lector de código de barras DE2120 SparkFun . . . 16

4.2.4. Conexión de piezas impresas por ajuste a presión 17

5. Metodología **19**
5.1. Medición de condición actual 19
5.2. Diseño de sistema 20
5.2.1. Definicion de funciones del sistema 22
5.2.2. Selección de componentes 22
5.2.3. Diseño y ensamble del sistema electrónico 26
5.2.4. Diseño de código en la nube 28
5.2.5. Proceso de verificación de despacho. 30
5.3. Diseño y fabricación de carcasa. 32
5.4. Capacitación de operarios. 36

6. Resultados **39**
6.1. Reducción de errores en despachos 39
6.2. Sistemas 40
6.2.1. Sistema en dispositivo 40
6.2.2. Resultados y seguimiento de despachos en la nube 41
6.3. Última iteración del prototipo 42
6.4. Capacitación de operarios. 43

7. Discusión de resultados **45**
7.1. Medición de impacto 45
7.2. Operación con el dispositivo 46

8. Conclusiones **49**

9. Recomendaciones **51**
9.1. Futuras iteraciones 51
9.2. Reducción de costos operativos 51

10. Bibliografía **53**

11. Anexos **55**
11.1. Imágenes 55
11.2. Algoritmo para verificar despacho 58
11.3. Código en despachador 62
11.4. Diagrama de conexiones 69
11.5. Juego de planos 70

Lista de figuras

1. Comparación entre formatos JSON y XML . . . 10
2. Ejemplo de parámetro de URL . . . 12
3. Ejemplo de programación con Zapier . . . 13
4. Placa de desarrollo ESP8266 Node MCU . . . 15
5. Lector de códigos de barra Sparkfun DE2120 . . . 17
6. Conexión por ajuste a presión . . . 18

7. Proceso de despacho con verificación automatizada. . . . 21
8. Diagrama de flujo de información para verificar despacho. . . . 23
9. Placa de desarrollo *ESP8266 NodeMCU* . . . 24
10. Lector de códigos de barra Sparkfun DE2120 . . . 25
11. Primera iteración del dispositivo . . . 26
12. Código C++ para concatenar y subir información . . . 27
13. Diagrama de conexiones de última iteración . . . 28
14. Flujo de sistema en Zapier . . . 29
15. Flujo en Zapier cuando la orden escanéada existe . . . 30
16. Diagrama de flujo de algoritmo para verificar despachos . . . 31
17. Respuesta del dispositivo en caso de despacho incorrecto . . . 32
18. Diseño CAD de carcasa . . . 33
19. Conexión por ajuste a presión CAD. . . . 33
20. Primera iteración de diseño . . . 34
21. Generación de código G para impresión en Ultimaker Cura . . . 34
22. Diseño original de pedestal. . . . 35
23. Diseño modificado de pedestal . . . 35
24. Falla en pedestal . . . 36
25. Dispositivos siendo ensamblados . . . 36
26. Modo de registro de número de orden . . . 37
27. Modo de registro de códigos SKU . . . 37
28. Modo de registro de números de parte . . . 37
29. Respuesta del dispositivo . . . 38

30. Última iteración de sistema programado en dispositivo . . . 40
31. Historial de corridas exitosas en *Zapier* . . . 41

32. Dispositivo ensamblado 42
33. Video de capacitación de operarios 43
34. Video de proceso de despacho realizado por operario. 43
35. Guía de despacho colocado en área de trabajo. 44

36. Productos con etiquetas equivocadas 46

37. Anexo 2. Primera conexión con carcaza 55
38. Anexo 1. Primera iteración 56
39. Anexo 3. Despachador con componentes Electrónicos Ensamblado 57
40. Anexo 4. Iteración de despachador con componentes electrónicos ensamblados 57
41. Anexo 5. Diagrama de conexiones 69

Lista de cuadros

1. Nomenclatura de entradas del ESP8266 en IDE de Arduino 16

2. Cantidad de devoluciones y sus razones, en un periodo de 30 dias. 20
3. Costos directos por cada despacho incorrecto. 20

4. Devoluciones y sus razones después de implementar el dispositivo 39

5. Devoluciones después de implementar el dispositivo con postprocesado de datos 46

Resumen

Los sistemas de ERP (*Enterprise resource planning*) son excelentes herramientas para la administración de inventarios. Estos sistemas pueden carecer de verificaciones automáticas en las etapas de una venta. Una de las etapas más importantes es el despacho de la orden. Durante el proceso de despacho pueden existir errores humanos por parte del operador cuando no existe esta etapa de verificación. El error más común es enviar una cantidad incorrecta de artículos ó artículos no solicitados en la orden. Este proyecto consistió en el diseño, construcción e implementación de un dispositivo electrónico para verificar y registrar despachos de órdenes en el ERP Cin 7 Core. El objetivo del proyecto es eliminar los errores humanos del operador durante el despacho de órdenes de venta, obligando al operador a escanear todos los productos de la orden, comprobando automáticamente que esta esté correcta, para luego registrar la operación en el ERP. Se construyó un dispositivo de internet de las cosas (*IoT*. Este capta todos los códigos de barra del despacho y los sube a la nube, donde se verifica por medio de un algoritmo el despacho contra la orden de venta. Si el despacho es correcto, el proceso se registra en el ERP, de no ser correcto se alerta al operador y se detiene el registro.

Abstract

Enterprise Resource Planning (ERP) systems are excellent tools for inventory management, but these systems may lack automatic checks at different stages of a sale, specially *SaaS ERPs* with limited customization options. During the fulfillment process, human errors can occur on the part of the operator when there is no verification step in place. The most common error is sending an incorrect quantity of items or items not requested in the order. This project involved the design, construction, and implementation of an internet of things device to verify and record order fulfillment in the ERP Cin 7 Core. The project's objective is to eliminate human errors by the operator during the order fulfillment, requiring the operator to scan all the products in the order, automatically verifying its correctness through an array comparison algorithm, and then recording the operation in the ERP.

The device captures all the barcode data from the fulfillment process and uploads it to the cloud, where it is checked against the sales order using an algorithm developed in python. If the fulfillment is correct, the process is recorded in the ERP. If it is not correct, the operator is alerted, and the fulfillment is halted. The entire process occurs in a matter of seconds, while the order is being packed, preventing the error reaching the customer.

CAPÍTULO 1

Introducción

Este proyecto propone la construcción, diseño e implementación de un dispositivo de internet de las cosas donde el operador debe escanear cada artículo que empaca durante el proceso de despacho. Este dispositivo sube a la nube, verifica, registra el despacho y envía una respuesta de vuelta al operador sobre el resultado de forma automática. El ERP *Cin 7 Core*, no cuenta con una función qué detenga al operador de completar el despacho, sin haber completado una verificación. Actualmente, no existe un escáner de códigos de barra inteligente que se integre con *Cin 7 Core* para comprobar los despachos.

La metodología de desarrollo fue incremental con diseño de la ingeniería. Este enfoque implica dividir un proyecto de desarrollo en etapas pequeñas, donde cada etapa se completa antes de pasar a la siguiente. Cada paso o etapa se valida y verifica antes de avanzar, lo que permite detectar posibles problemas o errores temprano en el proceso de desarrollo. Evita modificaciones por cambios en los pilares del diseño.

Se midieron los despachos incorrectos durante un mes previo a implementar la solución, para así poder medir el impacto del dispositivo en el proceso. Para desarrollar el dispositivo primero se investigó sobre placas de desarrollo de internet de las cosas. Se seleccionó la placa de desarrollo ESP8266 como plataforma, ya que cuenta con un módulo *wifi* y una comunidad amplia que brinda soporte y librerías. Se seleccionó un escáner de códigos de barra marca *SparkFun* compatible con la placa de desarrollo. Para alcanzar comunicación entre el dispositivo y el ERP, se investigó sobre protocolos de comunicación *HTTP* y módulos de *API*. El escáner se integró con el ERP por medio de comunicación *HTTP* y la plataforma *Zapier*, en donde sucede el análisis de cada despacho. El proyecto se apoya en herramientas de internet de las cosas y sistemas de *NoCode* y *LowCode* como *Zapier* para crear un prototipo.

Se diseñó e imprimió una carcasa para colocar este dispositivo en Bodega, una vez instalado se limitó el ERP para que los despachos solo pudiesen hacerse por medio del dispositivo. Después de un mes de operar exclusivamente con el dispositivo se midieron las razones por las cuales se recibieron devoluciones, con esta información se midió el impacto de la implementación del dispositivo en el proceso. Se obtuvo una reducción casi completa en los errores humanos en el proceso de despacho.

CAPÍTULO 2

Justificación

La empresa Celovendo S.A. es una tienda en línea que se dedica a la distribución de componentes electrónicos. Actualmente, experimenta un problema sistémico donde están despachando y enviando erróneamente el 1 % de las órdenes recibidas de la tienda en línea. La empresa emplea el *ERP* Cin7 Core para el manejo de inventario y registro de procesos de compras y ventas.

Cin7 Core es un ERP SaaS (*software as a service*) que la empresa emplea para gestionar su inventario y procesos. Al ser un SaaS el nivel de personalización es limitado, ya que Celovendo S.A no puede hacer modificaciones en la arquitectura del ERP. Actualmente, el proceso de despacho de órdenes de venta en Cin7 Core consiste en las siguientes etapas: cotización, orden, factura, recolección, empaque y envío. Cin7 Core no tiene un método de verificación de empaque o recolección obligatorio. Previo a implementar una solución de verificación, el operario hacía la recolección en bodega de los artículos en orden de venta y marca manualmente en Cin7 Core como "recolectado" y "empacado" para registrar el despacho. No existía un procedimiento de comprobación automatizado dentro del proceso de despacho que alerte al operario en caso de que esté enviando incorrectamente una orden.

La empresa estima que sus costos directos por cada despacho incorrecto son de Q63.10. Actualmente, la empresa tiene pérdidas mensuales por Q.1,200 a Q.1,300.00 por despachos incorrectos. Celovendo proyecta aumentar la cantidad de órdenes que procesa cada mes, por lo que es importante solucionar este problema para evitar mayores costos por despachos incorrectos en el futuro.

CAPÍTULO 3

Objetivos

3.1. Objetivo general

- Eliminar los errores en el proceso de despacho en Celovendo S.A, incorporando un escáner de códigos de barra que verifica la exactitud del despacho contra la orden.

3.2. Objetivos específicos

- Diseñar un sistema de internet de las cosas que suba a un servidor los códigos de barra registrados durante el despacho de la orden.
- Crear un sistema para comprobar la exactitud de un despacho y registrarlo en el ERP Cin7 Core.
- Diseñar y construir una carcasa para el dispositivo.
- Medir la disminución de errores en despachos antes y después de instalar el dispositivo.
- Capacitar al personal de bodega sobre el uso del nuevo dispositivo en el proceso de despacho.

CAPÍTULO 4

Marco teórico

4.1. Sistemas

4.1.1. Sistema como un servicio (SaaS):

Software as a Service (SaaS) o Sistema como un servicio es un modelo de distribución de software en el que el proveedor del software aloja la aplicación en sus servidores y la pone a disposición de los usuarios a través de internet. En lugar de descargar y ejecutar el software localmente, los usuarios acceden a la aplicación a través de un navegador web y pagan por el uso de la misma, generalmente en forma de suscripción mensual o anual.(Dedase, 2019)

Este modelo de entrega de software ha ganado popularidad en los últimos años debido a su accesibilidad y conveniencia. Los usuarios no tienen que preocuparse por instalar, actualizar o mantener el software, ya que todo el proceso es manejado por el proveedor del servicio. La principal desventaja es que el cliente jamás llega a ser dueño del sistema que utiliza, lo alquila. Por lo que el nivel de personalización puede ser limitado. Desde otro punto de vista, SaaS permite a las empresas escalar fácilmente su uso del software, ya que pueden agregar o reducir usuarios según sea necesario sin preocuparse por la infraestructura subyacente. (Musick, 2020)

4.1.2. Sistema de planificación de recursos (ERP):

Un ERP (Enterprise Resource Planning) es un software que se utiliza para gestionar operaciones empresariales, como la contabilidad, la gestión de inventarios y la planificación de la producción. (Gutteridge, 2016)

En los últimos años, ha surgido una nueva forma de emplear los ERP, conocida como ERP *software as a service* (ERP SaaS). En este modelo, el software se proporciona a través de la nube, y los clientes acceden a él mediante una conexión a Internet. En lugar de instalar y mantener el software en sus propios servidores, los clientes pagan una tarifa mensual o anual para usar el software en la nube. (Marko, 2014)

El modelo de ERP SaaS tiene varias ventajas. En primer lugar, elimina la necesidad de invertir en infraestructura de tecnologias costosa, como servidores y hardware de red. En segundo lugar, reduce los costos de mantenimiento, ya que la responsabilidad de actualizar el software y realizar copias de seguridad de los datos recae en el proveedor del servicio en la nube.

Idealmente, un ERP es un sistema que enlaza y registra todas las operaciones de una empresa, como por ejemplo: contabilidad, ventas, producción, inventarios y compras. Una transacción en una de estas áreas crea registros y procesos en las demás. (Gutteridge, 2016)

4.1.3. ERP Cin7 Core / Dear Inventory:

Cin7 Core es un sistema de gestión de inventarios en la nube diseñado para pequeñas y medianas empresas. Permite a las empresas gestionar sus operaciones de inventario en tiempo real, lo que les permite optimizar su cadena de suministro y maximizar la eficiencia operativa.

Entre las funciones clave de Cin7 Core se incluyen la gestión de órdenes de compra, la gestión de órdenes de venta, la gestión de inventario y el seguimiento de la cadena de suministro. La plataforma también ofrece una amplia gama de informes y análisis para ayudar a las empresas a tomar decisiones informadas sobre su inventario y su cadena de suministro. (DearSystems, 2021)

Cin 7 Core también se integra con una variedad de plataformas de comercio electrónico, incluyendo Shopify, WooCommerce, Magento y Amazon, lo que permite a las empresas administrar sus inventarios y ventas en línea desde una sola plataforma. Una de las principales ventajas de este sistema es que cuenta con un *API* (*application programming interface*) abierto, lo que permite hacer realizar por de forma programática, esto permite integrar Cin7 Core con soluciones personalizadas.

A pesar de ser un ERP SaaS, la API permite personalizar los procesos a la empresa en la cual se utiliza esta plataforma, cuenta con documentación extensa, ejemplos y se encuentra en constante actualización. (DearSystems, 2021)

4.1.4. Formatos de intercambio de información

El intercambio eficiente y efectivo de datos entre sistemas y aplicaciones es necesario. Entre las herramientas fundamentales para lograr este objetivo, se encuentran los formatos de intercambio de datos, y dos de los más comunes son JSON (*JavaScript Object Notation*) y XML (*Extensible Markup Language*). Estos son formatos para estructurar e intercambiar datos entre servidores y aplicaciones. (Radečić, 2020)

JSON (*JavaScript Object Notation*):

JSON, como su nombre lo indica, es un formato originado en el entorno de JavaScript. Su estructura se basa en una organización simple de pares clave-valor, donde los datos son representados en forma de objetos y matrices. Los objetos JSON son encapsulados por llaves y contienen una serie de pares clave-valor separados por dos puntos. Las matrices, por otro lado, se encierran en corchetes y contienen una lista. Esta estructura simple y legible por humanos la hace popular en aplicaciones de desarrollo web. (Radečić, 2020)

Uno de los usos más notables de JSON se encuentra en la comunicación entre cliente y servidor en aplicaciones web y móviles. Además, JSON se ha convertido en el formato de elección para muchas APIs (Interfaces de Programación de Aplicaciones), facilitando la integración de diferentes sistemas. Además, JSON encuentra su nicho en el almacenamiento de configuraciones y preferencias de usuario debido a su estructura legible por humanos, permitiendo una fácil modificación y mantenimiento. (Radečić, 2020)

XML (Extensible Markup Language):

En contraste con JSON, XML es un lenguaje de marcado que enfatiza la definición de etiquetas personalizadas para describir la estructura y el contenido de los datos. A través de etiquetas de apertura y cierre, los elementos y atributos pueden ser detalladamente definidos, lo que otorga una mayor flexibilidad en la organización de datos. Aunque XML puede parecer más complejo en comparación con JSON, esta flexibilidad lo hace especialmente útil cuando se requiere una estructura de datos más compleja y cuando la validación es crucial.(Alawi, 2018)

En el ámbito empresarial, XML encuentra su aplicación en el intercambio de datos entre sistemas internos y externos. Su capacidad para definir esquemas personalizados es particularmente útil en casos en los que se necesita una validación rigurosa de los datos. Asimismo, XML es empleado para almacenar configuraciones y metadatos en aplicaciones que requieren una jerarquía más profunda y una organización precisa de los datos. La Figura 1 muestra la diferencia entre JSON y XML para representar una misma muestra de datos. (Alawi, 2018)

XML es un formato de intercambio de información más antiguo, creado en la década de los años 90. La elección de formato a utilizar depende de la arquitectura del servidor. Todo lo que se puede transmitir con JSON se puede transmitir con XML y viceversa. La elección de formatos es últimamente una decisión del programador que diseño el servidor. La mayoría de sistemas modernos utilizan JSON, debido a la reciente adoptación y crecimiento de JavaScript. (Hemel, 2007)

4.1.5. Métodos HTTP

HTTP es un juego de reglas de comunicación digital, es decir, un protocolo de transmisión de información en la web. El Protocolo de Transferencia de Hipertexto (HTTP) es el pilar fundamental de la comunicación del internet. Define cómo los clientes (como navegadores web o cualquier entidad que utiliza el recurso) y los servidores interactúan para solicitar y entregar información, como páginas web, datos, imágenes etc. Los métodos HTTP son

XML

```
<empinfo>
   <employees>
      <employee>
         <name>James Kirk</name>
         <age>40></age>
      </employee>
      <employee>
         <name>Jean-Luc Picard</name>
         <age>45</age>
      </employee>
      <employee>
         <name>Wesley Crusher</name>
         <age>27</age>
      </employee>
   </employees>
</empinfo>
```

JSON

```
{  "empinfo" :
      {
            "employees" :  [
            {
                  "name" : "James Kirk",
                  "age" : 40,
            },
            {
                  "name" : "Jean-Luc Picard",
                  "age" : 45,
            },
            {
                  "name" : "Wesley Crusher",
                  "age" : 27,
            }
                              ]
      }
   }
```

Figura 1: Comparación entre formatos JSON y XML

Fuente: Alawi, 2018

comandos que indican la acción que un cliente desea realizar en un recurso específico. Existen diferentes métodos para indicarle al servidor lo que el cliente desea.

POST: Creación de recursos

El método POST se utiliza para enviar datos al servidor y solicitar la creación de un nuevo recurso. A menudo, se emplea en formularios web para enviar información al servidor. Cuando un cliente envía una solicitud POST, los datos se incluyen en el cuerpo de la solicitud. El servidor procesa los datos y, si es válido, crea un nuevo recurso en el servidor.

Las solicitudes de tipo POST son comúnmente utilizadas para crear nuevos registros, la solicitud debe de incluir un cuerpo y encabezados. (Chateka, 2017)

GET: Obtención de recursos

El método GET se utiliza para solicitar recursos del servidor. Cuando un cliente envía una solicitud GET, los parámetros se incluyen en la URL. Esta solicitud se utiliza principalmente para recuperar información existente del servidor, como páginas web, imágenes o datos. Es importante destacar que las solicitudes GET no deben tener ningún efecto secundario en el servidor ni en los datos. (Chateka, 2017)

Un ejemplo común de uso de GET es cuando un usuario ingresa una URL en su navegador. El navegador envía una solicitud GET al servidor que aloja la página web solicitada. El servidor devuelve el contenido de la página como respuesta, y el navegador lo muestra al usuario. Las solicitudes GET a diferencia de POST no incluyen cuerpos. (Chateka, 2017)

PUT: Actualización de recursos

El método PUT se utiliza para actualizar o modificar un recurso existente en el servidor. Al enviar una solicitud PUT, el cliente proporciona los nuevos datos del recurso en el cuerpo de la solicitud. El servidor procesa los datos y realiza la actualización correspondiente en el recurso especificado. (Chateka, 2017)

Un ejemplo práctico de uso de PUT es en la edición de contenido en una plataforma de gestión de contenido. Si un usuario desea modificar una publicación existente, el cliente enviará una solicitud PUT con los nuevos datos del contenido. El servidor actualizará la entrada en la base de datos con la información proporcionada. (Chateka, 2017)

El tipo de solicitud determina si un servidor crea un nuevo registro (POST), lo actualiza (PUT) o solo lo devuelve la información solicitada al cliente, sin actualizar el registro (GET), el método define la acción del servidor al recibir la solicitud. (Chateka, 2017)

4.1.6. Interfaz de programación de aplicaciones:

Una interfaz de programación de aplicaciones conocida en inglés como: *Application Programming Interface (API)* se puede explicar como un puerto de acceso a un sistema a través de internet. Reciben y envían información de un usuario o de otras aplicaciones de forma estructurada. Los métodos más comunes para estructurar la información enviada y recibida por la API son: JSON y XML. En otras palabras, es una interfaz que permite que diferentes programas se comuniquen y compartan datos de manera eficiente. Las API trabaja con protocolos de comunicación HTTP. Como mencionamos anteriormente se puede consultar información de una API por medio de una solicitud GET. Cada vez que un navegador ingresa a un sitio web, está realizando una solicitud de tipo GET al servidor de dicho sitio web. También es posible crear nuevas instancias por medio de una solicitud POST o actualizarlas con un PUT. Existen otras funcionalidades dentro de los protocolos de la comunicación HTTP, pero estas son las principales. (Gazarov, 2016)

Las API se utilizan en una variedad de contextos, incluyendo aplicaciones web, aplicaciones móviles, servicios en línea y sistemas empresariales. Permiten a las empresas y desarrolladores acceder a datos y funcionalidades de otros sistemas y aplicaciones, lo que permite una mayor interoperabilidad y la creación de aplicaciones más sofisticadas.

En el contexto de los ERP, tener una API abierta significa que el software está diseñado para permitir a los desarrolladores externos conectarse y acceder a sus datos y funcionalidades de forma programática. Son una puerta para poder integrar dos sistemas que normalmente no trabajan juntos.

Parámetros de un URL:

Los parámetros de un URL son conocidos como Query Strings. Un parámetro en un API es una parte de la URL que se utiliza para enviar datos al servidor. En una solicitud HTTP, un Query String se compone de uno o más pares de clave-valor separados por el carácter & al final del url al cual se realiza la consulta. Estos parámetros son recibidos por

el servidor de la API y alteran la respuesta que se obtiene.(Crymble, 2022), La estructura de los parámetros se muestra en la Figura: 2

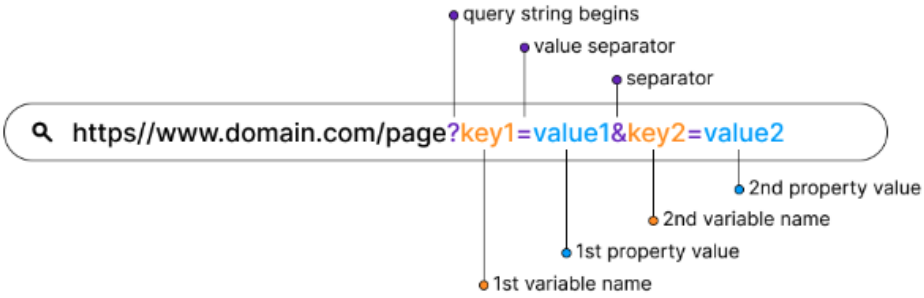


Figura 2: Ejemplo de parámetro de URL

Fuente: Ofiwe, 2021

En general, una API utiliza un conjunto de URLs que permiten a los clientes acceder a recursos específicos, y los Query Strings se utilizan para personalizar la respuesta del servidor a las solicitudes de los clientes. Por ejemplo, en una API de búsqueda, un Query String podría ser utilizado para especificar el término de búsqueda o el número de resultados que se desean recibir. El Query String en un API es una forma común de enviar datos al servidor a través de la URL de la solicitud. Proporciona una manera fácil para que los desarrolladores personalicen las respuestas del servidor a sus necesidades específicas y accedan a los recursos de una API de manera eficiente. (Crymble, 2022)

4.1.7. NoCode:

NoCode es un término que se refiere a la capacidad de crear aplicaciones y software sin la necesidad de tener conocimientos avanzados de programación. Este enfoque utiliza herramientas para permitir que los usuarios creen aplicaciones personalizadas sin tener que escribir código.

El concepto de NoCode ha existido por décadas, pero ha ganado popularidad recientemente gracias a la proliferación de herramientas de software que hacen que el proceso de crear aplicaciones sea más accesible para personas con un bajo nivel técnico en gestión de sistemas, bases de datos y programación. NoCode elimina estas barreras de entrada y permite al usuario programar al diseñar un diagrama o conectando bloques. Algunas de las herramientas de NoCode más populares incluyen Airtable, Bubble, Webflow, Glide, y Zapier.

Una de las principales ventajas de NoCode es que permite a los usuarios crear aplicaciones complejas rápidamente y con menos costo en comparación con el desarrollo de software tradicional. Además, las herramientas de NoCode son intuitivas y fáciles de usar, lo que significa que cualquier persona puede crear aplicaciones personalizadas sin tener que pasar por un largo proceso de aprendizaje.

NoCode es una excelente herramienta para crear y probar aplicaciones ya que permite

un desarrollo rápido y de bajo costo. Esto permite poner a prueba la hipótesis de solución al usuario que propone el software sin invertir meses o años en el desarrollo. Es ideal para prototipado rápido. (Khella, 2020)

4.1.8. Zapier

Zapier es una plataforma de automatización SaaS que permite a los usuarios conectar diferentes aplicaciones web y crear flujos de trabajo automatizados, sin necesidad de escribir código o tener conocimientos de programación. (Martinez, 2020) Para escribir un programa en esta plataforma se dibuja un diagrama de flujo como se muestra en la Figura: 3

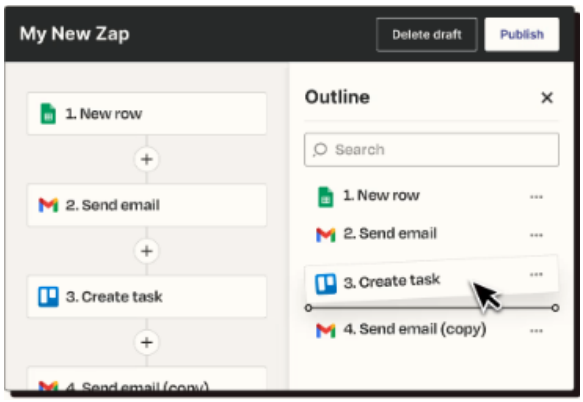


Figura 3: Ejemplo de programación con Zapier

Fuente: Martinez, 2020

Los usuarios pueden crear "zaps"(automatizaciones) que conectan dos o más aplicaciones y realizan acciones específicas en función de un disparador que inicializa el programa para luego correr una serie de acciones que pueden tomar diferentes caminos, contener secuencias repetitivas y filtros. Se integra con más de 4,000 aplicaciones web SaaS. Es posible integrar Zapier con aplicaciones que no se encuentran en su directorio aún ya que cuenta con una aplicación interna llamada: webhooks by zapier, que permite comunicación con servidores HTTP y API´s. Otra de las aplicaciones internas desarrolladas por zapier es llamada "Code Step" esta aplicación permite correr programas de Python o Javascript de forma headless, permite ejecutar aplicaciones que no tardan más de 30 segundos en ejecutarse. (Robertson, 2022)

Zapier se utiliza comúnmente para automatizar tareas repetitivas y ahorrar tiempo, mejorar la eficiencia de los procesos de trabajo y aumentar la productividad. La plataforma es especialmente útil para pequeñas y medianas empresas que no tienen los recursos para desarrollar su propia solución de automatización. Además, Zapier ofrece una amplia variedad de planes de precios para adaptarse a diferentes necesidades y presupuestos.(Herman, 2022)

4.1.9. Servicios de computación en la nube

En el panorama actual de la tecnología, los servicios de computación en la nube han revolucionado la forma en que las empresas y los desarrolladores crean, implementan y gestionan sus aplicaciones. Amazon Web Services (AWS) se ha posicionado como uno de los proveedores líderes en este campo. En esta investigación, exploraremos qué es AWS, su utilidad, y nos centraremos específicamente en las funciones Lambda, su definición y cómo se ejecutan en Python. AWS es una plataforma de computación en la nube ofrecida por Amazon.com. Proporciona una amplia gama de servicios, incluyendo almacenamiento, análisis de datos, inteligencia artificial, aprendizaje automático, bases de datos y más. Estos servicios se ofrecen en forma de bloques de construcción, permitiendo a las organizaciones utilizar solo los recursos que necesitan, de forma escalable y bajo demanda. AWS permite a los usuarios acceder a una infraestructura robusta y confiable sin incurrir en altos costos iniciales.

Dentro del conjunto de servicios que ofrece AWS, encontramos las funciones Lambda. Las funciones Lambda son una forma de ejecutar código sin tener que aprovisionar ni administrar servidores. Son unidades de código que se ejecutan en respuesta a eventos específicos y se escalan automáticamente según la carga de trabajo. Las funciones Lambda siguen el modelo de çomputación sin servidor"(serverless computing), donde los desarrolladores se centran en escribir código y dejar que AWS se encargue de la infraestructura subyacente. (Kononenko, 2018) AWS Lambda admite múltiples lenguajes de programación, incluyendo Python. Para ejecutar código Python en una función Lambda, se deben seguir los siguientes pasos:

a. Crear una función Lambda: En la consola de AWS o mediante la API de AWS, se crea una función Lambda y se define un nombre, lenguaje de programación y configuraciones específicas.

b. Escribir el código Python: Se debe escribir el código Python que se desea ejecutar dentro de la función Lambda. Este código puede incluir cualquier biblioteca o módulo necesario.

c. Configurar los desencadenantes (triggers): Seleccionar el evento o acción que desencadenará la ejecución de la función Lambda. Puede ser un evento de una aplicación web, una actualización en una base de datos, un mensaje en una cola, entre otros.

d. Configurar la función Lambda: Establecer los límites de recursos, como la memoria y el tiempo de ejecución máximo, así como cualquier otra configuración específica requerida para la función.

e. Prueba y despliegue: Una vez configurada, la función Lambda puede probarse localmente antes de ser desplegada en la nube. AWS proporciona herramientas para simular eventos y probar la función.

Mediante el uso de servicios como las funciones Lambda, los desarrolladores pueden ejecutar código sin preocuparse por la infraestructura subyacente. Python, como uno de los lenguajes compatibles, permite a los desarrolladores aprovechar las ventajas de AWS Lambda y construir aplicaciones escalables y eficientes. (Kononenko, 2018)

4.2. Componentes electrónicos

4.2.1. Placa de desarrollo ESP2866

El ESP8266 es un microcontrolador compacto de bajo costo con Wi-Fi integrado que se ha vuelto popular en el mundo de la electrónica y la IoT (Internet de las cosas). Desarrollado por la empresa china Espressif Systems. El ESP8266 es compatible con el IDE de Arduino, MicroPython y NodeMCU, lo que hace que sea fácil de programar y de integrar en proyectos de IoT. Cuenta con una gran cantidad de librerías desarrolladas por la comunidad de makers. Contiene 16 entradas y salidas digitales y una entrada analógica. Al ser un chip con Wi-Fi este puede comunicarse con el internet haciendo llamados HTTP a servidores. Se caracteriza por su tamaño compacto y bajo costo. (Allan, 2018)

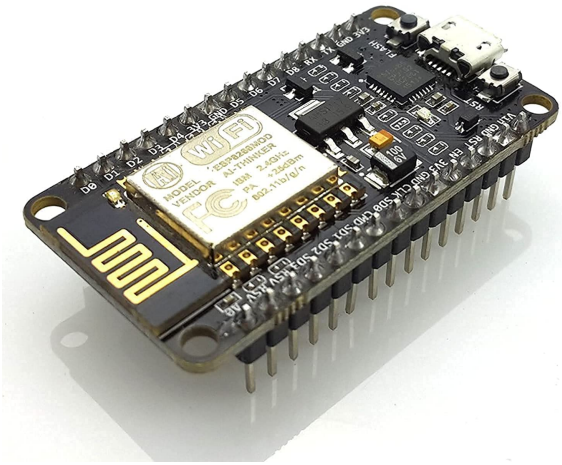


Figura 4: Placa de desarrollo ESP8266 Node MCU

Fuente: Allan, 2018

El chip ESP8266 es ampliamente utilizado en proyectos de IoT, como sensores de temperatura y humedad, interruptores inteligentes, sistemas de monitoreo remoto y mucho más. Los nombres de las entradas digitales y puertos del ESP8266 difieren de la nomenclatura del IDE de arduino. Por lo que se debe utilizar el Cuadro 1 para convertir los nombres utilizados por el ESP8266 a los nombres de dichos puertos al programar la placa utilizando el IDE de Arduino.

4.2.2. Placa de desarollo ESP32

El ESP32 destaca por su mayor potencia de procesamiento al compararla con su contraparte, el ES8266 esto gracias a su procesador de doble núcleo, lo que permite manejar tareas más complejas y exigentes en términos de cómputo. Además, su mayor capacidad de

Cuadro 1: Nomenclatura de entradas del ESP8266 en IDE de Arduino

Puerto	**Nombre**	**Equivalente en Arduino**
AO	AO	AO
DO	GPIO 16	16
D1	GPIO 5	5
D2	GPIO 4	4
D3	GPIO 0	0
D4	GPIO 2	2
D5	GPIO 14	14
D6	GPIO 12	12

memoria RAM y flash brinda más espacio para el almacenamiento de datos y la ejecución de aplicaciones más robustas.

En cuanto a la conectividad, tanto el ESP32 como el ESP8266 ofrecen soporte para Wi-Fi, lo que les permite conectarse a redes inalámbricas y facilitar la comunicación con otros dispositivos. Sin embargo, el ESP32 también cuenta con soporte para Bluetooth, lo que amplía aún más sus capacidades de conectividad y abre la puerta a una gama más amplia de aplicaciones.

Por otro lado, el ESP8266 sigue siendo una opción viable para proyectos de IoT más simples o de menor escala. Su menor costo y consumo de energía pueden ser factores determinantes en ciertos casos. Además, el ESP8266 ha sido ampliamente adoptado y cuenta con una comunidad de desarrolladores activa, lo que brinda un amplio soporte y documentación disponible. (Allan, 2018)

La elección entre el ESP32 y el ESP8266 dependerá de las necesidades específicas de cada proyecto. Si se requiere mayor potencia de procesamiento, capacidad de memoria y soporte adicional para Bluetooth, el ESP32 es la opción preferida. Sin embargo, si se busca una solución más económica y se tiene en cuenta la simplicidad del proyecto, el ESP8266 sigue siendo una opción sólida. Ambas plataformas ofrecen una excelente relación calidad-precio y son ideales para impulsar la innovación en el ámbito del Internet de las Cosas.

4.2.3. Lector de código de barras DE2120 SparkFun

Existen miles de lectores de códigos de barras. Para comunicarse con el microcontrolador ESP8266 es necesario un lector de códigos de barra con protocolo de comunicación TTL. El lector de código de barras 2D de Sparkfun Electronics modelo: DE2120, es un módulo compacto y fácil de usar que permite la lectura de códigos de barras 1D y 2D. El dispositivo utiliza el sensor de imagen SE4500 de Motorola para leer los códigos de barras, lo que le permite ser preciso y rápido en la lectura. Este dispositivo es compacto y puede ser integrado en proyectos electrónicos de reducido tamaño como se muestra en la Figura 5

Una de las razones por las que este lector de códigos de barras es ideal para su uso con un microcontrolador como el ESP8266 es su capacidad para funcionar con una variedad de interfaces de comunicación, incluyendo UART, TTL, USB y RS-232. Esto permite al lector

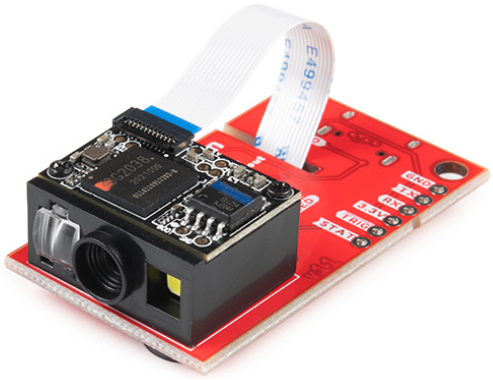


Figura 5: Lector de códigos de barra Sparkfun DE2120

Fuente: *2D Barcode Scanner Setting Manual*, 2019

de códigos de barras comunicarse fácilmente con el microcontrolador, lo que lo hace perfecto para proyectos de automatización de procesos o control de inventario.

Otra carácteristica deseable de este escáner es la capacidad de activarse el detectar movimiento. La configuración de este lector se hace por medio de un manual de configuración. Este manual cotiene una serie de códigos de barra con combinaciones reservadas para configurar el escáner.(*2D Barcode Scanner Setting Manual*, 2019)

Además, el lector de códigos de barras de SparkFun cuenta con una amplia documentación y biblioteca de software disponible en línea, lo que lo hace fácil de integrar en proyectos de programación. El dispositivo también es muy fácil de usar, ya que simplemente se conecta al microcontrolador a través de uno de los puertos disponibles y se configura para leer los códigos de barras.

4.2.4. Conexión de piezas impresas por ajuste a presión

La conexión de ajuste por presión se emplea habitualmente en la impresión 3D de piezas mecánicas, tales como engranajes, levas y ejes. Uno de los principales retos en la impresión 3D es establecer una conexión sólida y estable entre piezas impresas, que permita su ensamblaje sin necesidad de usar herramientas o adhesivos adicionales. Esta es la principal ventaja de este método de sujeción, no requiere de más piezas o pegamento para funcionar, depende únicamente de la geometría de las dos piezas que encajan entre sí. (Bayer, 2011) La conexión por ajuste a presión permite un encaje preciso y firme, mediante la creación de una tolerancia ajustada entre las piezas que se unen, de modo que puedan encajar entre sí sin movimiento excesivo y sin quedar demasiado ajustadas.

En la conexión de ajuste por presión, las piezas impresas se diseñan con características mecánicas que les permiten encajar entre sí de forma segura. Durante el encaje estas se

deforman temporalmente, dentro de su zona elástica. Esta deformación provoca una fuerza reacción en la pieza, que se utiliza para fijar el componente con el que encaja.(Carolo, 2022) Cuando se diseña correctamente, una junta por presión puede ser una unión más fuerte qué el pegamento epóxico. Estas juntas pueden ser en forma de pestañas, lengüetas como se muestra en la Figura: 6.

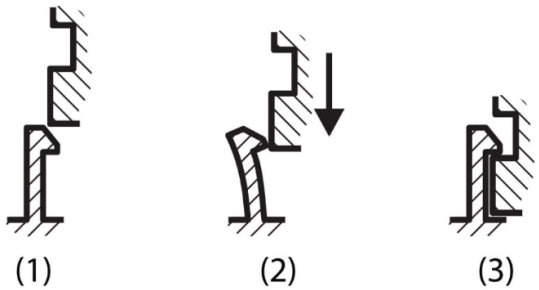


Figura 6: Conexión por ajuste a presión

Fuente: Carolo, 2022

El encaje se logra mediante la presión ejercida por la deformación elástica de los materiales durante el proceso de encaje. (Carolo, 2022) La precisión y la rigidez de la conexión son cruciales para garantizar el correcto funcionamiento de la pieza en su conjunto. Además, la conexión de ajuste por presión se utiliza para piezas que soportan cargas y esfuerzos, como marcos o soportes.

CAPÍTULO 5

Metodología

Para desarrollar el proyecto se utilizó la metodología de desarrollo incremental y diseño de la ingeniería. La primera etapa consistió en recopilar información de la condición actual del problema. Luego se conceptualizó un diagrama de flujo del proceso propuesto como solución. Se construyeron una serie de prototipos e iteraciones en los sistemas para llegar al diseño definitivo. Este último prototipo se instaló en las instalaciones de bodega y se capacitó a los operarios sobre su uso. Durante 30 días se operaron todos los despachos usando el dispositivo. Por último, se midió la reducción en reclamos y devoluciones por despachos incorrectos. Para hacer esta medición se exportaron las devoluciones recibidas durante el periodo con el que se operó con el dispositivo. Se compararon los datos para medir los resultados.

5.1. Medición de condición actual

La primera etapa consistió en recopilar información de los despachos incorrectos. La empresa recibe los reclamos por despachos incorrectos por medio de su tienda en línea. La tienda en línea cuenta con una sección de devoluciones donde el cliente indica el motivo de la devolución. Para medir la condición actual, se exportaron estos casos a formato CSV, se obtuvo una muestra de 144 devoluciones de un mes, esta se muestra en el Cuadro 2.

Como se muestra en el cuadro, de las 144 devoluciones se tienen 16 despachos enviados incorrectamente. Es decir el 8 % de las devoluciones son originadas por despachos incorrectos. Celovendo estima enviar 3,000 órdenes mensuales. La tasa global de devoluciones es cercana a 5 %. Es decir, por cada 100 órdenes habrá 5 devoluciones. Las causas de estas devoluciones son varias y se muestran en el Cuadro: 2 De esta muestra de 144 devoluciones, 16 se originaron por despachos incorrectos.

Se considera que esta información es una muestra significativa, ya que cumple con los

Cuadro 2: Cantidad de devoluciones y sus razones, en un periodo de 30 dias.

Razón de Devolución	**Cantidad**	**Porcentaje**
Me enviaron el producto equivocado	16	11 %
Pedí el producto equivocado	25	17 %
Producto defectuoso	92	64 %
Ya no necesito el producto	11	8 %
Total	144	100 %

criterios de tamaño de muestra de una población finita y satisface la ecuación 1 donde el tamaño de muestra es 2880 (el número de órdenes en el periodo observado), con un nivel de confianza del 95 % y un margen de error del 8 %.

$$n = \frac{N \cdot z^2 \cdot p \cdot (1-p)}{(N-1) \cdot e^2 + z^2 \cdot p \cdot (1-p)} = 143$$

2: Ecuación para determinar el tamaño de una muestra con población finita.

Donde N es el tamaño de la población. Z es el nivel estadístico de confianza. E es el error de estimación. p es la probabilidad de éxito y q la probabilidad de fracaso. Los costos directos por cada despacho incorrecto se muestran en el Cuadro: 3

Cuadro 3: Costos directos por cada despacho incorrecto.

Descripción	**Costo**
Envió de devolución	GTQ 30.00
Envió de despacho correcto	GTQ 30.00
Material de empaque	GTQ 3.00
Refacturación	GTQ 0.10
Total	GTQ 63.10

El costo mensual de despachos incorrectos se obtiene al multiplicar la cantidad de despachos incorrectos por los costos directos de cada despacho incorrecto. Asumiendo 3,000 ordenes al mes, con un 5 % de devoluciones, de las cuales el 11 % corresponde a despachos incorrectos, esto nos da un costo directo mensual de Q.1,072.70 y anual de Q12,872.4. Este cálculo no toma en cuenta el costo intangible de insatisfacción del cliente. Cada vez que un cliente recibe una orden de forma incorrecta aumenta su nivel de insatisfacción y reduce la probabilidad de recompra.

5.2. Diseño de sistema

El nuevo proceso de despacho de órdenes de venta propone una etapa de verificación automatizada, como se muestra en la Figura 7. Se inicia el proceso de diseño con una conceptualización general del funcionamiento del dispositivo y como este se comunicará con el ERP de la empresa.

El proceso implementado está dividido en 4 etapas:

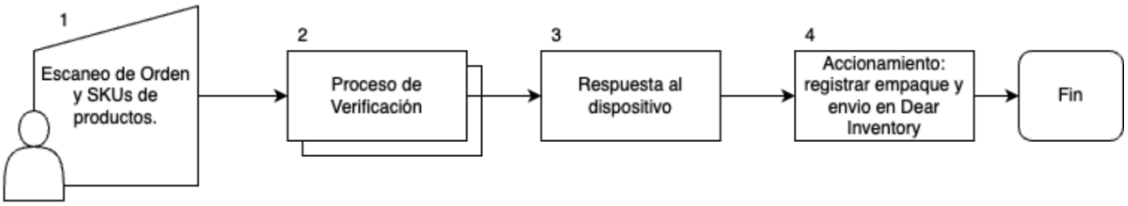


Figura 7: Proceso de despacho con verificación automatizada.

Fuente: Elaboración propia.

- **1. Digitalización**: El ERP necesita identificar la orden sobre la cual se está operando el despacho. El operador debe escanear el número de orden que está despachando. Para ello escanea la lista de recolección, la cual se le imprime automáticamente en su estación de trabajo cada vez que recibe una orden. Esta lista contiene un código de barras en la parte superior con el número de identificación del pedido. Este número es único e irrepetible. Luego el operador debe escanear todos los artículos recolectados mientras realiza el empaque de la orden. Este proceso garantiza que todos los productos que se empaquen se escanean y verifican.

- **2. Proceso de verificación**: se explica a detalle en la siguiente sección, ocurre en su totalidad en la nube. No requiera acción del operario. Consiste en verificar si la orden existe en el ERP y luego verificar que los artículos escaneados corresponden exactamente a los ordenados que se encuentran disponibles, comparando ambas listas. Esta verificación se ejecuta en un *Code Step* de *Python* en Zapier.

- **3. Respuesta al dispositivo**: inmediatamente después de correr la verificación de la etapa 2 del proceso descrito en la Figura: 7, existen tres posibles respuestas al dispositivo despachador detalladas a continuación:

  - **La orden no existe**: la orden no se puede encontrar en el sistema de la empresa. Por lo que se envía una respuesta de alerta al operario, indicando que no se pudo encontrar la orden. Al suceder esto se detiene el flujo y no se registra ningún despacho.
  - **El despacho es correcto**: si el despacho se realizó correctamente, se opera el despacho en el ERP y se le notifica al operario que él despachó fue correcto.
  - **El despacho es incorrecto:** se muestra el error en la pantalla del dispositivo, se emite una alarma auditiva y se enciende una luz que notifica al operario del error en el despacho. La alarma auditiva no se detiene hasta que el operador presiona un botón, confirmando que percibió la alarma.

  Si la orden no se encontró o está incorrecta, se encenderá una alarma que emitirá un sonido que alerte al operador, así mismo requiere que el operador presione un botón para silenciar la alarma, confirmando que está consciente del error.

- **4. Accionamiento**: si el resultado de la verificación indica que este fue correcto, se procede a registrar el despacho en el ERP de la empresa, completando así el proceso de venta.

5.2.1. Definicion de funciones del sistema

El sistema está dividido en una sección que corre en el microcontrolador y otra que corre en la nube, en la plataforma de Zapier. A continuación se explica la función y responsabilidad de cada sección.

- **Sección en microcontrolador**: El sistema que corre en el microcontrolador es programado en C++ a través del *IDE de Arduino*. Es responsable de recopilar la información de la orden y desplegar al operario el resultado del despacho. Está compuesto en sí por dos sistemas, el despachador; responsable de recopilar el número de orden y códigos *sku (stock keeping unit)* escaneados por el operario. Una vez el operario termina el despacho, presiona un botón que llama a la sección que corre en la nube, por medio de una solicitud de tipo GET, pasando como parámetros del URL los valores escaneados. Asimismo, el notificador, despliega el resultado luego de evaluar el despacho en la nube. Ninguno de los dispositivos realiza algún tipo de análisis sobre la información escaneada durante el proceso de despacho, dejando este análisis a la sección que corre en la nube. Se diseñó de esta forma para agilizar el tiempo de respuesta de la evaluación. En las primeras iteraciones se intentó correr el análisis y comparación entre la orden y el despacho directamente en el microcontrolador. Es posible hacerlo, pero el tiempo de respuesta era de varios minutos, debido a la capacidad limitada de los microcontroladores utilizados. Es necesaria una respuesta rápida al operador para no crear un cuello de botella en el proceso de ventas. Por esta razón se eligió esta arquitectura híbrida. La arquitectura hibrida también permitió iterar el sistema si necesidad de conectarse al dispositivo para hacer los cambios. Esto agilizó el desarrollo.

- **Sección en la nube**: Está compuesta por una rutina en Zapier, conocida como un *Zap*. Un *Zap* es un programa lineal que inicia a partir de un disparador y corre una serie de etapas. Antes de hacer el *Zap* se diseñó el proceso con el diagrama de flujo mostrado en la Figura: 8. Como se observa en el diagrama de flujo, el disparador que activa esta rutina es el llamado del URL, por parte del despachador, cuando el operario presiona el botón de despachar. Este sistema se encarga de evaluar y comparar el despacho contra la orden en el ERP. Para evaluar la orden, utiliza una etapa llamada *Code Step by Zapier*, donde se corre una rutina en Python que consulta la orden en el ERP y verifica que el despacho sea correcto. Esta rutina envía la respuesta a la sección del notificador, informando así al operario del resultado de la operación como se describió anteriormente. Solo si el despacho es correcto, registra el despacho y completa la orden en el ERP.

Cada etapa del proceso de la Figura: 8 representa un bloque en la rutina (*Zap*) de Zapier. Esta rutina contiene las secciones de código en Python responsables de evaluar y comparar el despacho del operario contra la orden. El funcionamiento de estos bloques de código se explica a detalle en la sección 5.24.

5.2.2. Selección de componentes

Los componentes se seleccionaron con base a los requisitos de diseño y funciones esperadas. Para programar el dispositivo se seleccionó el microcontrolador ESP826 NodeMCU, debido a que cuenta con un módulo wifi incorporado. Esto lo hace ideal para construir un dispositivo de internet de las cosas. El ESP8266 es capaz de correr la librería de Arduino

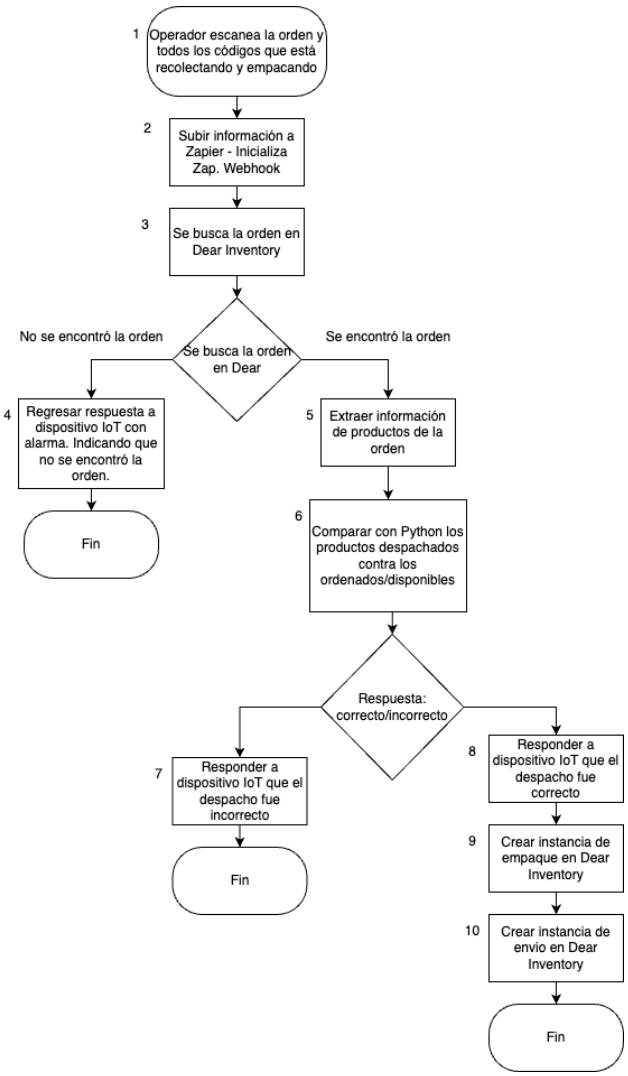


Figura 8: Diagrama de flujo de información para verificar despacho.

Fuente: Elaboración propia.

"Wifi Manager", la cual realiza solicitudes a la web de tipo *GET, PUT ó POST*. Asimismo, este microcontrolador cuenta amplia comunidad de desarrolladores, lo que facilitó la implementación del sistema, ya que existe una gran cantidad de ejemplos, librerías y documentación sobre la programación de este microcontrolador usando el IDE de Arduino. La principal desventaja de este dispositivo es que solo ofrece un voltaje de salida de 3.3 voltios. Por lo que todos los componentes deben funcionar con este voltaje no estándar. Todos los demás componentes se seleccionaron tomando en cuenta que deben funcionar con el voltaje de salida de la placa. Otra desventaja que se tomó en consideración es que el ESP8266 solo cuenta once entradas digitales y una entrada análoga, se diseñó el sistema tomando en cuenta estas limitaciones.

El dispositivo cuenta con dos placas de desarrollo ESP8266 NodeMCU, uno en la sección del escáner y el segundo en la sección del notificador. Por lo que en realidad son dos dispositivos unidos. En las primeras iteraciones del dispositivo se tenían ambas funciones en

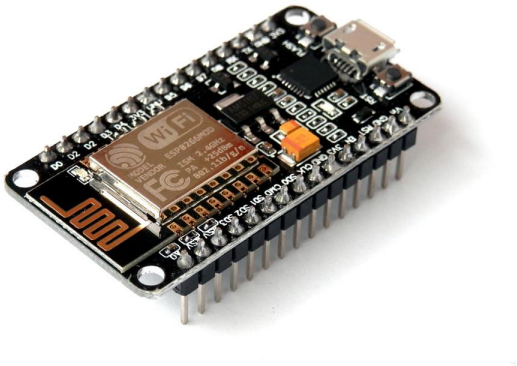


Figura 9: Placa de desarrollo *ESP8266 NodeMCU*

Fuente: Allan, 2018

un solo microcontrolador, debido a la cantidad limitada de entradas y salidas digitales del microcontrolador y con el objetivo de hacer más rápida la operación y tiempo de respuesta. Se separó en dos microcontroladores. El dispositivo está dividido en dos, la sección del escáner y la función del notificador.

- **Sección de escáner**: es la sección con la que el operador interactúa para realizar cada despacho. Debe escanear la orden y los productos que está empacado. Para seleccionar el modo de escaneo, cuenta con un potenciómetro. Con este, el operador puede seleccionar uno de los siguientes tres modos:

  - **Número de orden**: al estar seleccionado este modo, al escanear un código de barras, este se guardará a la variable que almacena el número de orden.
  - **SKUS**: al estar seleccionado este modo, se concatena y almacena el valor del código sku, la variable que almacena la lista de códigos sku.
  - **Números de parte**: al estar seleccionado este modo, se concatena y almacenan los números de parte en la variable que almacena la lista.

  Los modos de operación se seleccionan leyendo los valores de la entrada analógica. Los valores escaneados por el operador se muestran en una pantalla LCD de 32 caracteres, dividida en dos líneas. La pantalla LCD seleccionada es una LCD de 16X2 con módulo I2C. Se seleccionó esta pantalla ya que funciona con los 3.3 voltios proporcionados por el ESP8266, asimismo se puede conectar con la placa utilizando solo cuatro conexiones: tierra, voltaje, SCL, SDA, utilizando así solo dos entradas digitales.

  El escáner de códigos de barra seleccionado fue el DE2120 de la marca SparkFun. Este escáner opera con 3.3 voltios. Cuenta con el protocolo de comunicación TTL (*Transistor-Transistor Logic*) por lo que es posible conectarlo directamente a las entradas digitales tx y rx de la placa ESP8266 NodeMCU seleccionada. Este escáner es rápido, con la capacidad de escanear hasta 200 códigos de barra por minuto, por lo que no creará un cuello de botella en el proceso de ventas. Sus dimensiones reducidas y capacidad de comunicación TTL lo hacen ideal para integrarse con una placa de desarrollo ESP9266 NodeMCU.

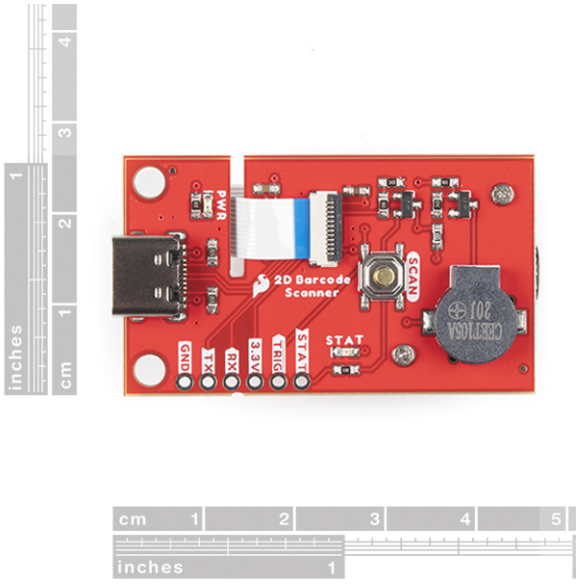


Figura 10: Lector de códigos de barra Sparkfun DE2120

Fuente: *2D Barcode Scanner Setting Manual*, 2019

Por último, esta sección cuenta con dos botones. Se dispone de un botón para eliminar los valores escaneados, en el caso de que se cometa algún error al escanear los códigos y se identifique de inmediato durante el proceso de despacho por parte del operario. Cuando el operador termina de escanear todos los artículos de la orden, presiona el botón de despacho que sube a la nube y dispara el flujo de verificación. Ambos botones son pulsadores, normalmente abiertos.

- **Sección de notificador**: el principal objetivo de esta sección del dispositivo es notificar al operario en caso de que exista un error durante el proceso de despacho. Por esta razón se equipó con un zumbador que produce un sonido de alerta de 60 decibeles al ser administrado 3.3 voltios. Cuenta con una placa *ESP8266 NodeMcu* y una pantalla *LCD I2C de 16X2* idénticas a la sección del escáner. Asimismo cuenta con dos luces LED, todos estos son medios para alertar al operador en caso de error. Esta sección del dispositivo Cuenta con tres modos de operación:

  - **Despacho correcto:** Se activa cuando el dispositivo recibe la respuesta de la nube indicando que el despacho fue correcto. Enciende una luz verde, muestra en pantalla un mensaje que le indica al operador que el despacho fue correcto y emite un sonido breve que indica de forma auditiva qué el despacho fue correcto.
  - **Despacho incorrecto:** Notifica con una alarma al operario, en donde se enciende el zumbador, una luz *LED* de color rojo y se muestra en pantalla el número de pedido indicando que el despacho fue incorrecto. El sonido emitido por el zumbador no se detiene hasta que el operario presiona un botón. Reconociendo qué escuchó la alarma.
  - **Orden no encontrada:** Notifica con el mismo tipo de alarma qué una orden incorrecta, pero muestra en pantalla que la razón del error fue qué no se encontró el número de pedido en el ERP de la empresa.

La activación de cualquiera de estas tres etapas depende del flujo de verificación que corre en la nube en la plataforma *Zapier*.

Los componentes seleccionados son de bajo costo y distribución amplia en Guatemala. Con la excepción del escáner de códigos de barra DE2120. Este no está a la venta en Guatemala, por lo que se importó directamente de la marca *Sparkfun*.

5.2.3. Diseño y ensamble del sistema electrónico

La construcción del dispositivo fue incremental e iterativa, se realizó por etapas para comprobar la viabilidad de cada una. Como primer hito para comprobar la viabilidad del proyecto se construyó un prototipo que integrara el escáner de códigos de barra DE2120 con la placa *ESP8266 NodeMCU* con una pantalla LCD. El objetivo de este prototipo fue capturar un código de barras y mostrarlo en pantalla.

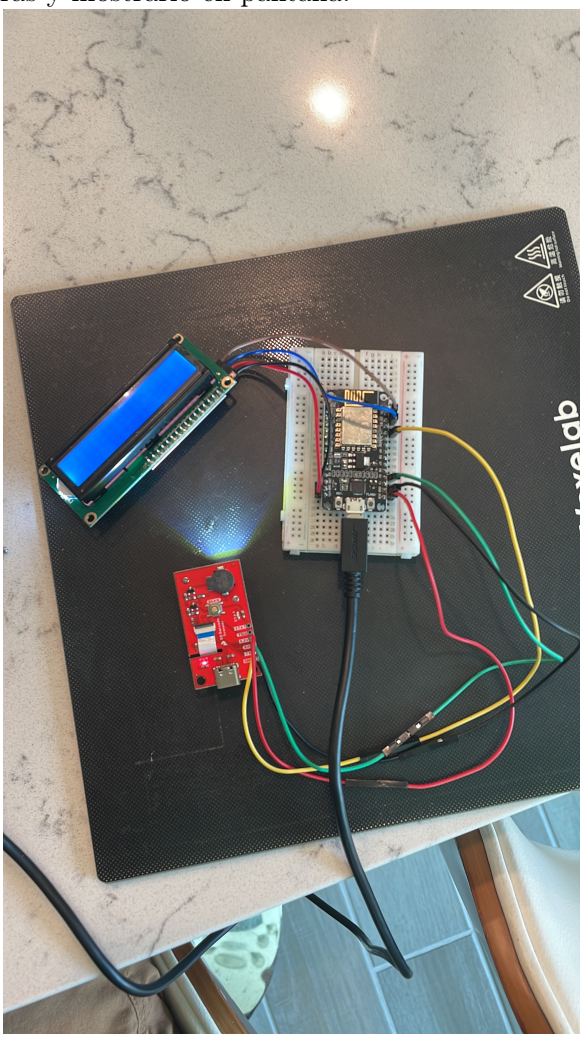


Figura 11: Primera iteración del dispositivo

La firgura 11 muestra la conexión de los componentes de la priemra iteración. Esta primera iteración fue exitosa, se logró configurar el escáner utilizando el manual de ajustes para hacerlo entrar en el protocolo de comunicación TTL, de igual forma se configuró el escáner para activarse al detectar movimiento, por lo que ya no requiere que el operario presione un botón para crear una lectura.

El segundo hito del proyecto consistió en subir la información escaneada a una API por medio de una consulta de tipo *GET*, pasando como parámetro del URL el código de barras escaneado. La placa ESP266 *NodeMCU* captura los códigos de barra, un carácter a la vez. Cada carácter se concatena a una variable de tipo texto, cuando termina, genera la consulta *GET* a la web.

Figura 12: Código C++ para concatenar y subir información

```
if (scanner.readBarcode(scanBuffer, BUFFER_LEN)) {
    for (int i = 0; i < strlen(scanBuffer); i++)
      pedido = pedido + String(scanBuffer[i]);
      pedido.trim();
      lcd.print(pedido);
    url = url + pedido;
    client.print(String("GET ") + url + " HTTP/1.1\r\n" + "Host: " +
        host + "\r\n" + "Connection: close\r\n\r\n");
```

Como se muestra en la sección de código anterior, cada carácter recibido del escáner se recibe de una cola *ScanBuffer* estos se concatenan al *URL* del API, que recibirá la información del despacho. En esta iteración, cuando el escáner termina de enviar los caracteres, el ESP8266 sube a la API el código escaneado. Cada vez que se escanea un código, este se sube a la API. No existe un botón para disparar el accionamiento del API de forma manual. A partir de este diseño se construyó otro proyecto que consistió un escáner simple para automatizar tareas que solo requerían del número de pedido como entrada.

Para poder registrar y evaluar el despacho de una orden, la empresa necesita recibir el número de orden qué se está escaneando, el código *SKU* y adicionalmente la empresa almacena números de parte enviados, para llevar un control de las garantías y devoluciones.

En la siguiente iteración del diseño se agregó un botón y un potenciómetro. El botón sirve como disparador del API, cuando este se presiona, la placa realiza una consulta de tipo *GET* pasando como parámetros del URL las variables recibidas. La perilla se utiliza para seleccionar el tipo de variable que se está escaneando: número de orden, *SKU* o número de parte. El operador empleará el potenciómetro para seleccionar el modo de operación. El microcontrolador recibe una señal analógica que oscila desde 0 hasta 1023. De esta forma se estructura la información enviada a la API. Este rango se dividió en 3 partes iguales, para seleccionar así el nombre de la variable escaneada. Por último, se volvió aparente la necesidad de un botón de borrar para el dispositivo. Debido a que no había forma de volver a empezar la rutina sin presionar el botón de "despachar". Por lo que se incorporó un botón para borrar los campos registrados. Este botón borra todos los campos de la variable. Por ejemplo, si se presiona borrar luego de haber escaneado 3 códigos *SKU*, los tres códigos se borran y se debe empezar a registrarlos de nuevo. Con esto se completó la sección del despachador del dispositivo.

Se procedió a diseñar la sección de notificador, este consiste en otra placa de desarrollo ESP8266 con una pantalla *LCD* idéntica a la del despachador, una bocina tipo *buzzer*, un botón pulsador y dos focos *LED* para notificar el estado del despacho. Esta sección se conectó a la alimentación de 5 voltios de salida del módulo despachador. De esta forma, todo el dispositivo solo requiere de un cable USB para alimentar ambos ESP8266. El notificador opera independiente al despachador, solo comparten la fuente de energía. La Figura 13 muestra el diagrama de conexión de la última iteración del dispositivo.

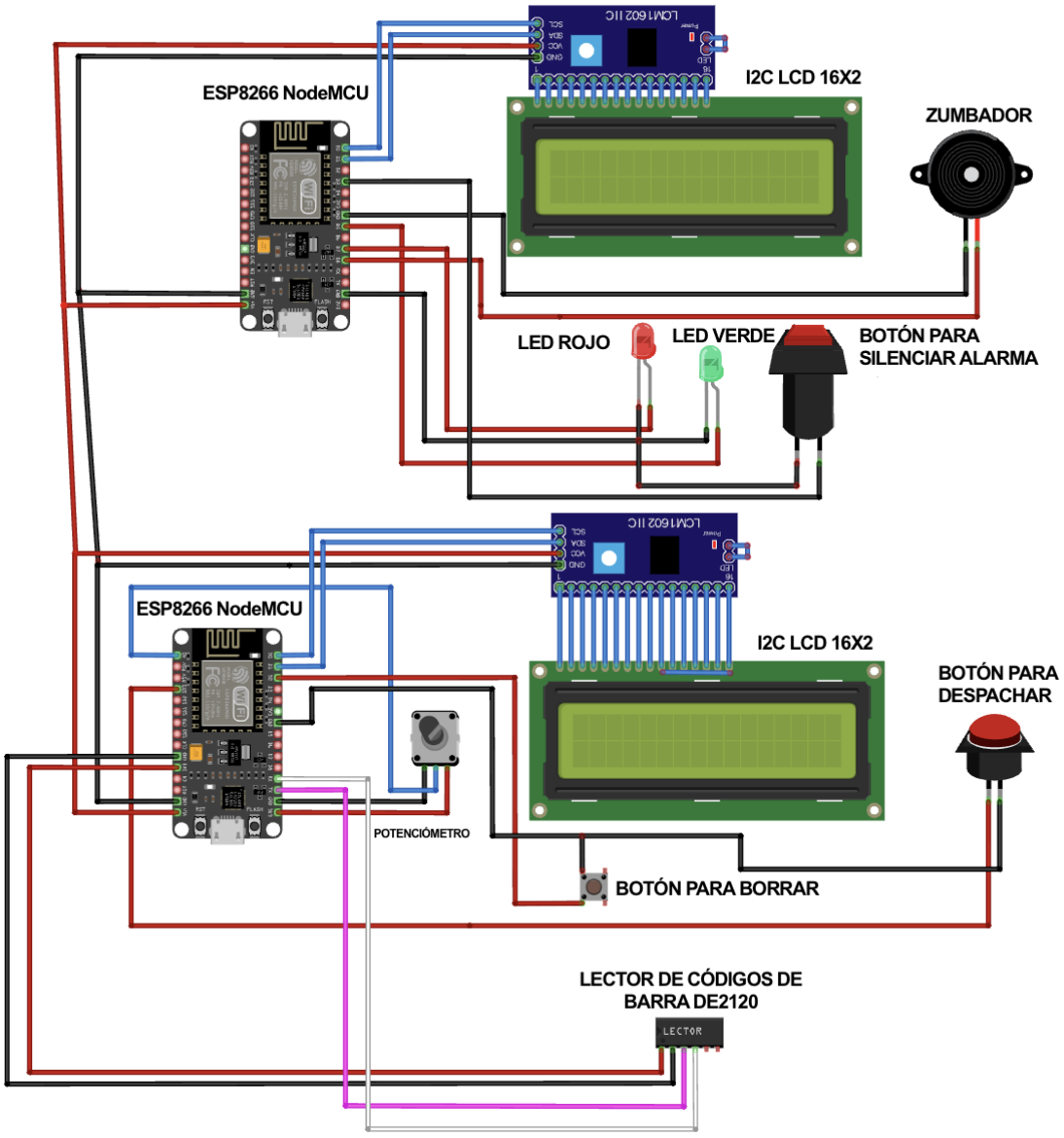


Figura 13: Diagrama de conexiones de última iteración

El código que corre en ambos microcontroladores se muestra en los anexos de este trabajo.

5.2.4. Diseño de código en la nube

Como se mencionó anteriormente, la función del dispositivo es únicamente la de transmitir y recibir información. El proceso de verificación de la orden sucede en la nube en la plataforma Zapier. Una vez el operador completa el proceso de registrar los códigos y presiona el botón de despachar. El ESP8266 realiza una consulta HTTP de tipo *GET* a una API de Zapier. Pasando como parámetros del URL los valores registrados. A partir de ese momento la nube corre el flujo qué ese muestra en la Figura 14 y 15. Estas dos figuras combinadas son el flujo en Zapier representado inicialmente en la Figura 8. Esto debido a que los programas en Zapier funcionan como un flujo accionado por un disparador, seguido por una serie de acciones. El disparador en Zapier es una etapa de *Webhook* que recibe los pa-

rámetros de URL cuando el operador presiona el botón de despachar. La Figura 15 muestra el flujo del sistema que opera en Zapier. La primera etapa nombrada: 1. *Webhook* - Recibir información, captura los parámetros de URL del dispositivo. Los parámetros transmitidos son los códigos *SKU*, números de parte y números de orden. Estos valores se encuentran separados por comas. El programa inicia recibiendo estos tres vectores de una dimensión.

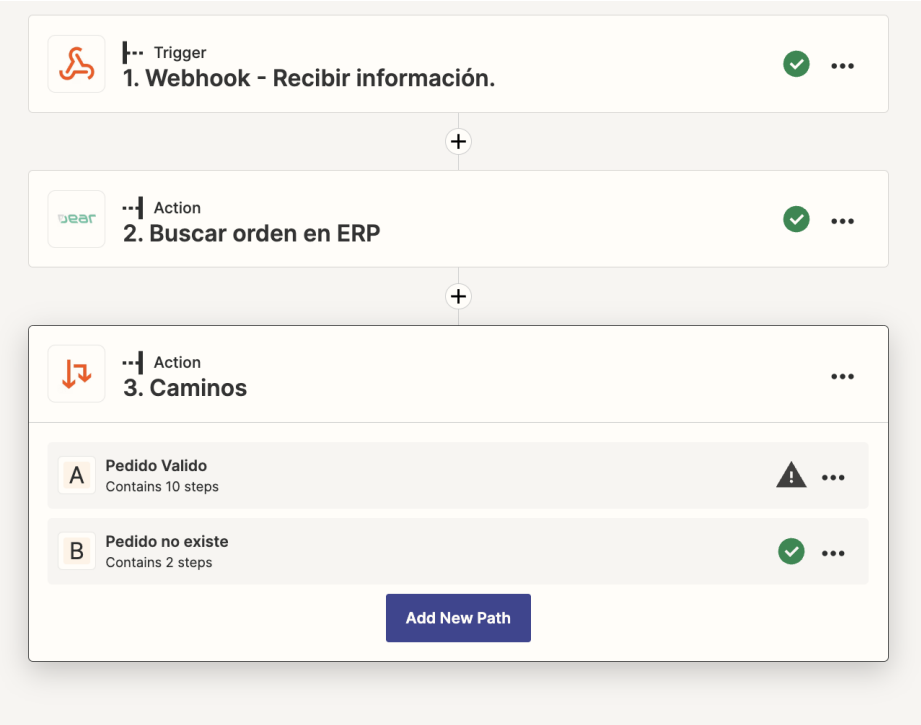


Figura 14: Flujo de sistema en Zapier

La etapa más significativa de este procedimiento es la etapa seis, la cual se evidencia tanto en la Figura 8 como en la Figura 15 con el número seis. Ya que es ahí donde se verifica si el despacho es correcto o incorrecto. Esta etapa se denomina verificación de despacho, la cual se explica detalladamente en la siguiente sección. Se trata de una rutina de *Python* que compara los artículos despachados con la orden. El código de esta rutina se encuentra disponible en la sección de anexos. Luego de la comprobación, el notificador recibe la respuesta del proceso de revisión. Como se muestra en la Figura 15, etapa número siete. Si este es correcto, enciende una luz verde y muestra en pantalla el número de pedido. Si el despacho es incorrecto, el notificador genera una alerta audiovisual, enciende una luz roja, enciende un zumbador auditivo y muestra en pantalla el número de pedido erróneo. El zumbador no se detiene hasta que el operador presiona un botón, indicando qué el operador está enterado del problema en el despacho.

Figura 15: Flujo en Zapier cuando la orden escanéada existe

Si el despacho es correcto, el flujo en la nube continua, completando el despacho en el ERP. Si es incorrecto se detiene. Esto se hace por medio de un filtro en *Zapier*, se muestra en la Figura 15 como la etapa número ocho del proceso. Un filtro en *Zapier* es una etapa que detiene el flujo a menos que una condición dada se cumpla. En esta misma Figura se muestra cómo el proceso continúa si el filtro se supera. La función de las etapas que siguen a este filtro son para registrar dentro del ERP el despacho de la orden.

5.2.5. Proceso de verificación de despacho.

Para verificar si una orden fue despachada correctamente se diseñó un algoritmo en Python, este algoritmo corre dentro de la etapa seis de la Figura 8 y Figura 15. Funciona por medio de una comparación de vectores de códigos *SKU*, separados por comas. El primer vector proviene del dispositivo, es un vector de una dimensión que contiene todos los códigos escaneados por el operador. Este ingresó al flujo en *Zapier* en la etapa número uno mostrada en la Figura 14. El segundo vector proviene de la orden en el ERP. Ingresó al flujo en la etapa número cinco mostrados en la Figura 15

El vector del dispositivo debe coincidir con el vector de los productos disponibles de dicha orden. Para poder comparar ambos vectores es necesario pre procesar el vector de códigos *SKU* obtenido de la orden. Ya qué este puede contener servicios como el envío o garantías extendidas. Adicionalmente, la orden puede contener productos ordenados bajo pedido, es decir, qué la empresa genera una orden de compra de dicho artículo hasta que recibe la orden del cliente, por lo que no se tienen en inventario durante el primer despacho. El pre procesado identifica y retira estos códigos del vector a comparar. A continuación se muestra el diagrama de flujo del algoritmo.

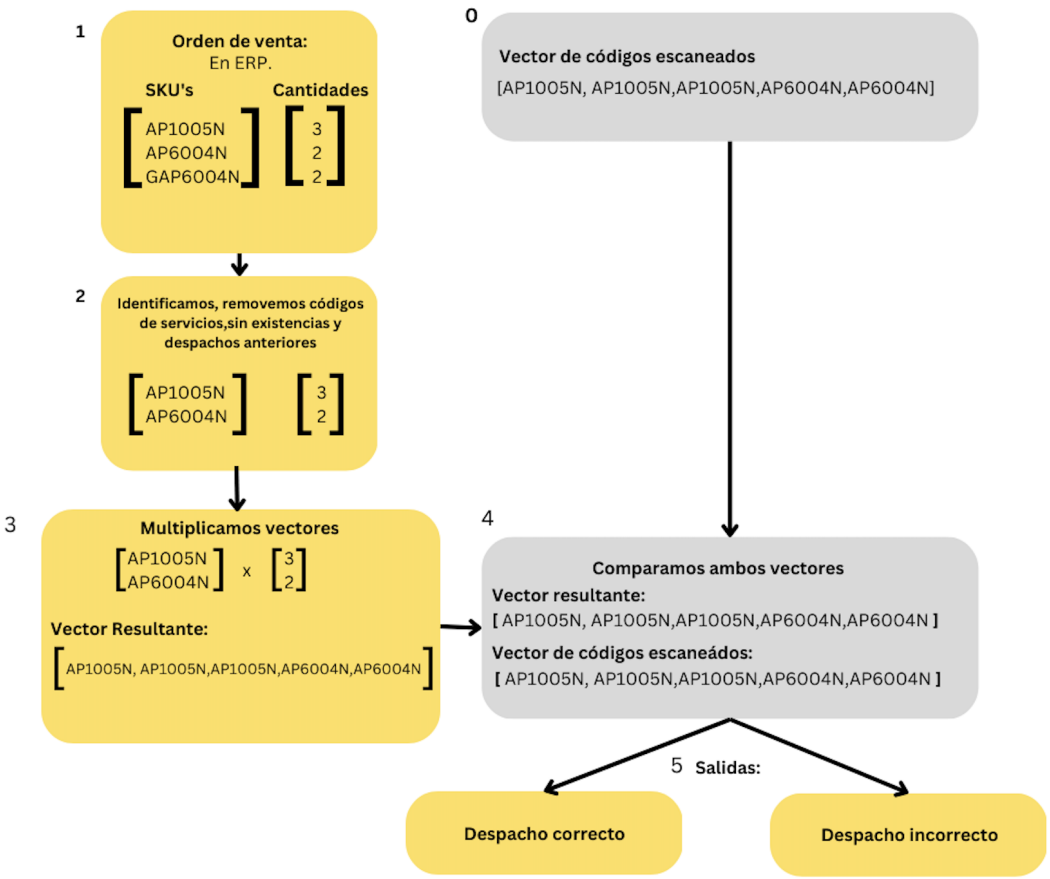


Figura 16: Diagrama de flujo de algoritmo para verificar despachos

Luego de obtener los vectores de códigos y cantidades del ERP por medio de la *API*, se consulta cada código con el ERP para verificar su disponibilidad y tipo. Se crea un nuevo vector corregido donde solo se incluyen los códigos disponibles y productos en venta, descontando los productos comprados bajo pedido y los servicios, completando el pre procesado. Luego se multiplica este vector comprobado por el vector de cantidades, para obtener un vector de una dimensión qué contiene todos los códigos disponibles para despacho de la orden, mostrado en la etapa número tres de la Figura 16 Por último se ordena en orden alfabético este vector verificado.

El vector proveniente del dispositivo también se ordena en orden alfabético. De este modo se puede comprobar si este es exactamente igual al vector procesado y ordenado de una dimensión proveniente del ERP, como se muestra en la etapa cuatro de la Figura 16 Si

estos dos son iguales, el despacho es correcto, de lo contrario el despacho es incorrecto. Este algoritmo corre en menos de un segundo.

Si el resultado del despacho es correcto, se envía por medio de la plataforma *Adafruit IO* una respuesta al segundo ESP8266 del dispositivo, qué actúa como notificador. Esta respuesta enciende una luz verde e indica en pantalla qué el despacho fue correcto. Si el despacho es incorrecto, a la respuesta enviada enciende una luz roja, muestra en pantalla el número de orden con despacho incorrecto y enciende una alarma auditiva, para detener esta alarma el operador debe presionar un botón, reconociendo qué hubo un error.

Si el despacho es correcto, el flujo en la nube continua, marcando el despacho y envió de la orden en el ERP. Finalizando el proceso de venta. De lo contrario, la salida de este algoritmo es incorrecta. Cuando esto sucede se envía la respuesta al notificador, cuando el notificador recibe una respuesta incorrecta. Enciende una luz roja, una alarma auditiva y muestra en pantalla el número de pedido erróneo. Como se muestra en la Figura 17

Figura 17: Respuesta del dispositivo en caso de despacho incorrecto

El sistema de alarma no se silencia hasta que pulsa un botón. Luego de responder con este mensaje de error, el flujo en la nube se detiene por medio de un filtro en *Zapier*. No registra ningún despacho. El operador deberá efectuar una revisión y repetición de la operación con el fin de completar el procedimiento de despacho de la orden.

5.3. Diseño y fabricación de carcasa.

El diseño de la carcasa se llevó a cabo al concluir la selección de los componentes electrónicos. El objetivo de la carcasa es proteger todos los componentes y facilitar el uso del dispositivo por parte del operador. Se utilizó un sistema CAD para diseñar y modelar los componentes de la carcasa, como se muestra en la Figura 18. Se midieron los componentes electrónicos, siendo las dimensiones de los mismos la única restricción del diseño. Se elaboraron dos carcasas, una para el módulo de despachador y otra para el notificador. Ambos diseños están compuestos por una carcasa y una tapadera con uniones por ajuste a presión. Como se muestra en la Figura 19 y en la sección de anexos, donde se muestra el juego de planos completo que detalla el diseño de la carcasa y pedestal.

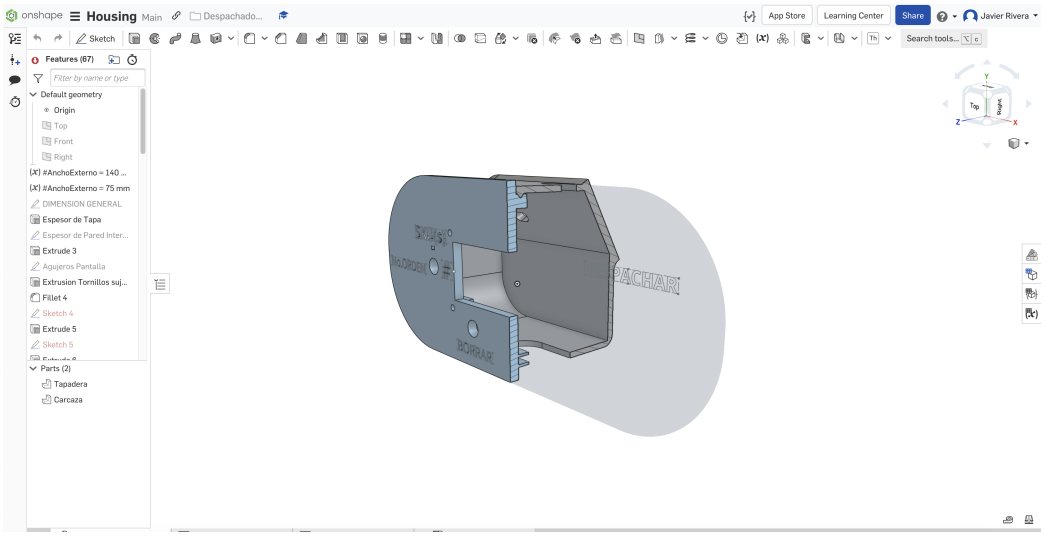


Figura 18: Diseño CAD de carcasa

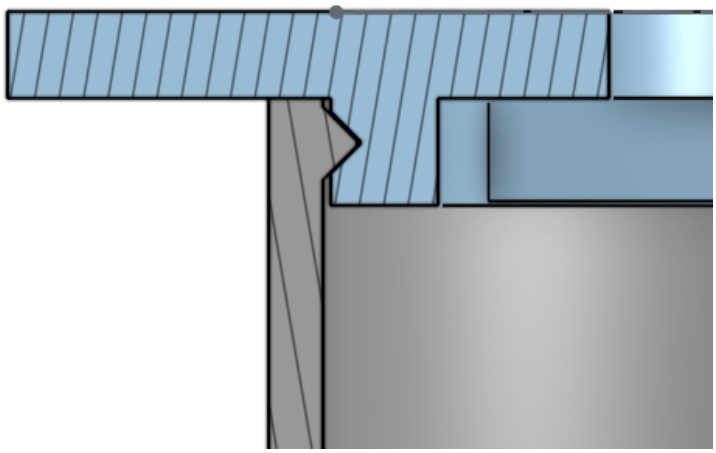


Figura 19: Conexión por ajuste a presión CAD.

El dispositivo se utilizará en un entorno de bodega semiindustrial, a temperatura ambiente, sin exposición a químicos o altas temperaturas. La cantidad de polvo y sedimentos en el entorno de operación es reducida, debido a que se trata de una bodega en la que se almacenan dispositivos electrónicos. Considerando el entorno de trabajo, se determinó que la carcasa podía ser confeccionada de plástico *PLA* impreso mediante una impresora 3D de tipo *FDM*. Se procesó el diseño *CAD* en *Ultimaker Cura* para transformarlo en código G. En la Figura 21 se muestran los parámetros de impresión de la carcasa. El proceso de diseño fue iterativo, se imprimieron más de diez versiones para llegar al diseño final. A continuanción se muestra la primera iteración de la carcasa.

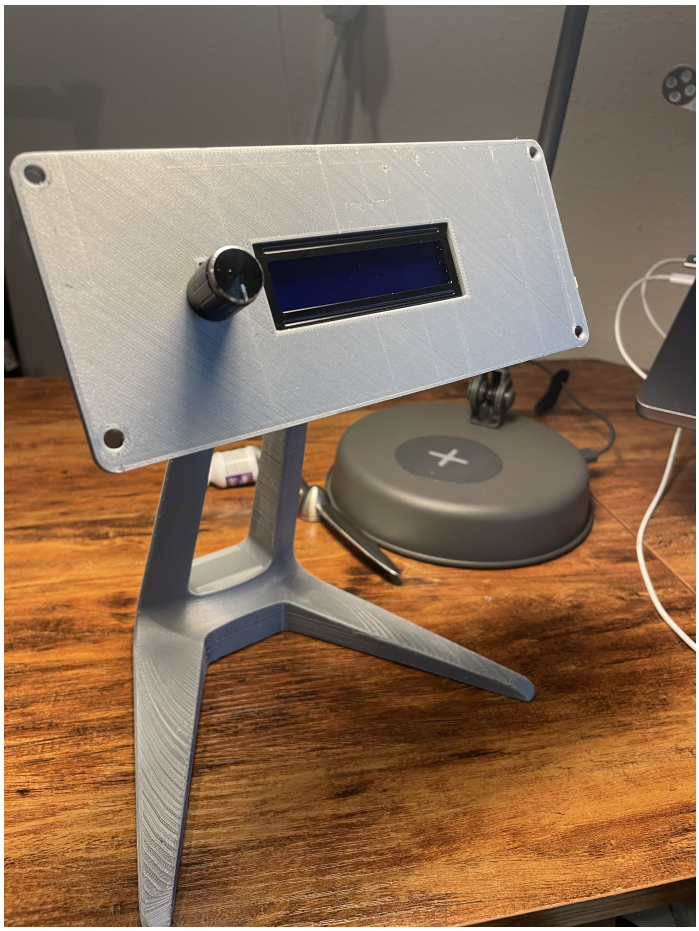


Figura 20: Primera iteración de diseño

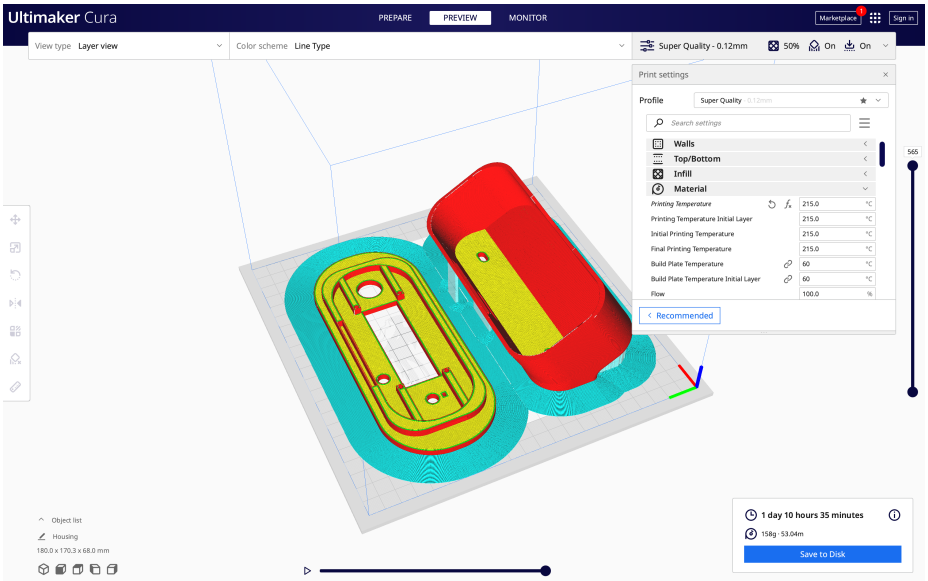


Figura 21: Generación de código G para impresión en Ultimaker Cura

Todos los elementos de la carcasa son de diseño y fabricación propio, excepto el pedestal. El pedestal se muestra en el juego de planos anexos, pieza número 4. El pedestal qué soporta la carcasa proviene de un soporte para colgar audífonos de la empresa *MakerBot*. El

pedestal es un diseño modificado de este modelo. La Figura 22 muestra el diseño original por *Ultimaker*, este es de dominio público y se encuentra disponible para descargar en la página web de Ultir

Figura 22: Diseño original de pedestal.

Fuente: Ultimaker,2018

Este diseño se modificó, agregando una superficie plana con dos agujeros distanciados por 100mm entre ellos. Este diseño permite montar una gran variedad de accesorios o dispositivos en esta base. El diseño es estable, ya que coloca el centro de gravedad de la carcasa dentro del área de soporte del pedestal. De igual manera, lo eleva de manera suficiente de la superficie en la que se ubica, con el fin de permitir la utilización del lector de códigos de barra sin interferencia. La Figura 23 muestra el diseño *CAD* del pedestal modificado para soportar el dispositivo. Esta modificación también contempla dos agujeros en la parte inferior del pedestal, lo que permite montarlo invertido.

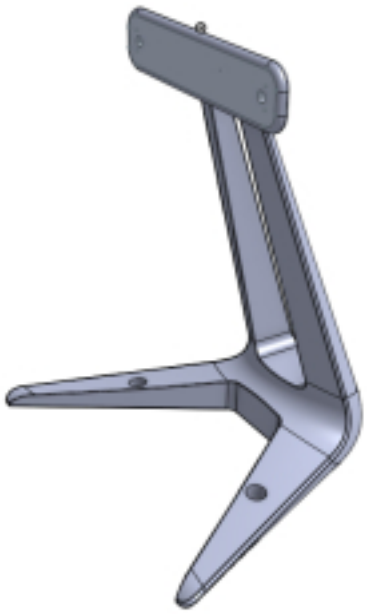


Figura 23: Diseño modificado de pedestal

El primer pedestal se imprimió con un relleno del 50 %. La instalación de esta pieza fue invertida, donde el dispositivo cuelga de una repisa. Al instalarse en bodega, esta cedió después de dos semanas de operación. La ruptura ocurrió en el punto de menor área de las columnas del pedestal, en la parte superior. Como se muestra en la Figura 24.

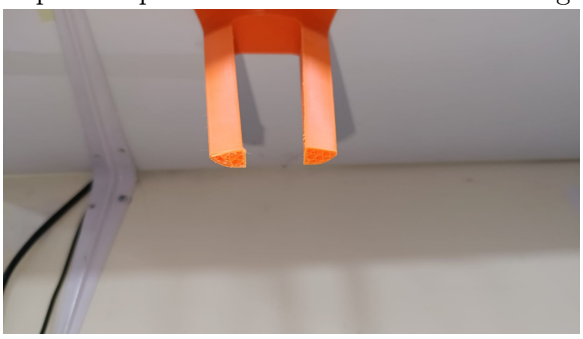


Figura 24: Falla en pedestal

La falla se debió a una concentración de esfuerzos, la primera iteración del pedestal se imprimió con un relleno del 50 %, por lo que la segunda iteración se imprimió con un relleno del 90 % reduciendo los esfuerzos en el material. Con esta segunda iteración no se volvió a presentar la falla al montar el dispositivo con el pedestal invertido.

Se armaron tres dispositivos, de los cuales dos se instalaron en bodega y el tercero se utilizó para este trabajo. La Figura 25 muestra los dispositivos siendo ensamblados.

Figura 25: Dispositivos siendo ensamblados

5.4. Capacitación de operarios.

Con el fin de asistir a los trabajadores en el empleo del dispositivo, se ha presentado una guía que detalla la ejecución de cuatro fases. Esta guía se colocó en la estación de trabajo del operario para facilitar la capacitación en caso de rotación de inventario. A continuación se muestran los cuatro modos de operación detallados en la guía.

- **Registro de número de orden**: registrar el número de orden de la lista de selección. El operador gira la perilla a la posición de número de orden, toma la lista de selección qué se imprime automáticamente en papel adhesivo el recibir una orden y escanea el código de barras que contiene el número de orden en la parte superior.

Figura 26: Modo de registro de número de orden

- **Registro de códigos *SKU***: el operador gira la perilla a la posición de "SKUS

  para escanear todos los códigos *SKU* qué se colocan dentro de la caja.

Figura 27: Modo de registro de códigos SKU

- **Registro de números de parte**: el operador gira la perilla a la posición de número de artículos, para escanear todos los números de parte qué se colocan dentro de la caja.

Figura 28: Modo de registro de números de parte

- **Confirmación**: mientas el operador termina de empacar la orden y cerrar la caja, el dispositivo sube a la nube los datos y evalúa el despacho contra la orden. Dando como respuesta si este fue correcto o incorrecto.

Figura 29: Respuesta del dispositivo

Tras de una hora de capacitación, los dos operarios en bodega se mostraron cómodos con el uso del dispositivo y lo continuaron utilizando para completar los despachos. Se les indicó qué debían reportar cualquier anomalía en el funcionamiento del dispositivo. De esta manera se detectaron muchos de los errores de diseño en el flujo en la nube, en el código de verificación y registro de la orden en el ERP. Cada vez que uno de los operadores reportaba una falla, se investigaba y corregía la causa, para que esta falla no se pudiera presentar de nuevo. Después de dos semanas de operación, el dispositivo funcionó correctamente el 98 % de los despachos.

CAPÍTULO 6

Resultados

6.1. Reducción de errores en despachos

Después de la construcción, implementación y capacitación del personal respecto al uso del nuevo dispositivo para realizar despachos, se operó durante un mes utilizando exclusivamente el dispositivo para hacer los despachos de las órdenes. Se establecieron restricciones en los permisos del operador en el sistema de gestión de recursos ERP con el fin de evitar que el usuario no pudiese omitir el uso del dispositivo. La presente medida garantizó la exclusividad del dispositivo en el proceso. Se tuvo un total de 2,121 despachos, de los cuales existieron 130 devoluciones. Como se mencionó anteriormente, la empresa captura la razón de la devolución cuando existe una solicitud de cambio. De esta forma se puede recopilar la información y medir el impacto del dispositivo en el proceso.

Se volvieron a exportar las razones de las devoluciones y se construyo el Cuadro 4, este muestra todas las devoluciones recibidas durante el mes de mayo de 2023.

Cuadro 4: Devoluciones y sus razones después de implementar el dispositivo

Razón de devolución	**Cantidad**	**Porcentaje**
Me enviaron el producto equivocado	4	3 %
Pedí el producto equivocado	27	21 %
Producto defectuoso	90	69 %
Ya no necesito el producto	9	7 %
Total	130	100 %

Las quejas por despachos incorrectos se redujeron de 16 mensuales como se mostró en el Cuadro 2 a 4 mensuales como se muestra en el Cuadro 4

6.2. Sistemas

6.2.1. Sistema en dispositivo

Al seguir una metodología de diseño incremental, se llegó a una versión del dispositivo que permite capturar 3 diferentes tipos de variables: número de pedido, códigos *SKU* y números de parte. Al presionar el botón de despacho, la unidad de escáner hacer una solicitud *HTTP* pasando como parámetros del URL las variables recopiladas. La unidad del notificador recibe la respuesta después de la evaluación en *Zapier*. La Figura 30 muestra el diagrama de flujo del sistema que corre en el dispositivo.

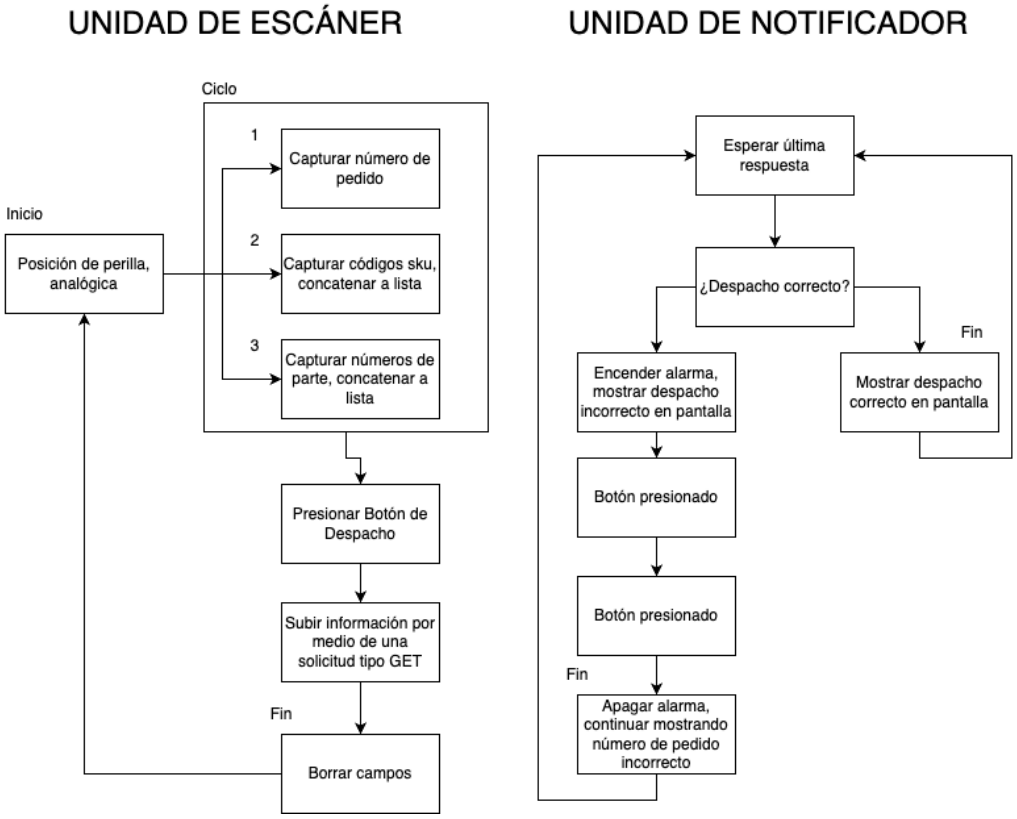


Figura 30: Última iteración de sistema programado en dispositivo

6.2.2. Resultados y seguimiento de despachos en la nube

El sistema desarrollado en la nube en la plataforma *Zapier* permite monitorear cada despacho de forma individual y remota. Cuenta con una sección llamada *Zap History* donde se muestra cada despacho. La Figura 31 muestra el historial de despachos visto en Zapier. Durante el mes de junio de 2023, el sistema se disparó 3200 veces, no arrojó ningún error debido a problemas en las etapas. Esto no quiere decir que el sistema sea perfecto, pero ya se considera lo suficientemente bueno para ser utilizado de manera confiable en bodega.

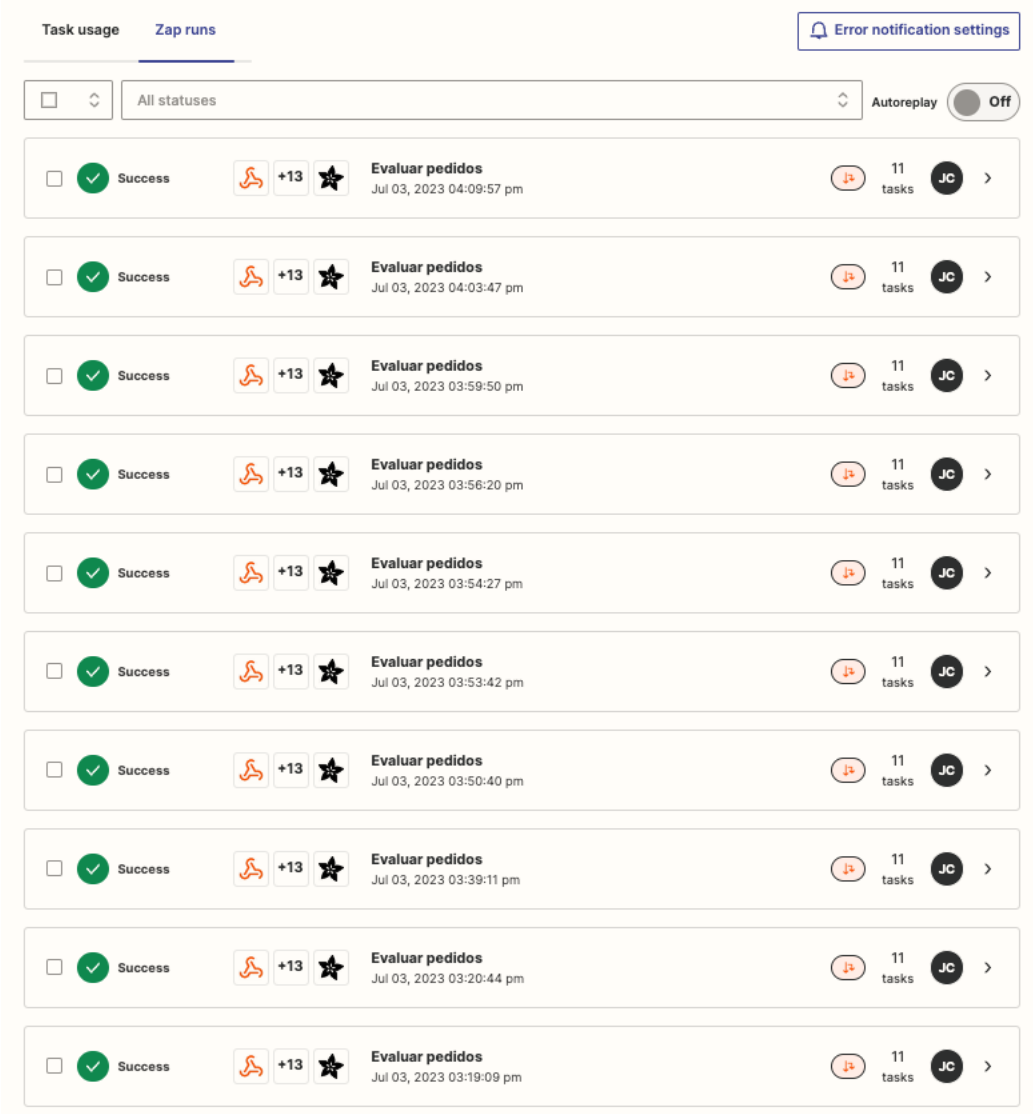


Figura 31: Historial de corridas exitosas en *Zapier*

6.3. Última iteración del prototipo

La última iteración de la carcasa del dispositivo se muestra ensamblada en la Figura 32. Muestra el dispositivo ensamblado por completo. El juego de planos en la sección de anexos muestra a detalle todas la dimensiones y características para replicar el diseño de la carcasa.

Figura 32: Dispositivo ensamblado

6.4. Capacitación de operarios.

Para capacitar a los operarios primero se les mostró el proceso de despacho utilizando el dispositivo. El código QR de la Figura 33 muestra la capacitación por medio de los ejemplos. Durante el resto del día se trabajó junto a los operarios, supervisado el proceso de despacho de todas las órdenes utilizando el dispositivo. Las figuras 33 y 34 muestran videos de los dos operarios utilizando el dispositivo para hacer los despachos.

Así mismo se les mostró por medio de ejemplos el proceso de despacho. Los operarios rápidamente comprendieron el nuevo proceso de despacho. La adopción del cambio fue buena, ya que reduce la carga de trabajo del operario, no requiere que esté utilice una computadora para buscar la orden, por lo que el proceso es más rápido. El código QR de la Figura 34 muestra el proceso de despacho realizado por un operario en la bodega.

Figura 33: Video de capacitación de operarios

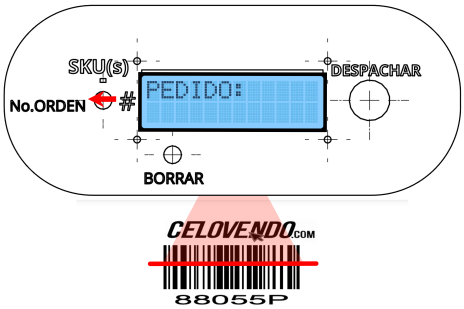
Figura 34: Video de proceso de despacho realizado por operario.

Se colocó la guía de despacho mostrada en la Figura 35 en el área de trabajo de los operarios, para facilitar el proceso de aprendizaje de nuevos operarios en el futuro.

CELOVENDO.COM

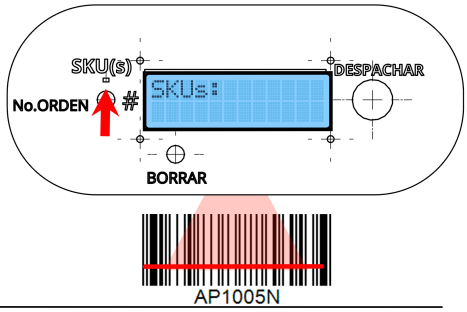
GUÍA DE DESPACHO.

1. Coloca la perilla en la posición: "*No. Orden*" y escanea el número de pedido de la lista de recolección.



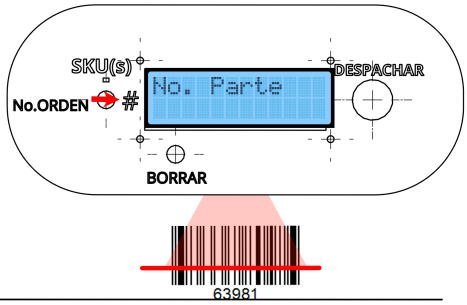
*Presiona borrar para borrar valores escanéados.

2. Gira la perilla a "*SKU(s)*" y escanea todos los códigos SKU de todos los productos en la lista de recolección.



*Presiona borrar para borrar valores escanéados.

3. Gira la perilla a "*#*" y escanea todos los códigos de parte de los productos. Presiona: "DESPACHAR".



4. **ESPERA UNA RESPUESTA**

Si es correcto, completa el despacho. Si es incorrecto, revisa la orden y vuelve a intentarlo.

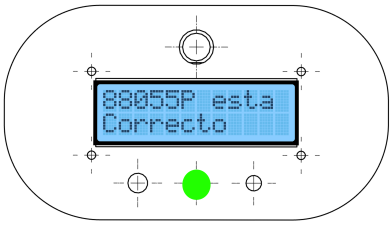


Figura 35: Guía de despacho colocado en área de trabajo.

CAPÍTULO 7

Discusión de resultados

7.1. Medición de impacto

Al comparar el Cuadro 2 con el Cuadro 4 se puede ver la reducción de despachos incorrectos de un 11 % a un 3 % mensual del total de las devoluciones. Sin embargo, el objetivo de este proyecto era eliminar por completo los despachos incorrectos, por lo que se procedió a investigar las cuatro devoluciones recibidas por despacho incorrecto.

Zapier permite revisar el historial de cada despacho y observar las entradas y salidas de cada etapa en el flujo en la nube. De esta forma se investigaron estos cuatro despachos. Los cuatro despachos mostraban que los códigos *SKU* escaneados durante el proceso de despacho eran correctos. Entonces, ¿por qué se recibieron estas cuatro devoluciones?

La Figura 36 muestra que el producto se etiquetó incorrectamente durante la recepción de la importación. Cuando el producto ingresa a la bodega, este se cuenta y clasifica. Durante este proceso de clasificación el producto fue etiquetado con el código SKU incorrecto. De tal forma que durante el proceso de despacho, el código correcto fue escaneado, pero el producto incorrecto fue enviado. Este caso sucedió dos veces.

Por otra parte, los otros dos despachos marcados como incorrectos se debieron a una clasificación incorrecta de los clientes al seleccionar la razón de la devolución. Estos cambios se realizaron, ya que el cliente solicitó un producto equivocado o no compatible con sus necesidades. Se revisaron estas dos corridas en la plataforma *Zapier* y se puedo observar que los despachos fueron correctos. Así mismo se comprobó el producto y se determinó que este fue correctamente etiquetado. Por lo que es probable que el cliente haya seleccionado erróneamente "me enviaron un producto equivocado" durante el proceso de cambio en la página web. De esta manera se puede modificar el Cuadro 4 para obtener el Cuadro 5

Con el Cuadro 5 se puede medir el impacto de la implementación del dispositivo en el proceso de despacho. Al compararlo con el Cuadro 2 podemos medir una disminución del 11 % de las devoluciones al 1.5 %. Esto representa una reducción del 86 % en los errores por

Figura 36: Productos con etiquetas equivocadas

Cuadro 5: Devoluciones después de implementar el dispositivo con postprocesado de datos

Razón de devolución	**Cantidad**	**Porcentaje**
Me enviaron el producto equivocado	2	1.5 %
Pedí el producto equivocado	29	21 %
Producto defectuoso	90	69 %
Ya no necesito el producto	9	7 %
Total	130	100 %

despacho incorrecto. Donde los únicos errores que existieron se debieron a un proceso previo al despacho que no se realizó correctamente, llevando así a la falla. No hay forma que el dispositivo detecte si el despacho fue incorrecto, si el operador escanea el código correcto, esto se encuentra fuera del alcance del dispositivo. La clasificación del producto se debe hacer de manera correcta cuando este ingresa a bodega para así evitar este tipo de errores en el proceso de despacho.

7.2. Operación con el dispositivo

Como es común con cualquier sistema nuevo, se detectaron varios problemas en su operación. Especialmente con el flujo de verificación en la nube, ya que este flujo depende de diferentes plataformas. Esta revisión en la nube ocurre específicamente en *Zapier*. Se observó que la plataforma de *Zapier* en el primer mes de operación tendió a fallar por periodos de tiempo limitados. Es decir, la disponibilidad de *Zapier* y su funcionamiento se ve reducido cuando este tiene problemas en sus servidores. En este periodo de tiempo *Zapier* presentó fallas por falta de servicio tres veces. Por lo que la operación del dispositivo se vio afectada.

En estos períodos de falta de disponibilidad, los operadores notificaron que los despachos no se estaban completando en el ERP. Fuera de estos periodos en los que Zapier estuvo fuera de servicio, se observó que los despachos sucederían en su mayor parte sin problemas.

Zapier es un *SAAS* con modelo de suscripción mensual, la empresa actualmente paga $ 129.00 mensuales para un plan que incluye 10,000 automatizaciones al mes, cada etapa que corre durante el flujo de verificación y despacho es considerado una automatización. Durante el mes de mayo, el 78 % del plan se consumió en el flujo para evaluar y despachar órdenes. Por lo que el costo estimado de este sistema en la nube fue de $100.72 mensuales ó $1,200.44 / Q9363.44 anuales. (tomando en cuenta un tipo de cambio promedio de Q7.8 por dólar al 31/05/2023) Cabe recordar que los costos aproximados por despachos incorrectos eran de Q.12,874.00 anuales. Por lo que el sistema es efectivo en eliminar los despachos incorrectos, pero no del todo efectivo en reducir los costos operativos, ya que se incurre en costos de suscripción mensuales de la plataforma Zapier.

Asimismo se encontraron errores en los programas de python en la etapa de verificación en casos extremos, estos casos se evidenciaron al estar utilizando el dispositivo en el campo. Un error que se encontró fue la existencia de comas dentro de los códigos *SKU*, como se mencionó anteriormente, la revisión de la orden se realiza por medio de una comparación de vectores. Estos vectores son listados separadas por comas, cuando un código *SKU* contiene una coma, la lista aparenta ser más grande de lo que realmente es, produciendo errores en el código. Para resolver esto se revisaron todos los códigos *SKU* de la empresa y se modificaron de tal forma que un código *SKU* no pueda contener un carácter especial o coma. Esta regla se agregó al manual operativo de la empresa, en la sección de creación de nuevos productos.

Gracias a la reducción de errores durante el proceso de despacho, la empresa continuó utilizando exclusivamente los dispositivos para realizar sus despachos.

CAPÍTULO 8

Conclusiones

Se construyó e implementó el dispositivo en el centro de distribución. Luego de operar por un mes exclusivamente usando los dispositivos instalados en bodega para realizar los despachos, se midió una reducción del 11 % del total de devoluciones (16 mensuales) a un 1.5 % (2 mensuales). Los dispositivos contribuyeron a la reducción en los despachos incorrectos casi por completo, los errores qué existieron se debieron a un etiquetado incorrecto, causa qué se encuentra fuera del alcance de este proyecto.

Se diseñó e implementó un algoritmo en lenguaje Python para pre-procesar y comparar matrices para así poder comparar el despacho contra la orden de venta, dando una respuesta inmediata al operario. El diagrama de flujo de este algoritmo se observa en la Figura 16 y el programa completo se encuentra en los anexos, sección 11.2. Este algoritmo toma como entradas la información del despacho escaneada por el operario y la información de la orden proveniente del ERP, devuelve una respuesta booleana respecto a la exactitud del despacho.

Así mismo se diseñó una carcasa a la medida en CAD, el juego de planos se encuentra en los anexos. Se imprimió utilizando una impresora 3D FDM. La primera iteración del pedestal falló, ya que se imprimió con un relleno del 50 % causando una concentración de esfuerzos de tensión en la sección de menor área transversal del pedestal. Al aumentar el relleno a 90 % la falla no volvió a presentarse. Esto se debe a que la carga del dispositivo está distribuida en más material, reduciendo los esfuerzos en el material.

Se capacitó al personal de bodega sobre el uso del dispositivo, por medio de una guía impresa y capacitación uno a uno. Actualmente, la empresa utiliza exclusivamente los dispositivos para hacer sus despachos.

CAPÍTULO 9

Recomendaciones

9.1. Futuras iteraciones

Debido a que los errores del despacho pueden proceder de los ingresos de mercadería a bodega. Se recomienda implementar un proceso de verificación durante la recepción de productos, qué garanticé la exactitud del proceso de clasificación y etiquetado de productos.

El problema más habitual en el escaneo de códigos de barras con el dispositivo construido es que este lea solo parte del código de barras, acción ante la cual el operador debe presionar el botón de borrar y repetir el proceso. Se recomienda utilizar códigos de barra que cuenten con caracteres qué denotan el inicio y final del código de barras, como lo son los códigos tipo 93. Otra posible solución a este problema sería utilizar códigos de barra con carácter de verificación.

Se recomienda que el botón de borrar que utiliza el operador en el dispositivo, solo borre el último valor escaneado, ya que en órdenes extensas, cuando se presiona el botón de borrar, el operador debe empezar de nuevo todo el proceso.

9.2. Reducción de costos operativos

Para futuras iteraciones del diseño se recomienda implementar el proceso de verificación en la nube en una plataforma más robusta. *Zapier* es una herramienta de *Nocode* ideal para prototipos, ya que permite probar una idea más rápido con menores costos de desarrollo inicial. Esta no es ideal para correr flujos en la nube a largo plazo debido a sus altos costos y riesgos por indisponibilidad. Se recomienda para futuras iteraciones del diseño utilizar *AWS (Amazon web services)* o una herramienta similar para correr el flujo de verificación. De esta manera se reducen las fallas por falta de disponibilidad de *Zapier* y los costos de suscripciones mensuales, haciendo más efectiva la reducción de costos de este proyecto.

CAPÍTULO 10

Bibliografía

2D Barcode Scanner Setting Manual. (2019). DongGuan DYscan Technology Co., Ltd. DongGuan, China, DongGuan DYscan Technology Co., Ltd.

Alawi, O. (2018). *What is the difference between HTML vs XML vs JSON?* Consultado el 16 de agosto de 2023, desde https://medium.com/@oazzat19/what-is-the-difference-between-html-vs-xml-vs-json-254864972bbb

Allan, A. (2018). *Getting Started with the ESP8266*. Consultado el 22 de abril de 2023, desde https://medium.com/@aallan/getting-started-with-the-esp8266-270e30feb4d1

Bayer. (2011). *Snap-Fit joints for plastics - A design guide*. Consultado el 20 de agosto de 2023, desde https://fab.cba.mit.edu/classes/S62.12/people/vernelle.noel/Plastic_Snap_fit_design.pdf

Carolo, L. (2022). *3D Printing Snap Fit Joints: How to Design Print Them*. Consultado el 9 de abril de 2023, desde https://all3dp.com/2/3d-printing-snap-fit-design-simply-explained/

Chateka, S. (2017). *Why should you use standard HTTP methods while designing REST APIs?* Consultado el 16 de agosto de 2023, desde https://medium.com/@suhas_chatekar/why-you-should-use-the-recommended-http-methods-in-your-rest-apis-981359828bf7

Crymble, A. (2022). Download de Múltiplos Registros usando Query Strings (F. Lamarca, Trad.). *Programming Historian Em Português*, *2*. https://doi.org/10.46430/phpt0034

DearSystems. (2021). *Ecommerce Industry Insights and Trends*. Consultado el 9 de abril de 2023, desde https://dearsystems.com/ecommerce-industry/

Dedase, A. G. (2019). *Architecting a Scalable Software as a Service*. Consultado el 9 de abril de 2023, desde https://medium.com/swlh/architecting-a-scalable-software-as-a-service-14a4073ae7c2

Gazarov, P. (2016). *What is an API? In English, please.* Consultado el 9 de abril de 2023, desde https://medium.com/p/b880a3214a82

Gutteridge, L. (2016). *What is ERP?* Consultado el 9 de abril de 2023, desde https://medium.com/@drgutteridge/what-is-erp-64c8b613c35

Hemel, Z. (2007). *JSON vs XML*. Consultado el 15 de agosto de 2023, desde https://medium.com/zef-me/json-vs-xml-9f69aabb1528

Herman, H. (2022). *Coffee for everyone: How Fellow scaled a global wholesale business with Zapier*. Consultado el 9 de abril de 2023, desde https://zapier.com/blog/fellow-scaling-with-zapier/

Khella, A. (2020). *Who should use NoCode tools?* Consultado el 9 de abril de 2023, desde https://medium.com/makervana/who-should-use-nocode-tools-2a2b648f8d9a

Kononenko, K. (2018). *Amazon Web Services (AWS) explained by operating a brewery*. Consultado el 19 de abril de 2023, desde https://medium.com/free-code-camp/amazon-web-services-aws-explained-by-operating-a-brewery-8f1e91eacc40/

Marko, K. (2014). *The SAP Money Pit: Plenty of Waste in Need of SaaS Efficiencies*. Consultado el 9 de abril de 2023, desde https://medium.com/@krmarko/the-sap-money-pit-plenty-of-waste-in-need-of-saas-efficiencies-79ba803aa4bb

Martinez, K. (2020). *Automation basics*. Consultado el 9 de abril de 2023, desde https://zapier.com/learn/zapier-quick-start-guide/quick-start-the-basics/

Musick, M. (2020). *How Software as a Service (SaaS) Benefits Your Business*. Consultado el 9 de abril de 2023, desde https://medium.com/illumination/how-software-as-a-service-saas-benefits-your-business-a09ec4b931dd

Ofiwe, M. (2021). *A Beginner's Guide to URL Parameters*. Consultado el 9 de abril de 2023, desde https://www.semrush.com/blog/url-parameters/

Radečić, D. (2020). *JSON explained for Python users: Data Science Edition*. Consultado el 15 de agosto de 2023, desde https://medium.com/towards-data-science/json-explained-for-python-users-data-science-edition-18e9859944da

Robertson, T. (2022). *Code by Zapier: How to add customizable triggers and actions to your Zaps*. Consultado el 9 de abril de 2023, desde https://zapier.com/blog/code-by-zapier-guide/

CAPÍTULO 11

Anexos

11.1. Imágenes

Figura 37: Anexo 2. Primera conexión con carcaza

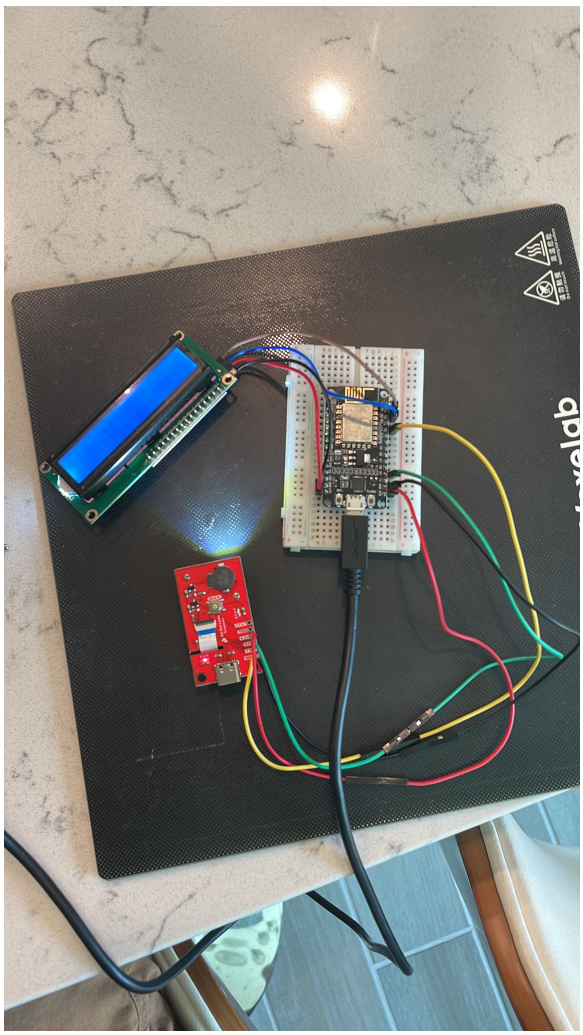


Figura 38: Anexo 1. Primera iteración

Figura 39: Anexo 3. Despachador con componentes Electrónicos Ensamblado

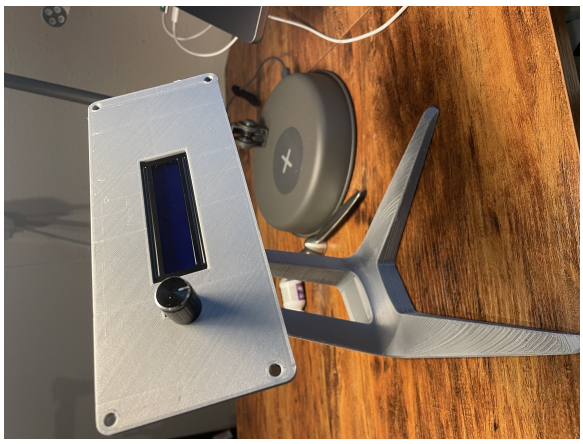


Figura 40: Anexo 4. Iteración de despachador con componentes electrónicos ensamblados

11.2. Algoritmo para verificar despacho

```
#El algorimo se define como una funcion, las entradas son: codigos escaneados (skus), Lista de codigos provenientes de ERP (DearSKu), Lista de cantidades provenientes de ERP y el identificador de la orden (order_id)
def SKUVeri(Skus,DearSKU,DearSKUQt,dear_order_id):
    # Declaracion de variables
  try:
    SKUs = Skus.split(",") # SKUs
    SKUs = ["None"]

  try:
    DearSKU = DearSKU.split(",") #Listado de SKUS de ERP
  except:
    DearSKU = ["None"]

  try:
    DearSKUQt = DearSKUQt.split(",") # Listado de cantidades de ERP
  except:
    DearSKUQt = ["1"]
#Generacion de listas vacias
  stock_array=[]
  comparison_array=[]
  #Definicion de encabezado para consultas API al ERP.
  headers_lib={
    'content_type':'application/json',
    'api-auth-accountid':#confidencial,
    'api-auth-applicationkey':#confidencial'
  }
#El algoritmo inicia revisando si existen despachos previos
  skus_despachados=[]
  cantidad_despachada=[]
  respuesta_despacho=requests.get("https://inventory.dearsystems.com/ExternalApi/v2/sale/fulfilment?SaleID={sale_id}&IncludeProductInfo=false".format(sale_id=dear_order_id),headers=headers_lib).json()
  cantidad_despachos_previos=len(respuesta_despacho["Fulfilments"])
  packi=[]
  for i in range(cantidad_despachos_previos):
    cantidad_lineas=len(respuesta_despacho["Fulfilments"][i]["Pack"]["Lines"])
    if cantidad_lineas>0:
      for j in range(cantidad_lineas):
       packi.append(respuesta_despacho["Fulfilments"][i]["Pack"]["Lines"][j])

  cantidad_articulos_despachados=len(packi)

  for i in range(cantidad_articulos_despachados):
    articulo=packi[i]
    skus_despachados.append(articulo["SKU"])
    cantidad_despachada.append(articulo["Quantity"])
```

```
44
45  print("SKUs despachados: ",skus_despachados)
46  print("Cantidades despachadas: ",cantidad_despachada)
47
48  dear_qty_despues_despachos=[]
49
50  for i in range(len(DearSKU)):
51    if DearSKU[i] in skus_despachados:
52      index_despachado=skus_despachados.index(DearSKU[i])
53      dear_qty_despues_despachos.append(str(int(DearSKUQt[i])-int(
          cantidad_despachada[index_despachado])))
54    else:
55      dear_qty_despues_despachos.append(str(int(DearSKUQt[i])))
56
57  print("Cantidad ajustada despues de despachos: ",
      dear_qty_despues_despachos)
58
59  qty_sin_ceros=[]
60  skus_sin_ceros=[]
61  #Elimino los elementos que ya estan despachados.
62  for i in range(len(dear_qty_despues_despachos)):
63    if dear_qty_despues_despachos[i]!="0":
64      qty_sin_ceros.append(dear_qty_despues_despachos[i])
65      skus_sin_ceros.append(DearSKU[i])
66  print("Skus despues de quitar ceros: ",skus_sin_ceros)
67  print("Cantidades Despues de quitar los ceros: ",qty_sin_ceros)
68
69  #Skus_sin_ceros y qty_sin_ceros ingresan al siguiente bloque de
      verificacion.
70
71  DearSKU=skus_sin_ceros
72  DearSKUQt=qty_sin_ceros
73
74  #Chequeamos la disponibilidad de los productos para descartar los
      articulos bajo pedido.
75
76  for i in range(len(DearSKU)):
77    respuesta=requests.get("https://inventory.dearsystems.com/
        ExternalApi/v2/ref/productavailability?Sku={sku}".format(sku=
        DearSKU[i]),headers=headers_lib).json()
78
79    #El siguiente bloque existe ya que cada articulo puede tener su
        inventario en mas de un almacen virtual en el ERP.
80
81    if respuesta['Total']==0:
82      stock_array.append(0)
83    if respuesta['Total']>0:
84      sumatoria_temporal_almacenes = 0
85      #inicializo el contador
86      for i in range(respuesta['Total']):
87      #No tomamos en cuenta el inventario en el almacen de garantias
          .
88        if 'Garantias' in respuesta['ProductAvailabilityList'][i]['
            Location']:
```

```
            sumatoria_temporal_almacenes+=0
          else:
            sumatoria_temporal_almacenes+=respuesta['
               ProductAvailabilityList'][i]['OnHand']

        stock_array.append(sumatoria_temporal_almacenes)

    for i in range(len(stock_array)):
      if stock_array[i]>=int(DearSKUQt[i]):
        comparison_array.append(int(DearSKUQt[i]))
      else:
        comparison_array.append(int(stock_array[i]))
    print("Array de comparasion: ",comparison_array)


    comparison_array_string=[]
    for i in range(len(comparison_array)):
      comparison_array_string.append(str(comparison_array[i]))

    DearSKUQt=comparison_array_string


    # Inicia la comparacion de vectores para verificar despacho.

    for i in range(0,len(SKUs)):
      SKUs[i]=SKUs[i].upper() #Pasamos el pedido a mayusculas.
      SKUs[i]=SKUs[i].replace(" ","")

    for i in range(0,len(DearSKU)):
      DearSKU[i] = DearSKU[i].upper() #Pasamos el pedido a mayusculas.
      DearSKU[i] = DearSKU[i].replace(" ","")
      DearSKUQt[i] = DearSKUQt[i].replace(" ","")

    # Para convertir la lista de cantidades en un numeros enteros
    DearSKUQt = [int(x) for x in DearSKUQt]

    # Se inicia la prevariable de SKU con su largo
    DearSKUcountPRE = []
    for i in range(len(DearSKUQt)):
      DearSKUcountPRE.append([None] * sum(DearSKUQt))
#Multiplicamos el vector de SKU con el vector de cantidades, ambos
    provienen del erp.
    for i in range(0, len(DearSKU)):
      DearSKUcountPRE[i] = [DearSKU[i]] * DearSKUQt[i]

    # Vector de una dimension
    DearSKUcountPOST = []
        # Iterate through the outer list
    for element in DearSKUcountPRE:
      if type(element) is list:
        # If the element is of type list, iterate through the sublist
        for item in element:
          DearSKUcountPOST.append(item)

```

```
    else:
      DearSKUcountPOST.append(element)



#Para filtrar los "None" de la lista.
DearSKUcountPOST = [i for i in DearSKUcountPOST if i is not None]


DearSKUcountNoG = [None] * len(DearSKUcountPOST)


# Para eliminar los que empiezan con G y filtrar las garantias
for i in range(0,len(DearSKUcountPOST)):
  if(DearSKUcountPOST[i].startswith("G")):
    DearSKUcountNoG[i] = DearSKUcountPOST[i][1:]
  else:
    DearSKUcountNoG[i] = DearSKUcountPOST[i]


## Se realiza la interseccion entre los dos vectores, para obtener los SKUs unicos.
DearSKUinter = [value for value in DearSKUcountNoG if value in DearSKUcountPOST]

DearSKUleftinter = [value for value in DearSKUcountNoG if value not in DearSKUcountPOST]

for i in range(0,len(DearSKUleftinter)):
    DearSKUleftinter[i] = "G" + DearSKUleftinter[i]

DearSKUfullinter = DearSKUleftinter + DearSKUinter

DearSKUfullinter = set(DearSKUfullinter)
DearSKUfullinter = (list(DearSKUfullinter))

# Para seleccionar los elementos de la lista Post que esta en la de inter
DearSKUcountCONS = [item for item in DearSKUcountPOST if item in DearSKUfullinter]

# Para reordenar ambos vectores
SKUs.sort()
DearSKUcountCONS.sort()
# Se compara si ambos vectores son identicos
if SKUs == DearSKUcountCONS:
   respuesta="si esta correcto"

else:
   respuesta="no esta correcto"

if len(SKUs)==0:
  respuesta="no esta correcto"

```

```
190
191   return respuesta
```

11.3. Código en despachador

```
1  //DESPACHADOR.
2  //Ultima version editada por Javier Rivera 01/04/2023
3  //Codigo Funcional Probado en el 2023 para el ajuste del dispositivo
       de scanner de pedidos.
4
5
6  #include <ESP8266WiFi.h>
7  #include "SoftwareSerial.h"
8  #include "SparkFun_DE2120_Arduino_Library.h"  http://librarymanager/
       All#SparkFun_DE2120
9  #include <LCD_I2C.h>
10 #include <DNSServer.h>
11 #include <ESP8266WebServer.h>
12 #include <WiFiManager.h>
13 #define BUFFER_LEN 40
14 LCD_I2C lcd(0x27, 16, 2);
15 SoftwareSerial softSerial(2, 3);
16 DE2120 scanner;
17
18 const char* host = "hooks.zapier.com";  // base del link para hacer
       el HTTP request
19
20 // Arreglo para el buffer del scanner.
21 char scanBuffer[BUFFER_LEN];
22
23 // Pin del boton y delpotenciometro.
24 const int analogInPin = A0;
25 const int despacharbot = 10;
26 const int borrarbot = 16;
27
28
29 // Valor del sensor del boton y de los contadores de productos.
30 int sensorValue = 0;
31 int buttonValue = 0;
32 int eraseValue = 0;
33 int SKUcount = 0;
34 int ITEMcount = 0;
35 int continueorder = 1;
36 String ordenflag = "";
37
38 // URL del webhook.
39 String url = "/hooks/catch/13989032/bjk1zca?";
40 String BIGhook = "";
41
42 //String para arreglos de Items y producto.
43 String NoOrden;
```

```
String SKUunico;
String SKUultimo;
String NoItemunico;
String NoOrdenCons;
String SKUvec = "";
String NoItemvec;
String num_item_ultimo;

//////////////////////////////////////////////////////////////////






void setup() {

  lcd.begin();
  lcd.setCursor(0, 0);
  lcd.backlight();
  lcd.print("Conectando Wifi");
  Serial.begin(115200);
  delay(10);

  WiFiManager wifiManager;



  // Cremos portal cautivo y comprobamos si
  // se establece la conexi n
  if(!wifiManager.autoConnect("Despachador #3")){
    lcd.clear();
    lcd.print("Portal Wifi");
    Serial.println("Fallo en la conexi n (timeout)");
    ESP.reset();
    delay(1000);
  }

  //inicializacion de var
  //config de botones---
  pinMode(despacharbot, INPUT);
  digitalWrite(despacharbot, HIGH);

  pinMode(borrarbot, INPUT);
  digitalWrite(borrarbot, HIGH);





  //Serial.println();
  //Serial.print("Connecting to ");
  //Serial.println(ssid);

```

```
  //WiFi.begin(ssid, password);

  //while (WiFi.status() != WL_CONNECTED) {
    //delay(500);
    //Serial.print(".");
  //}
  //Serial.println("");
  //Serial.println("WiFi connected");
  //Serial.println("IP address: ");
  //Serial.println(WiFi.localIP());
  if (scanner.begin(softSerial) == false) {
    //Serial.println("Scanner did not respond. Please check wiring.
       Did you scan the POR232 barcode? Freezing...");
    while (1)
      ;
  }
  Serial.println("Scanner online!");
  Serial.print("connecting to ");
  Serial.println(host);
  lcd.clear();
  lcd.print("Wifi conectado!");
  delay(2000);
  lcd.clear();

  if (scanner.readBarcode(scanBuffer, BUFFER_LEN)) {
    for (int i = 0; i < strlen(scanBuffer); i++) {}
  }
}
void loop() {
lcd.clear();
  // Use WiFiClient class to create TCP connections
  WiFiClientSecure client;
  const int httpPort = 443;

  client.setInsecure();  // Se habilita el cliente como inseguro.


  ////----------------------------------------/////
  ////----------Filtro para el Potenciometro-----------/////
  ////----------------------------------------/////

  int numLec = 10;
  int senseSum = 0;

  for (int k = 0; k < numLec; k++) {
    senseSum += analogRead(analogInPin);
    delay(1);
  }

  int senseAve = senseSum / numLec;
  // Lee posicion de la perilla para que puedan ser divisiones
     claras.
  sensorValue = (senseAve / 340);

```

```
150   //Serial.println(sensorValue);
151
152   ////---------- Termina Filtro -------------//////
153
154   //-Valor-del-boton ------------------------------
155   buttonValue = digitalRead(despacharbot);
156   eraseValue = digitalRead(borrarbot);
157   switch (sensorValue) {
158     case 0:
159       {
160         lcd.clear();
161         lcd.print("1.No. ORDEN");
162         lcd.setCursor(13, 0);
163         lcd.print(ordenflag);
164         lcd.setCursor(0, 1);
165         lcd.print(NoOrdenCons);
166         if (scanner.readBarcode(scanBuffer, BUFFER_LEN)) {
167           for (int i = 0; i < strlen(scanBuffer); i++)
168             NoOrden = NoOrden + String(scanBuffer[i]);  // Variable
                  de No de orden escaneado.
169           NoOrden.trim();
170           //lcd.print(NoOrden);
171           //delay(800);
172           ordenflag = "OK";
173           NoOrdenCons = NoOrden;
174           NoOrden = "";
175           delay(500);
176         }
177
178         else {
179           delay(200);
180         }
181
182         if (eraseValue == 1) {
183           Serial.println(eraseValue);
184           ordenflag = "";
185           NoOrdenCons = "";
186           NoOrden="";
187           pinMode(borrarbot, OUTPUT);
188           digitalWrite(borrarbot, LOW);
189           pinMode(borrarbot, INPUT);
190         }
191         break;
192       }
193
194     case 1:
195       {
196         lcd.clear();
197         lcd.print("2.SKUs");
198         lcd.setCursor(13, 0);
199         lcd.print(SKUcount);
200         lcd.setCursor(0, 1);
201         lcd.print(SKUultimo);
202         if (scanner.readBarcode(scanBuffer, BUFFER_LEN)) {
```

```
203           for (int i = 0; i < strlen(scanBuffer); i++)
204             SKUunico = SKUunico + String(scanBuffer[i]);  //
                  Variable de sku escaneado.
205           SKUunico.trim();
206           lcd.print(SKUunico);
207           //delay(800);
208           //Contadores para llevar la cuenta del SKU
209           if (SKUcount > 0) {
210             SKUvec += "," + SKUunico;
211             SKUcount = SKUcount + 1;
212           } else {
213             SKUvec += SKUunico;
214             SKUcount = SKUcount + 1;
215           }
216           // Se borra el valor de SKUunico.
217           SKUultimo=SKUunico;
218           SKUunico = "";
219           //delay(500);
220         }
221         else {
222           delay(200);
223         }
224
225         if (eraseValue == 1) {
226           SKUvec = "";
227           SKUcount = 0;
228           SKUultimo="";
229           pinMode(borrarbot, OUTPUT);
230           digitalWrite(borrarbot, LOW);
231           pinMode(borrarbot, INPUT);
232         }
233
234         break;
235       }
236
237     default:
238       {
239         lcd.clear();
240         lcd.print("3.No Item");
241         lcd.setCursor(13, 0);
242         lcd.print(ITEMcount);
243         lcd.setCursor(0, 1);
244         lcd.print(num_item_ultimo);
245         if (scanner.readBarcode(scanBuffer, BUFFER_LEN)) {
246           for (int i = 0; i < strlen(scanBuffer); i++)
247             NoItemunico = NoItemunico + String(scanBuffer[i]);  //
                  Variable de item escaneado.
248           NoItemunico.trim();
249           num_item_ultimo=NoItemunico;
250           //lcd.print(NoItemunico);
251           if (ITEMcount > 0) {
252             ITEMcount = ITEMcount + 1;
253             NoItemvec += "," + NoItemunico;
254           }
```

```
255
256           else {
257             ITEMcount = ITEMcount + 1;
258             NoItemvec += NoItemunico;
259           }
260
261           // Se borra el valor de Item unico.
262           NoItemunico = "";
263           delay(200);
264         }
265
266         else {
267           delay(200);
268         }
269
270         if (eraseValue == 1) {
271           NoItemvec = "";
272           ITEMcount = 0;
273           num_item_ultimo="";
274           pinMode(borrarbot, OUTPUT);
275           digitalWrite(borrarbot, LOW);
276           pinMode(borrarbot, INPUT);
277         }
278
279         break;
280       }
281   }
282
283
284   while (buttonValue == 0 ) {
285     lcd.clear();
286     lcd.print("Despachando...");
287
288     //Revisa si esta conectado.
289     if (!client.connect(host, httpPort)) {  //if not working
290       lcd.clear();
291       lcd.setCursor(0, 0);
292       lcd.print("Conexion Perdida");
293       delay(4000);
294       continueorder = 0;
295     }
296
297     else{
298       //Esta bien
299       continueorder = 1;
300     }
301
302     if (continueorder == 1) {
303       //Se manda al cliente --- parte importante ---
304       BIGhook = url + "pedido=" + NoOrdenCons + "&skus=" + SKUvec +
              "&partes=" + NoItemvec;
305       client.print("GET " + BIGhook + " HTTP/1.1\r\n" + "Host: " +
              host + "\r\n" + "Connection: close\r\n\r\n");
```

```
306       //Serial.println("GET " + BIGhook + " HTTP/1.1\r\n" + "Host: "
              + host + "\r\n" + "Connection: close\r\n\r\n");
307
308       // Se borran las variables
309       NoOrdenCons = "";
310       SKUvec = "";
311       SKUultimo="";
312       NoItemvec = "";
313       num_item_ultimo="";
314       ordenflag = "";
315       SKUcount = 0;
316       ITEMcount = 0;
317       lcd.clear();
318       lcd.print("Despacho enviado");
319       delay(800);
320
321       //Se reactiva el boton.
322       pinMode(despacharbot, OUTPUT);
323       digitalWrite(despacharbot, HIGH);
324       pinMode(despacharbot, INPUT);
325       buttonValue = digitalRead(despacharbot);
326     }
327   }
328 }
```

11.4. Diagrama de conexiones

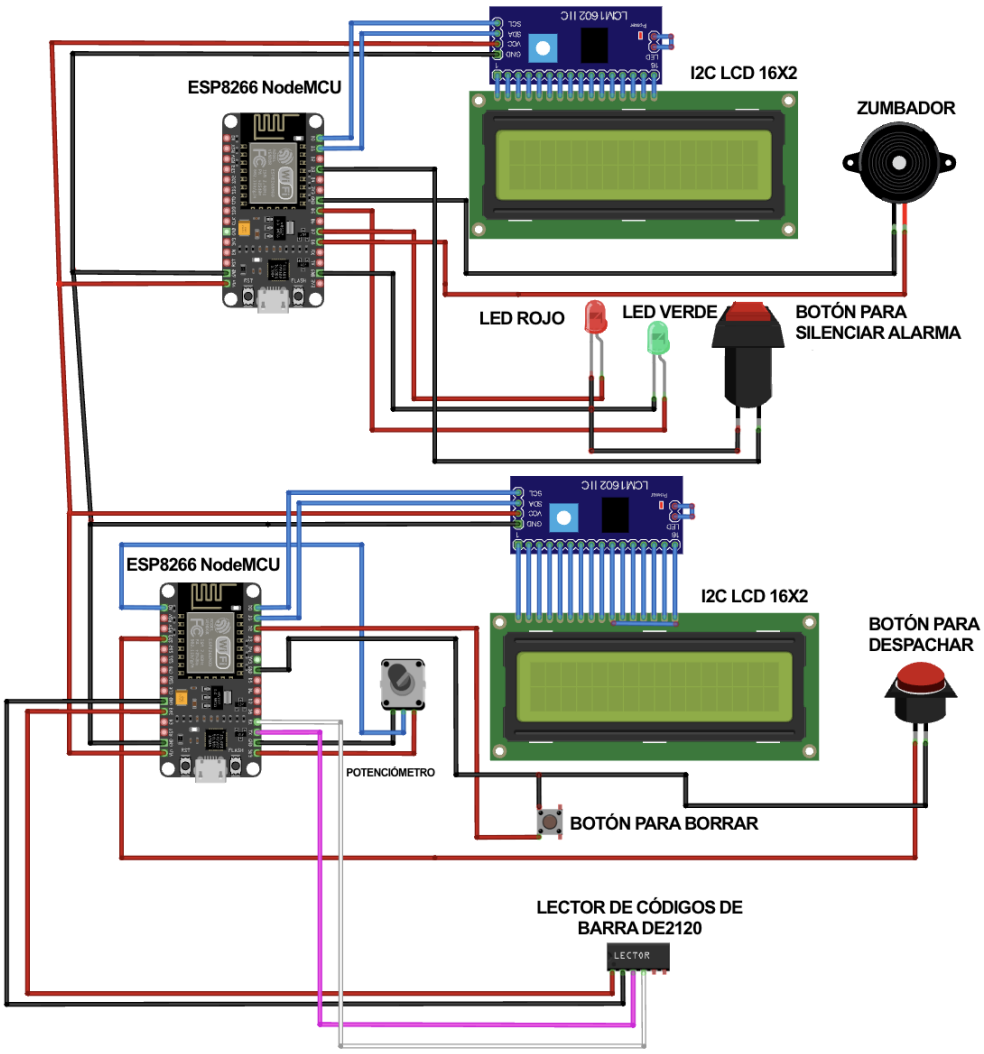
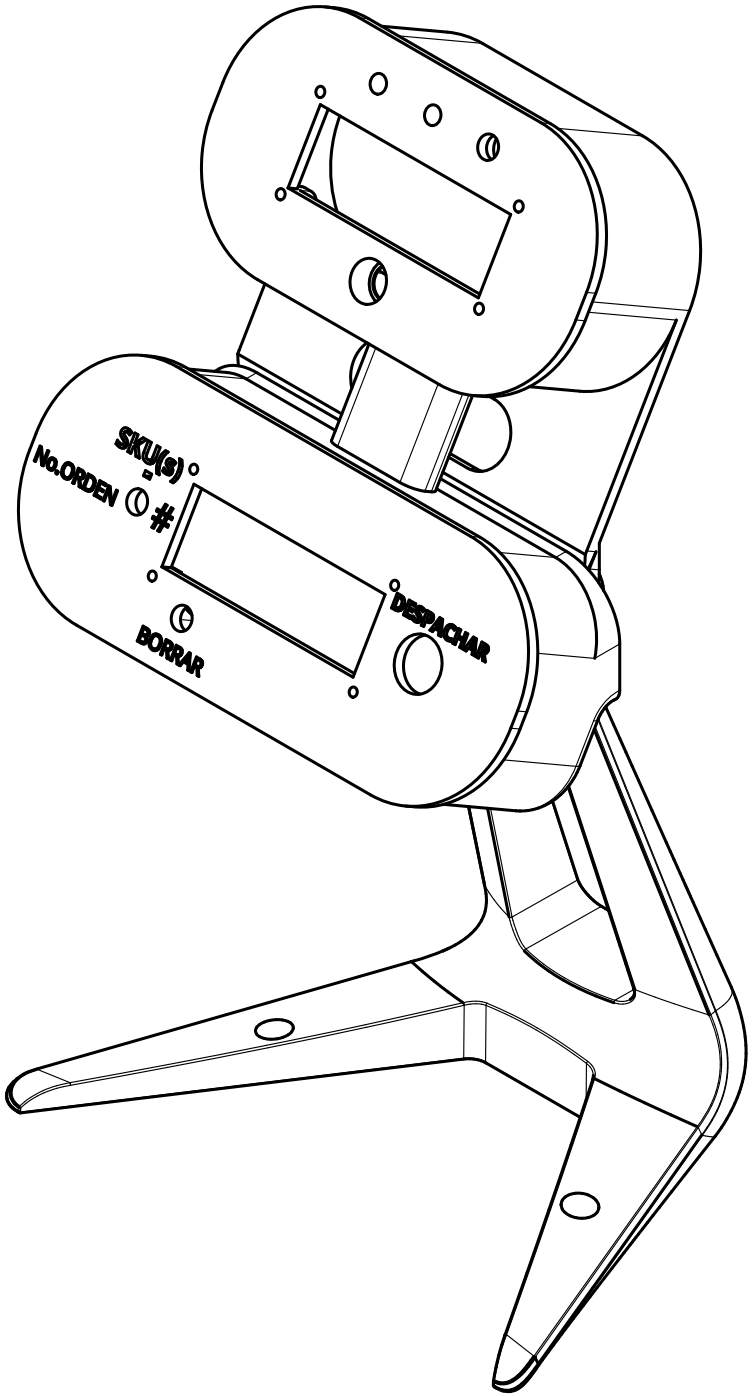
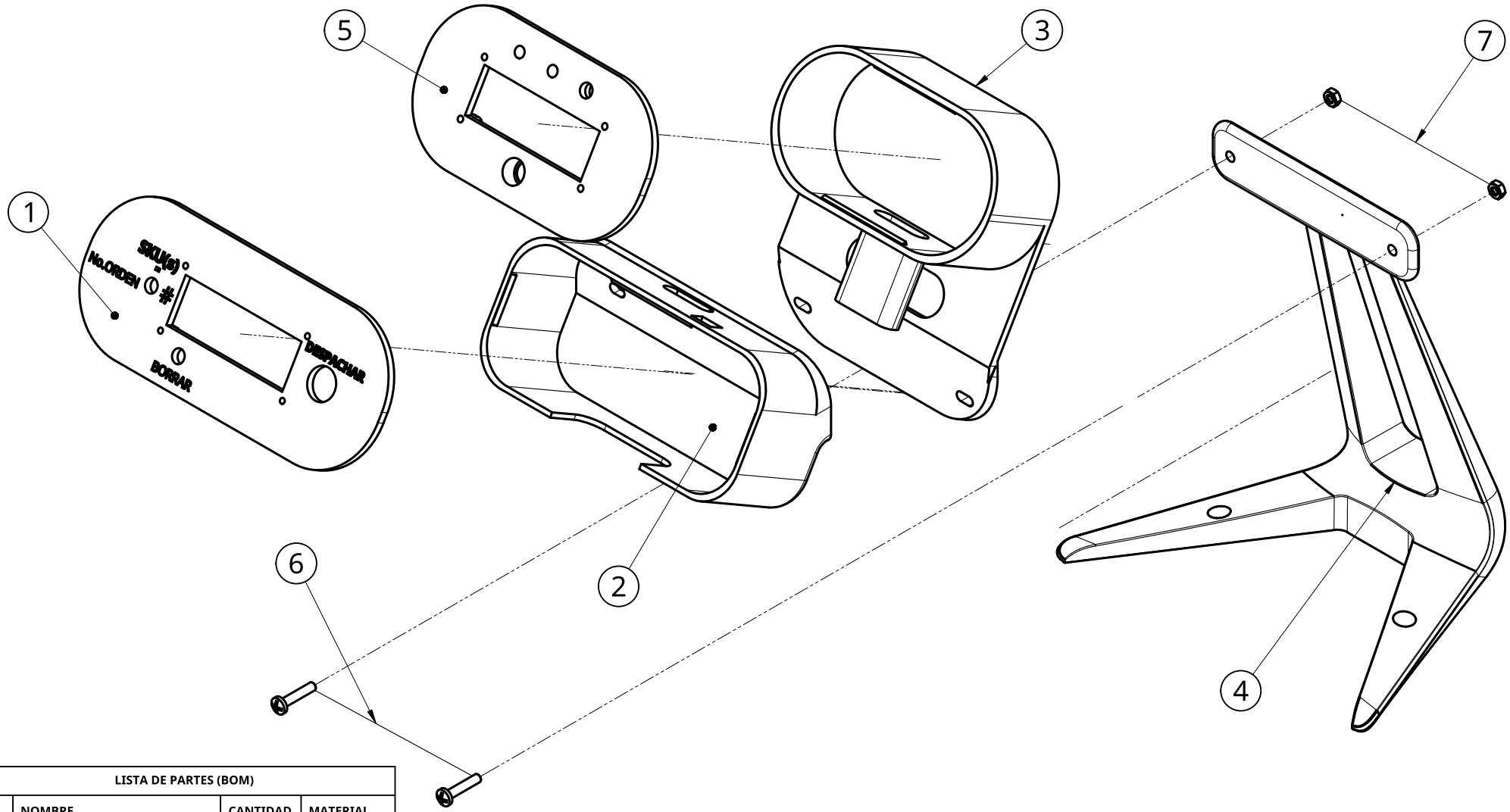


Figura 41: Anexo 5. Diagrama de conexiones

11.5. Juego de planos



CELOVENDO.COM			TÍTULO: VISTA ISOMÉTRICA DE ENSAMBLE
TERCER ÁNGULO DE PROYECCIÓN	ITERACIÓN: 1	DISEÑADOR: JAVIER RIVERA U.	DESCRIPCIÓN / INSTRUCCIONES DE ENSAMBLE: TAPA FRONTAL CON AJUSTE A PRESIÓN A LA CARCAZA, EN ESTA PARTE SE ENSAMBLAN LOS COMPONENTES.
	FECHA: 01-12-2022	AUXILIAR: -	
	MATERIAL: PLA	MASA: 60g	TAMAÑO: CARTA / NÚMERO DE DIBUJO: 1 / REV.: 1
TODAS LAS DIMENSIONES EN MILIMETROS A MENOS QUÉ SE INDIQUE LO CONTRARIO	ACABADO: -		ESCALA: 1:2 / PAGINA: 1



LISTA DE PARTES (BOM)			
#	NOMBRE	CANTIDAD	MATERIAL
1	TAPA FRONTAL - DESPACHADOR	1	PLA
2	CARCAZA DESPACHADOR	1	PLA
3	CARCAZA DE NOTIFICADOR	1	PLA
4	PEDESTAL	1	PLA
5	TAPA PARA NOTIFICADOR	1	PLA
6	Tornillos M4 X 20	1	ACERO
7	TUERCAS M4	1	ACERO

CELOVENDO.COM

TÍTULO: VISTA ISOMÉTRICA EXPLOTADA DE ENSAMBLE

TERCER ÁNGULO DE PROYECCIÓN

ITERACIÓN: 1

DISEÑADOR: JAVIER RIVERA U.

FECHA: 01-12-2022

AUXILIAR: -

MATERIAL: PLA

MASA: 460g

DESCRIPCIÓN / INSTRUCCIONES DE ENSAMBLE:

GUÍA DE ENSAMBLE DE COMPONENTES DE DESPACHADOR.

TAMAÑO: CARTA

NÚMERO DE DIBUJO: 1

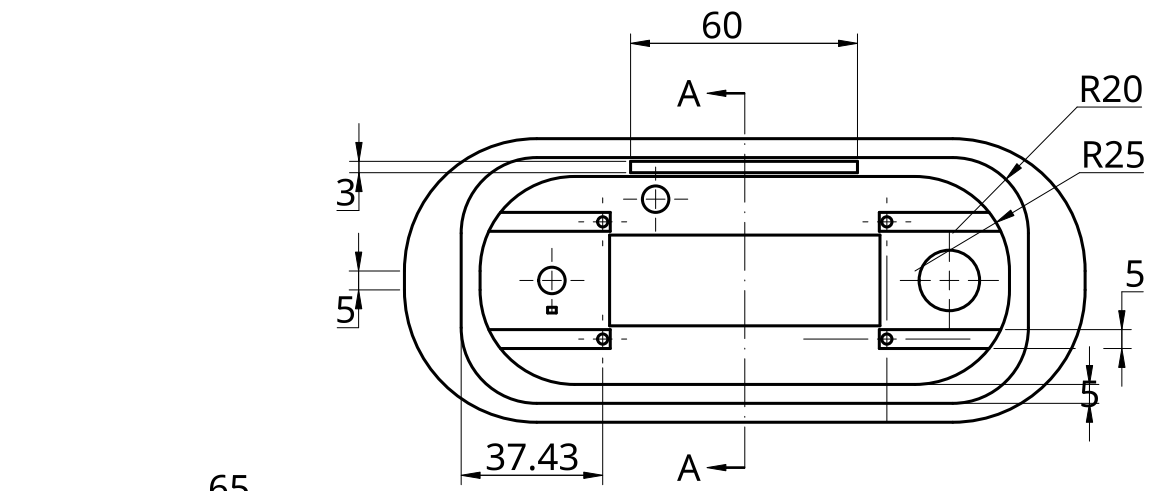
REV.: 1

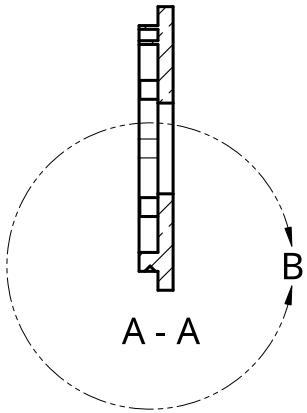
TODAS LAS DIMENSIONES EN MILIMETROS A MENOS QUÉ SE INDIQUE LO CONTRARIO

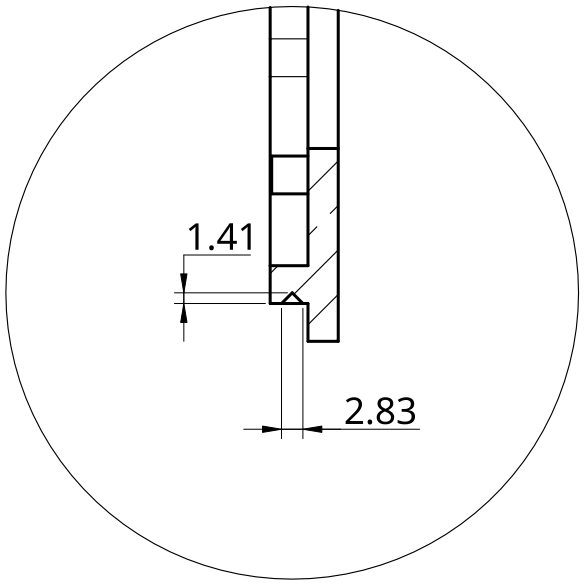
ACABADO: -

ESCALA: 1:2.5

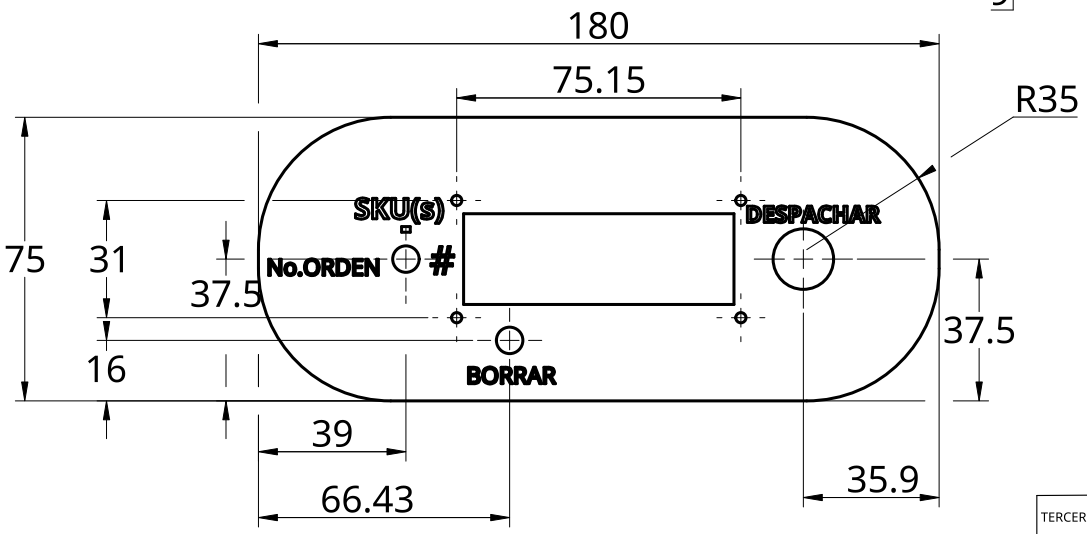
PAGINA: 2





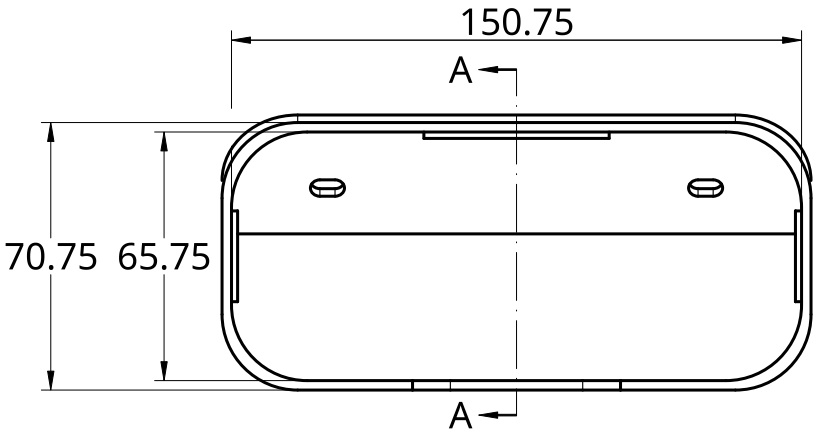


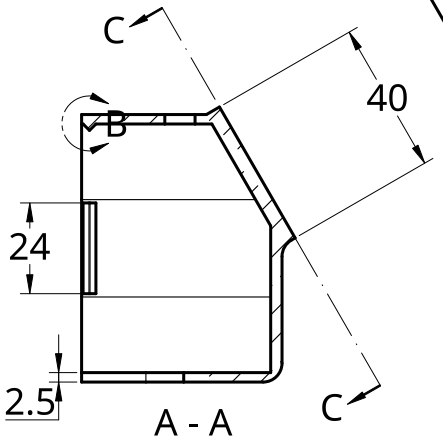
B
1:1



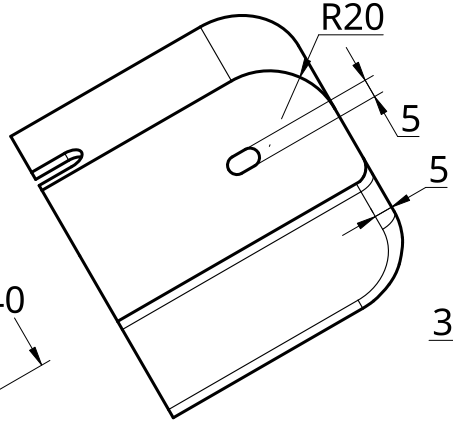
CELOVENDO.COM			TÍTULO: TAPA FRONTAL- DESPACHADOR
TERCER ÁNGULO DE PROYECCIÓN	ITERACIÓN: 1	DISEÑADOR: JAVIER RIVERA U.	DESCRIPCIÓN / INSTRUCCIONES DE ENSAMBLE: TAPA FRONTAL CON AJUSTE A PRESIÓN A LA CARCAZA, EN ESTA PARTE SE ENSAMBLAN LOS COMPONENTES.
	FECHA: 01-12-2022	AUXILIAR: -	
	MATERIAL: PLA	MASA: 60g	TAMAÑO: CARTA / NÚMERO DE DIBUJO: 1 / REV.: 1
TODAS LAS DIMENSIONES EN MILIMETROS A MENOS QUÉ SE INDIQUE LO CONTRARIO	ACABADO: IMPRIMIR SOBRE PLACA DE VIDRIO PARA OBTENER UN ACABADO LISO. NO REQUIERE SOPORTES		ESCALA: 1:2 / PAGINA: 3

NOTA: LA DISTANCIA ENTRE CENTROS DE LAS RAUNRAS ES DE 100MM

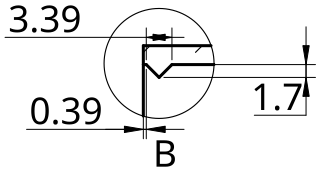




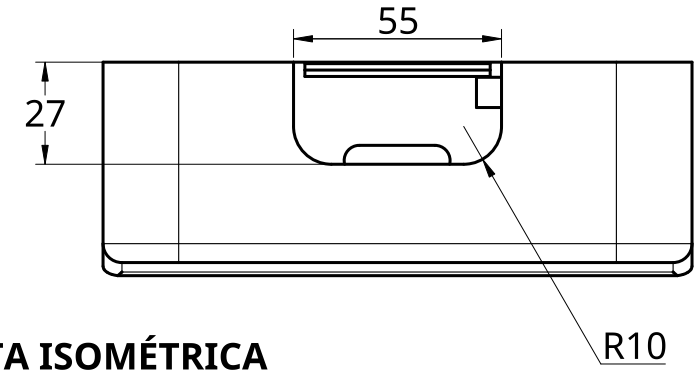
A - A

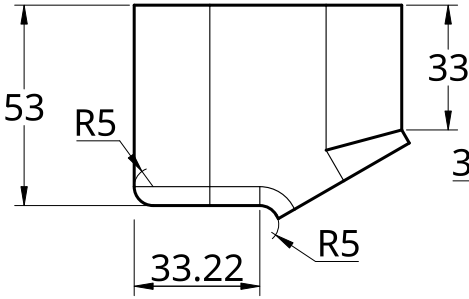


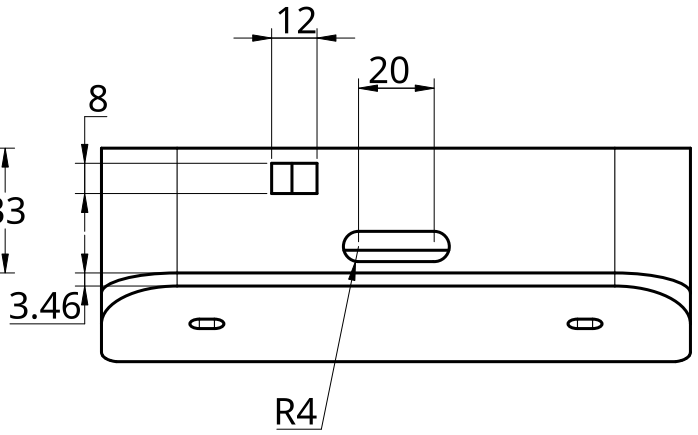
C - C



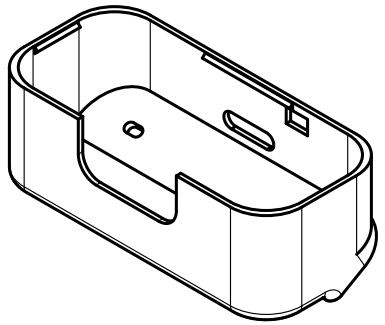
1:1





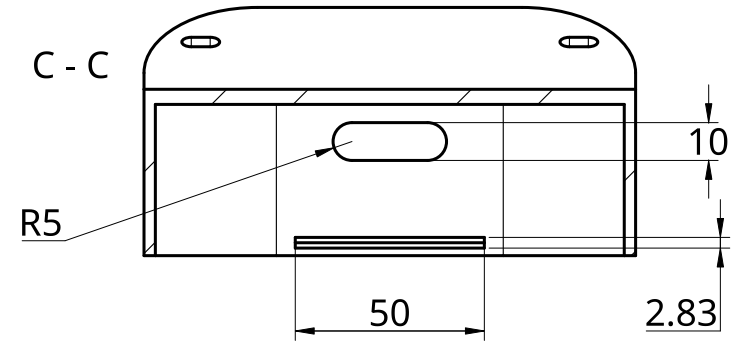


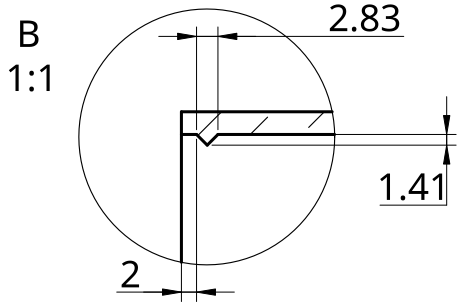
VISTA ISOMÉTRICA


1:3

NOTA: TODAS LAS TOLERANCIAS DE ±0.1MM A MENOS QUE SE INDIQUE LO CONTRARIO

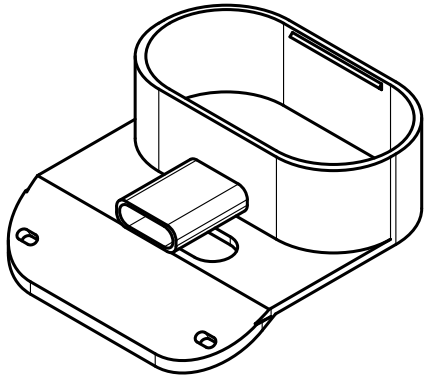
CELOVENDO.COM			TÍTULO: CARCAZA - DESPACHADOR
TERCER ÁNGULO DE PROYECCIÓN	ITERACIÓN: 1	DISEÑADOR: JAVIER RIVERA U.	DESCRIPCIÓN / INSTRUCCIONES DE ENSAMBLE: LA CARCAZA ENCAJA POR CONEXIÓN A PRESIÓN CON LA TAPA DEL DESPACHADOR. SE FIJA AL LA BASE USANDO TORNILLOS
	FECHA: 01-12-2022	AUXILIAR: -	
	MATERIAL: PLA	MASA: 70g	TAMAÑO: CARTA / NÚMERO DE DIBUJO: 1 / REV.: 1
TODAS LAS DIMENSIONES EN MILIMETROS A MENOS QUÉ SE INDIQUE LO CONTRARIO	ACABADO: PARA IMPRIMIR COLOCAR		ESCALA: 1:2 / PAGINA: 4



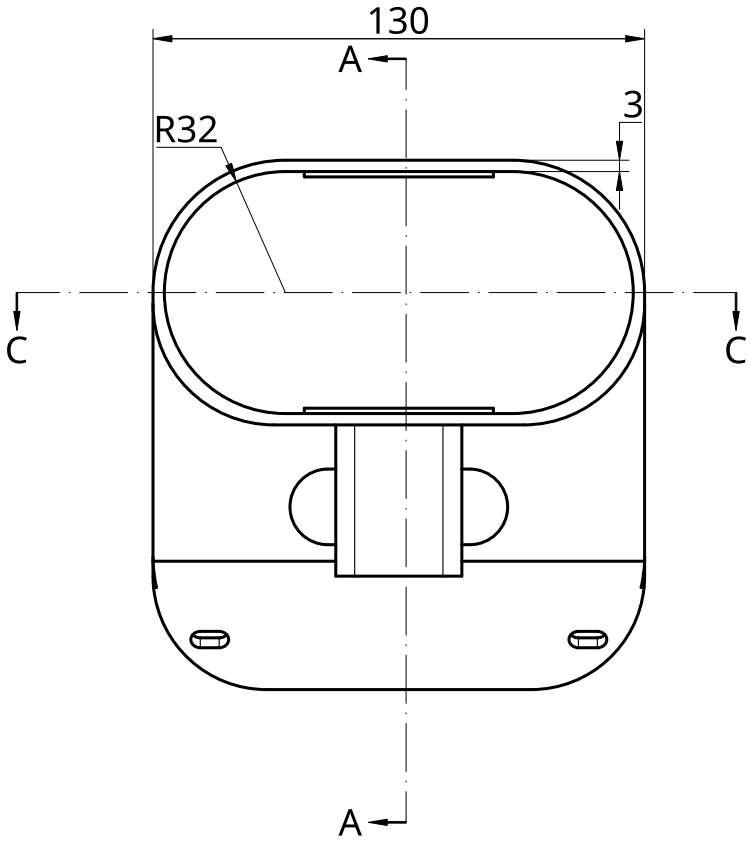


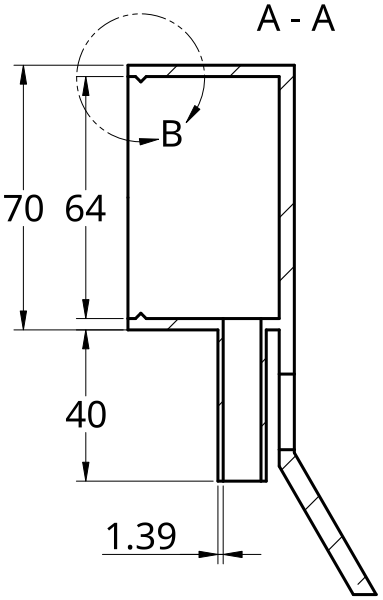
NOTA: TODAS LAS RANURAS DE LA CONEXIÓN POR AJUSTE A PRESIÓN TIENEN LAS MISMAS DIMENSIONES DEL DETALLE A-A ACERCAMIENTO B

VISTA ISOMÉTRICA



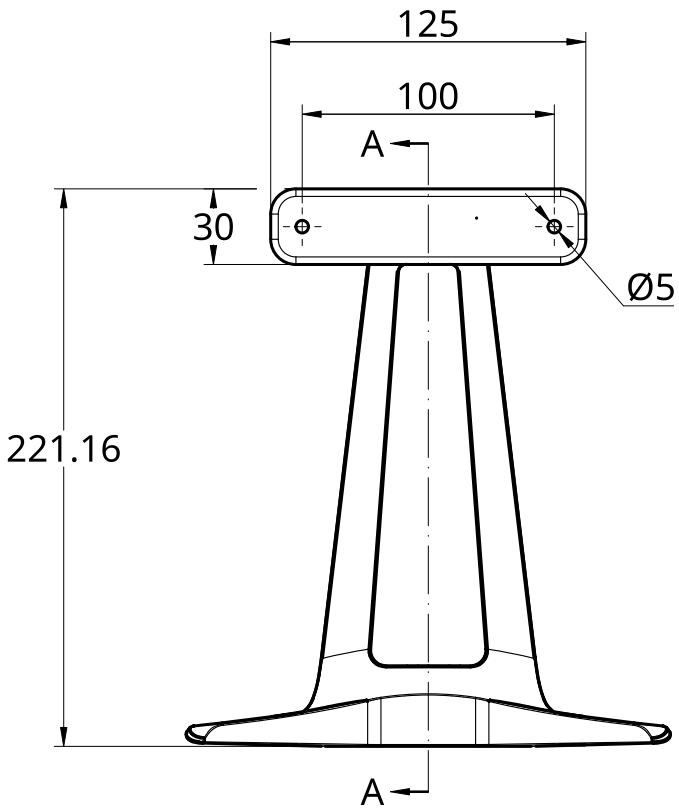
1:3

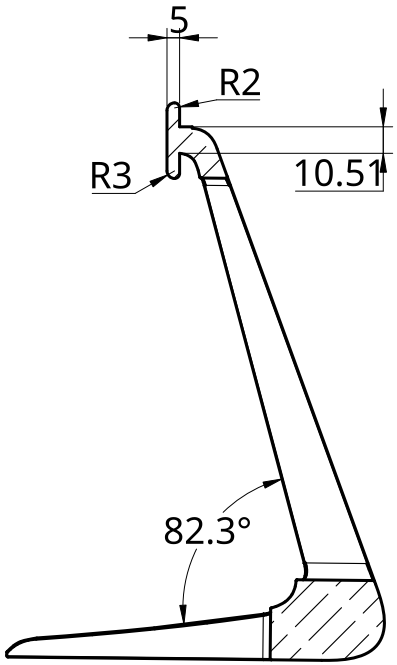




NOTA: TODAS LAS TOLERANCIAS DE ±0.1MM A MENOS QUÉ SE INDIQUE LO CONTRARIO.

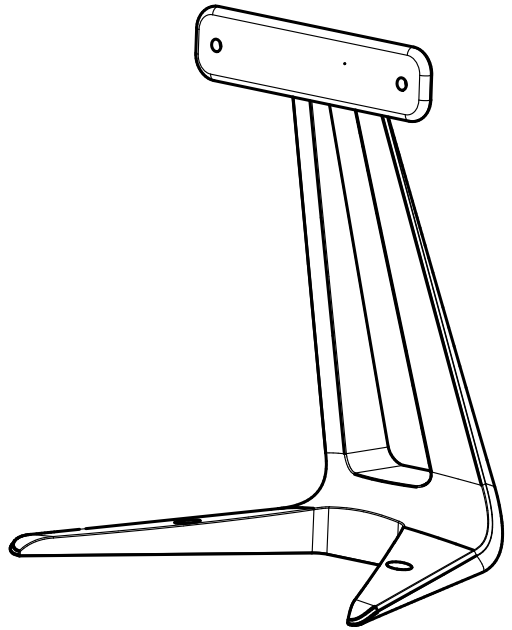
CELOVENDO.COM			TÍTULO: CARCAZA DE NOTIFICADOR
TERCER ÁNGULO DE PROYECCIÓN	ITERACIÓN: 1	DISEÑADOR: JAVIER RIVERA U.	DESCRIPCIÓN / INSTRUCCIONES DE ENSAMBLE: CARCAZA DEL NOTIFICADOR
	FECHA: 01-12-2022	AUXILIAR: -	
	MATERIAL: PLA	MASA: 60g	
TODAS LAS DIMENSIONES EN MILIMETROS A MENOS QUÉ SE INDIQUE LO CONTRARIO	ACABADO: IMPRIMIR SOBRE PLACA DE VIDRIO PARA OBTENER UN ACABADO LISO. NO REQUIERE SOPORTES		TAMAÑO: CARTA / NÚMERO DE DIBUJO: 1 / REV.: 1 / ESCALA: 1:2 / PAGINA: 5





A - A
1:3

VISTA ISÓMETRICA

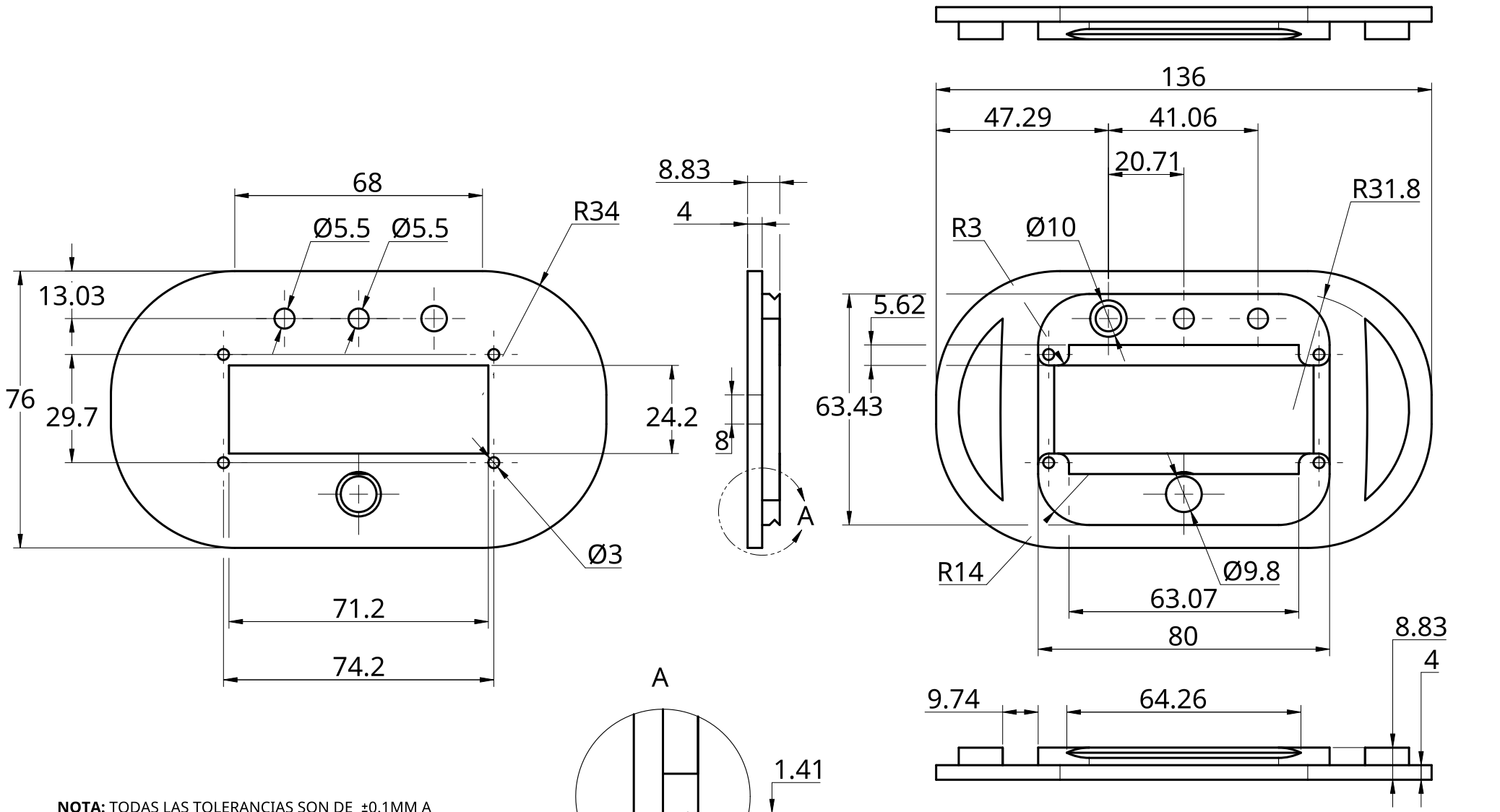


1:3

NOTA: LA DISTANCIA ENTRE CENTROS DE LOS AJUEROS INFERIORES ES DE 109MM, CON UN DÍAMETRO DE 10MM

NOTA: TODAS LAS TOLERANCIAS SON DE ±0.1MM A MENOS QUÉ SE INDIQUE LO CONTRARIO

CELOVENDO.COM			TÍTULO: PEDESTAL PARA DESPACHADOR
TERCER ÁNGULO DE PROYECCIÓN	ITERACIÓN: 1	DISEÑADOR: JAVIER RIVERA U.	DESCRIPCIÓN / INSTRUCCIONES DE ENSAMBLE: PEDESTAL SOBRE EL CUAL SE APOYA EL DISPOSITIVO DURANTE SU USO
	FECHA: 01-12-2022	AUXILIAR: -	
	MATERIAL: PLA	MASA: 60g	TAMAÑO: CARTA / NÚMERO DE DIBUJO: 1 / REV.: 1
TODAS LAS DIMENSIONES EN MILIMETROS A MENOS QUÉ SE INDIQUE LO CONTRARIO	ACABADO: IMPRIMIR SOBRE PLACA DE VIDRIO PARA OBTENER UN ACABADO LISO. NO REQUIERE SOPORTES		ESCALA: 1:3 / PAGINA: 6

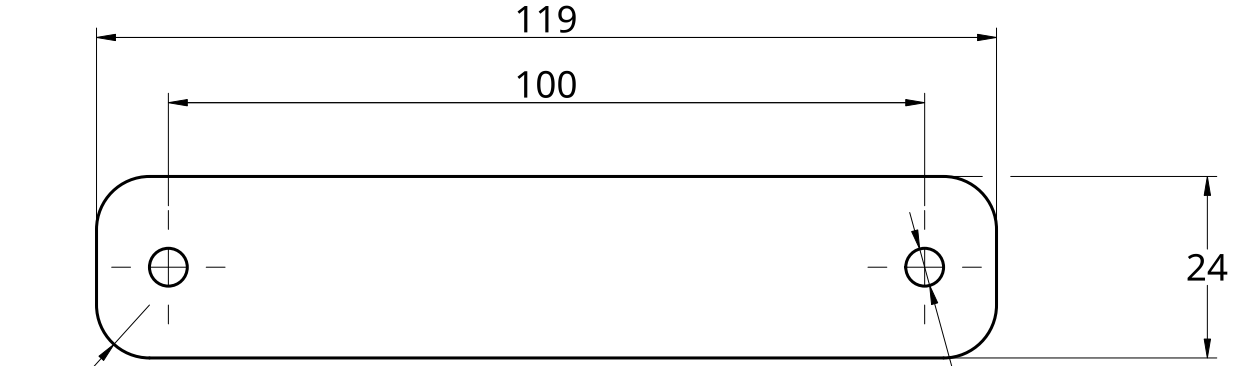


NOTA: TODAS LAS TOLERANCIAS SON DE ±0.1MM A MENOS QUÉ SE INDIQUE LO CONTRARIO. LAS RANURAS DE CONEXIÓN POR AJUSTE A PRESIÓN TIENEN LAS MISMAS DIMENSIONES DEL ACERCAMIENTO "A".

CELOVENDO.COM

TERCER ÁNGULO DE PROYECCIÓN	ITERACIÓN: 1	DISEÑADOR: JAVIER RIVERA U.	TÍTULO: TAPA FRONTAL- DESPACHADOR
	FECHA: 01-12-2022	AUXILIAR: -	DESCRIPCIÓN / INSTRUCCIONES DE ENSAMBLE: TAPA FRONTAL CON AJUSTE A PRESIÓN A LA CARCAZA, EN ESTA PARTE SE ENSAMBLAN LOS COMPONENTES.
	MATERIAL: PLA	MASA: 60g	
TODAS LAS DIMENSIONES EN MILIMETROS A MENOS QUÉ SE INDIQUE LO CONTRARIO	ACABADO: IMPRIMIR SOBRE PLACA DE VIDRIO PARA OBTENER UN ACABADO LISO. NO REQUIERE SOPORTES		Tamaño: CARTA · NÚMERO DE DIBUJO: 1 · REV.: 1 · ESCALA: 1:2 · PAGINA: 7

1:1



1:1

NOTA: TODAS LAS TOLERANCIAS DE ±0.1MM A MENOS QUÉ SE INDIQUE LO CONTRARIO

CELOVENDO.COM			TÍTULO: EXTRUSIÓN PARA MONTAJE INVERTIDO
TERCER ÁNGULO DE PROYECCIÓN	ITERACIÓN: 1	DISEÑADOR: JAVIER RIVERA U.	DESCRIPCIÓN / INSTRUCCIONES DE ENSAMBLE: EXTRUSIÓN PARA SEPARAR CARCAZA DE BASE CUANDO SE MONTA DE FORMA INVERTIDO EL DESPACHADOR
	FECHA: 01-12-2022	AUXILIAR: -	
	MATERIAL: PLA	MASA: 60g	TAMAÑO: CARTA / NÚMERO DE DIBUJO: 1 / REV.: 1
TODAS LAS DIMENSIONES EN MILIMETROS A MENOS QUÉ SE INDIQUE LO CONTRARIO	ACABADO: IMPRIMIR CON AL MENOS 60% DE INFILL YA QUE LA PIEZA QUEDA COMPRIMIDA POR LOS TORNILLOS		ESCALA: 1:2 / PAGINA: 8