ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS

IMPLEMENTACIÓN DE UN PROTOTIPO DE SISTEMA DE AUTOSERVICIO BASADO EN MICROCONTROLADORES PARA SUPERMERCADOS

TRABAJO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL TÍTULO DE TECNÓLOGO SUPERIOR EN REDES Y TELECOMUNICACIONES

Johan Sebastian Burbano Lasso
johan.burbano@epn.edu.ec

Juniel Sebastian Incerri Pinto
juniel.incerri@epn.edu.ec

DIRECTOR: ING. LEANDRO ANTONIO PAZMIÑO ORTIZ, MSC.
leandro.pazmino@epn.edu.ec

CODIRECTOR: ING. MÓNICA DE LOURDES VINUEZA RHOR, MSC.
monica.vinueza@epn.edu.ec

Quito, noviembre 2021

CERTIFICACIÓN

Certificamos que el presente trabajo fue desarrollado por el Sr. Burbano Lasso Johan Sebastián y por el Sr. Incerri Pinto Juniel Sebastian como requerimiento parcial a la obtención del título de TECNÓLOGO SUPERIOR EN REDES Y TELECOMUNICACIONES, bajo nuestra supervisión:

MSc. Leandro Antonio Pazmiño Ortiz

DIRECTOR DEL PROYECTO

MSc. Mónica de Lourdes Vinueza Rhor

CODIRECTORA DEL PROYECTO

DECLARACIÓN

Nosotros, Burbano Lasso Johan Sebastian con CI: 1727128579 e Incerri Pinto Juniel Sebastian con pasaporte: 141547761 declaramos bajo juramento que el trabajo aquí descrito es de nuestra autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y que hemos consultado las referencias bibliográficas que se incluyen en este documento.

Sin prejuicio de los derechos reconocidos en el primer párrafo del artículo 144 del Código Orgánico de la Economía Social de los Conocimientos, Creatividad e Innovación – COESC-, somos titulares de la obra en mención y otorgamos una licencia gratuita, intransferible y no exclusiva de uso con fines académicos a la Escuela Politécnica Nacional.

Entregamos toda la información técnica pertinente, en caso de que hubiese una explotación comercial de la obra por parte de la EPN, se negociará los porcentajes de los beneficios conforme lo establece la normativa nacional vigente.

Johan Sebastian Burbano Lasso

Juniel Sebastian Incerri Pinto

DEDICATORIA

Dedico este trabajo a mi madre, padre y hermanos que han solventado mis necesidades económicas y emocionales a fin de concluir exitosamente mis estudios.

Johan Sebastian Burbano Lasso

AGRADECIMIENTO

Agradezco a todos los docentes que he conocido en la carrera, de quienes no solo me llevo sus conocimientos técnicos; sino también las actitudes, aptitudes y ética para afrontar el mundo laboral de mejor manera.

Johan Sebastián Burbano Lasso

DEDICATORIA

Dedico este trabajo a mi madre y abuela, que han sido mis pilares más fuertes y motivación. Sus palabras que me han permitido ser quien soy y seguir adelante en todo lo que hago guiándome siempre por un buen camino. Por lo que doy este trabajo para agradecer la gran paciencia y amor que me proyectaron.

Juniel Sebastian Incerri Pinto

AGRADECIMIENTO

Primeramente, agradecer a mi compañero de proyecto de titulación y amigo, Johan Sebastián Burbano Lasso. Por todo el tiempo que hemos compartido para llegar a este punto e igual el de desarrollar este prototipo y como gracias a su conocimiento le permitió encontrar muchas soluciones en la realización del proyecto para lograr el resultado final.

También agradecer al director del proyecto de titulación, MSc. Leandro Antonio Pazmiño Ortiz, por todo su apoyo y conocimiento que ha servido de ayuda durante este tiempo que tomo realizar el proyecto. Sumándole agradecimiento a todos los profesores que han colaborado con mi adquisición de conocimientos durante todos los semestres.

Agradecer a este país por todas las oportunidades que me han brindado para seguir adelante y nunca rendirme.

Por último, queda agradecer a todos los que han apoyado emocionalmente como mis compañeros de la universidad, de mi familia, de mis amigos y compañeros de trabajo, ya que mediante ellos el camino fue mucho más fácil. No solo en el tiempo de desarrollo del proyecto de titulación sino durante estos últimos años de estudio de mi carrera universitaria.

Juniel Sebastian Incerri Pinto

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

RESUMEN

Las cajas de autoservicio en supermercados permiten a los usuarios escanear sus productos y realizar el pago por ellos mismos, se tratan de sistemas que buscan reducir los tiempos de estadía en los establecimientos. En la primera sección del presente documento se detalla la importancia y relevancia de este tipo de sistemas, además de fundamentos teóricos que describen los dispositivos y programas utilizados para la creación del prototipo.

La metodología del proyecto permite abordar de mejor manera cada uno de los objetivos que se persigue y explicar, de manera breve, qué se conseguirá en cada uno de ello.

En la sección de resultados y discusión se amplía y desarrolla los objetivos propuestos en la metodología. Además, se señala los requerimientos necesarios para crear el prototipo, tanto a nivel de *software* como de *hardware;* se exponen los circuitos realizados, se explica el funcionamiento del código realizado y finalmente cómo se integraron todos los dispositivos y las respectivas pruebas de funcionamiento.

Para finalizar el documento, se infieren conclusiones basadas en el cumplimiento de los objetivos propuestos; y recomendaciones de acuerdo a los desafíos que se han presentado durante la implementación del prototipo. Además de la bibliografía que sostiene los fundamentos teóricos utilizados.

PALABRAS CLAVE: *Raspberry*, *Python, SQLite,* celda de carga, caja de autoservicio.

ABSTRACT

Self-service checkouts in supermarkets allow users to scan their products and make the payment by themselves, these systems are intended to reduce the time spent in the stores. The first section of this document details the importance and relevance of this type of systems, as well as theoretical foundations that describe the devices and programs used for the creation of the prototype.

The methodology of the project allows to better address each of the objectives pursued and to explain, briefly, what will be achieved in each of them.

In the results and discussion chapter, the objectives proposed in the methodology are expanded and developed, and the requirements necessary to create the prototype, both at the software and hardware levels, are pointed out; the circuits made are presented, the operation of the code made is explained and finally how all the devices were integrated, and the respective operation tests are carried out.

At the end of the document, conclusions are inferred based on the fulfillment of the proposed objectives; and recommendations according to the challenges that have arisen during the implementation of the prototype. In addition, the bibliography that supports the theoretical foundations is included.

KEYWORDS: *Raspberry*, *Python, SQLite, load cell*, *self-checkout*.

1 INTRODUCCIÓN

Los sistemas de autoservicio persiguen disminuir los tiempos de permanencia de los clientes en los supermercados, esto se logra gracias a que en la actualidad muchos de los pagos se realizan con tarjetas de crédito o débito y a través de medios electrónicos o aplicaciones como *Payphone, PayPal* y similares. La factibilidad de estas cajas de autoservicio se basa en que cada vez se usa menos dinero físico o efectivo, esto permite que los pagos lo puedan realizar los mismos clientes sin intervención del personal de un establecimiento.

Las cajas de autoservicio prometen agilizar el tráfico de personas en horas picos de los supermercados y esto es muy evidente, en aquellas horas puntuales donde exista aglomeración de personas, estas cajas son la mejor opción para evitar filas y agilizar el proceso de compra.

Estos sistemas, desde la perspectiva de las cadenas de supermercados y tiendas, deben ser vistos como una inversión a largo plazo porque requieren de una cuantiosa inversión inicial; pero en el transcurso del tiempo, los clientes usarán con más frecuencia estas cajas de autoservicio lo que permitirá usar el personal de las empresas de manera más eficiente. Además, invertir en cajas de autoservicio aporta un factor diferencial a las empresas que las adquieren.

También se debe decir que las cajas de autoservicio no son un concepto nuevo y no han llegado para sustituir las cajas tradicionales operadas por talento humano, sino que auxilian y complementan al sistema clásico de cajas para evitar aglomeraciones y los usuarios pueden reducir sus tiempos de permanencia en un local. Además de incentivar el uso de pagos electrónicos y tarjetas, este último punto es muy relevante debido al contexto pandémico que se realizó este prototipo en donde se vela por el cumplimiento de medidas de bioseguridad como el distanciamiento social, disminución del contacto humano, desinfección de superficies, entre otras medidas similares.

1.1 Objetivo general

Implementar un prototipo de sistema de autoservicio basado en microcontroladores para supermercados.

1.2 Objetivos específicos

- Identificar los requerimientos necesarios para la implementación del prototipo.

- Analizar el *hardware* adecuado según los requerimientos necesarios para el prototipo.
- Desarrollar el *software* del prototipo.
- Construir el prototipo de autoservicio.
- Realizar pruebas de funcionamiento del prototipo.

1.3 Fundamentos

Placas de desarrollo

Raspberry Pi 4 *model B*

Raspberry Pi 4 es un pequeño computador contenido en una placa de tamaño reducido, muy utilizado para aplicaciones de domótica, levantar servidores ligeros, centro multimedia, desarrollo de programas multipropósito, entre otros fines [1].

Se diferencia de un ordenador habitual por sus cuarenta pines *General Purpose Input/Output* (GPIO) los cuales permiten ser programados y montar una gran variedad de sensores [2].

Al tratarse de un minicomputador, requiere de un sistema operativo que cumpla las actividades básicas para su funcionamiento, los sistemas operativos desarrollados para *Raspberry* son distribuciones de *Linux*; gracias a ello, es posible realizar grandes proyectos siempre y cuando el *software* y *hardware* lo permitan [3].

Arduino

Arduino es una placa de desarrollo la cual se programa en lenguaje de alto nivel, cuenta con una comunidad grande que proporciona soporte continuo, tiene muchas librerías, precios accesibles y un abanico de sensores y módulos que se ofrecen en el mercado para implementarse con *Arduino* [4].

Arduino crea una variedad de modelos de placas de acuerdo con las necesidades de un proyecto, entre un modelo y otro varía su cantidad de memoria, procesador y cantidad de pines de propósito general; *Arduino* está encaminado a proyectos pequeños y como puerta de entrada a la programación y electrónica [5].

Para establecer la conexión con sensores y periféricos se utilizan los pines incorporados en la placa, *Arduino* no cuenta con entradas *Universal Serial Bus* (USB) y, en las versiones más económicas, no dispone de comunicación inalámbrica; no obstante, es posible acoplar estas opciones a través de *shields* o módulos. *Arduino* solo puede ser

programado en un lenguaje similar a C++ a través de su *Interface Development Environment* (IDE) [5].

Lenguajes de programación

Python

Python es un lenguaje de programación interpretado de alto nivel con capacidad de ser orientado a objetos y enfocarlo al desarrollo de interfaces gráficas denominadas *Graphical User Interface* (GUI) [6].

Gracias a la gran popularidad y sencillez de este lenguaje de programación, se han desarrollado muchos entornos de desarrollo denominados *Integrated Development and Learning Environment* (IDLE) que es *software* utilizado para escribir código, los entornos difieren unos de otros, pero el lenguaje de programación es común [7]. Además, posee una librería estándar muy amplia y multiplataforma [6].

Java

Java es un lenguaje de programación orientado a objetos en donde se utilizan librerías creadas por la comunidad a las cuales también se las denominan módulos. Es un lenguaje de propósito general para desarrollar interfaces gráficas [8]. Además, se trata de un lenguaje compilado y puede ser ejecutado en todas las plataformas compatibles con *Java* [9].

Las aplicaciones *Java* una vez compiladas, pueden ser ejecutadas en cualquier máquina virtual *Java* (JVM) independientemente de la arquitectura de la computadora. La sintaxis de *Java* es similar a C y C++ pero más fácil de programar, es decir, se lo clasifica como lenguaje de alto nivel [9].

Desarrollo de interfaces gráficas

Tkinter

Es el módulo para crear una GUI integrada en *Python*, en ella se incluyen elementos gráficos a los que se denominan *widgets,* tales elementos son: ventanas, íconos, botones, menús, etiquetas, entre otros [10], *tkinter* es una alternativa a la línea de comandos, busca ser más amigable con el usuario final utilizando los elementos o *widgets* anteriormente mencionados [11]. La librería *tkinter* es adecuada para desarrollar interfaces gráficas pequeñas y es preinstalada por defecto con *Python* [10].

Sistema operativo

Raspberry Pi OS (Raspbian)

Raspberry Pi OS es un sistema operativo basado en *Debian* que, a su vez, está basado en *Linux* [12]. Es el *software* más recomendado para ser montado en una *Raspberry Pi* ya que ha sido desarrollado específicamente para los productos *Raspberry* y cuenta con una comunidad activa brindando soporte continuo [13].

Se ofrecen varias versiones de *Raspbian* en la *web* oficial de *Raspberry*, desde el manejo únicamente por consola, hasta el manejo por interfaz gráfica, inclusive con aplicaciones preinstaladas como: el lenguaje de programación *Python* [13], la plataforma *WordPress* para convertir la placa en un servidor *web*, *Kodi* que permite transformar la *Raspberry* en un centro multimedia, y otras funcionalidades instaladas por defecto con *Raspberry Pi OS* en su versión más completa [3].

Bases de datos

SQLite3

SQLite3 es una librería escrita en lenguaje de programación C para emplearse en bases de datos, es de dominio público, de código abierto y puede ser utilizada de manera privada y comercial [14].

Las bases de datos *SQLite* son almacenadas de manera local, esta característica permite utilizar recursos físicos propios como discos duros o unidades de estado sólido, así se garantiza que la información se encuentra alojada dentro de la organización. *SQLite3* no permite el modelo cliente servidor y solo admite cinco tipos de datos: *integer, numeric, text, real, blob;* y está enfocado para pequeños proyectos [15].

MySQL

Se trata de una base datos comercializada por *Oracle,* enfocada a albergar grandes cantidades de información, permite almacenar archivos y documentos de distintos tipos y se basa en un modelo cliente – servidor [16].

Lector de códigos de barras

Pistola lectora de códigos con pulsador *3nstar sc050*

El lector de código de barras *sc050* es un dispositivo de entrada al segmento de lectores de pistola muy popular para pequeños negocios con un precio asequible y una calidad muy buena; por ser un dispositivo muy comercializado, existe una gran cantidad de foros y soporte en línea [17].

Su configuración se realiza a través de los códigos de barra de configuración provistos en el manual de usuario. Su comunicación es a través de USB [18].

Sensor de peso

Celda de carga

Una celda de carga es un sensor que se deforma ligeramente en función del peso de un objeto, siendo este el principio básico para crear una balanza. Las celdas de carga se comercializan con geometrías variadas como en forma de viga o barra, en forma de parche o rectangular, en forma de S, entre otras [19].

Funcionamiento

El funcionamiento de la celda se basa en la deformación para convertir señales mecánicas (fuerza) a señales eléctricas (voltaje). Para realizar esa conversión, la barra se ha equipado con medidores de deformación que son las piezas electrónicas que receptarán el esfuerzo aplicado sobre la barra, la ubicación de los medidores de deformación (*strain gauge*) se ubican en cuatro puntos para las celdas tipo viga, tal como lo muestra la Figura 1.1, para fácil entendimiento, estos medidores representan un puente de *Wheatstone* formado por cuatro resistencias que se presenta en la Figura 1.2, con un voltaje de entrada constante; al aplicar una fuerza, se modifica el valor de las resistencias del puente y esas variaciones inciden en el voltaje de salida [19].

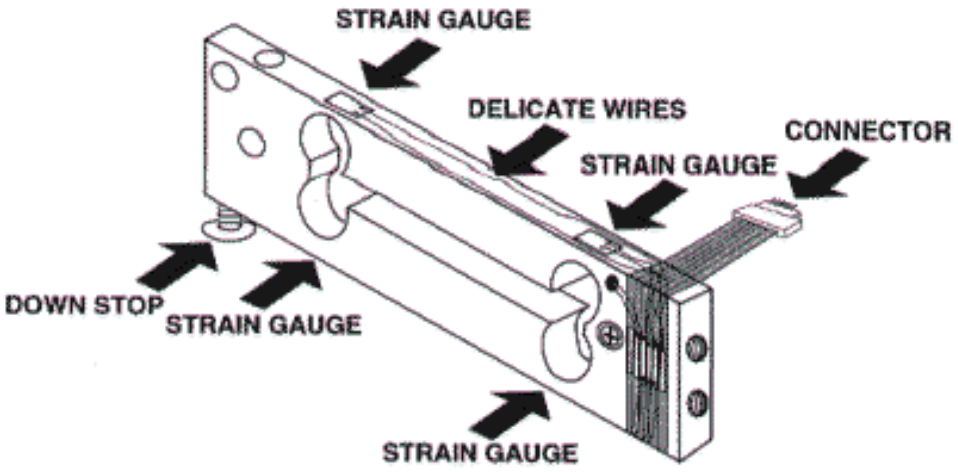


Figura 1.1 Partes de la celda de carga [19]

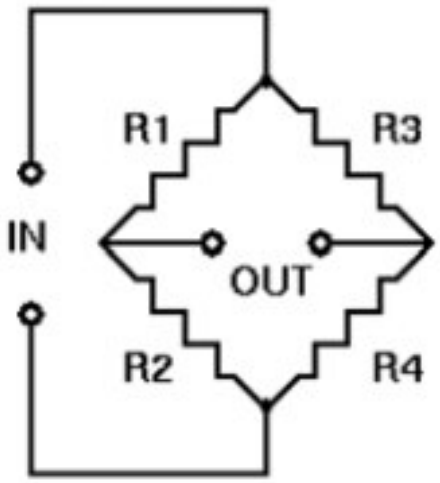


Figura 1.2 Puente de *Wheatstone* [19]

HX711

El HX711 es un amplificador de señal que permite leer los valores entregados por la celda de carga [20], incorpora un integrado el cual es un conversor de analógico a digital con resolución de 24 *bits* [21]. No requiere de programación inicial, pero puede ser configurado si se desea recolectar valores precisos. Existen varias librerías escritas para conectar el HX711 a cualquier microcontrolador con pines GPIO, utiliza 2 pines para la alimentación, y 2 pines más para disponer de una señal de reloj y el envío de datos [20].

2 METODOLOGÍA

2.1 Descripción de la metodología usada

La metodología basada en proyectos es una metodología ágil usada para solventar retos en la implementación de proyectos, se caracteriza por manejar un esquema prueba y error constante, este modelo insta a que cada modificación realizada debe ser oportunamente verificada y que su funcionamiento sea adecuado; e integra las competencias y conocimientos técnicos de sus integrantes.

Objetivo 1

Se realizará un análisis en función de los requerimientos del prototipo a implementarse; utilizando programación de alto nivel, se diseñará un sistema que guarde los datos que permitan relacionar a cada producto con su identificación única y su peso.

El prototipo será capaz de pesar los productos que ya hayan sido registrados en la base de datos, de tal modo que se pueda relacionar el peso con la identificación única del producto, la información de mayor relevancia se indicará al usuario.

Además, todos estos dispositivos a utilizarse deben ser acoplados en una estructura o armazón, a fin de dar un tratamiento estético.

Objetivo 2

En función de los requerimientos encontrados, se seleccionará un microcontrolador, donde se podrán programar las funciones de control y la base de datos necesaria para relacionar a cada producto con su correspondiente identificación única y peso.

El *hardware* permitirá visualizar el detalle de los productos a comprar y; mediante el uso de sensores, se tomará el peso de los productos y se podrá leer a cada producto por medio de su identificación única.

Se construirá un armazón a fin integrar los dispositivos del prototipo.

Objetivo 3

Se desarrollará el código para el microcontrolador, así como la creación y el manejo de una base de datos. Se empleará programación de alto nivel en microcontroladores para facilitar este proceso. Para un fácil entendimiento del código utilizado en el microcontrolador y la base de datos se emplearán diagramas de bloques; el programa se encargará de leer el identificador del producto escogido por el cliente, ese dato

rescatado se buscará en la base de datos de todos los productos, se utilizará esa información para que sea comunicada al sensor de peso y de esta manera, deberá existir concordancia con los artículos adquiridos por el cliente y el peso total de los mismos.

Objetivo 4

Se construirán placas *Printed Circuit Board* (PCB) que serán perforadas y soldadas a los diferentes circuitos electrónicos. Una vez completadas las conexiones, se diseñará el armazón o estructura más adecuada que albergue los dispositivos, el armazón deberá mostrar cierta resistencia y rigidez ya que los clientes lo manipularán de distintas maneras. La rigidez dotará de firmeza y durabilidad al sistema de autoservicio.

Objetivo 5

Al momento en que se encuentren instalados los dispositivos en la estructura, se verificará el funcionamiento correcto de *hardware* y *software*, las pruebas verificarán la correcta lectura de los productos almacenados en la base de datos mediante los parámetros establecidos en el código, a fin de identificar posibles fallas en el prototipo y obtener un funcionamiento satisfactorio del mismo.

3 RESULTADOS Y DISCUSIÓN

Las estaciones de autoservicio son sistemas que se implementan en supermercados y tiendas para que los propios clientes puedan servirse a sí mismos en la experiencia de pago; es decir, los clientes pueden acercarse a estas estaciones, escanear sus productos y realizar el pago por ellos mismos, esta solución permite evitar las filas en supermercados y agilizar el proceso de compra.

3.1 Identificación de los requerimientos necesarios para la implementación del prototipo

Para afrontar el problema, se partió de una investigación relacionada a los sistemas de cobro automáticos que existen actualmente para supermercados y con eso, dar paso a la selección de dispositivos requeridos y diseño.

Una vez que se determinó el sistema a diseñar, se procedió a la investigación de las necesidades a considerar para cubrir con los requerimientos del prototipo.

Investigación previa

La primera etapa analiza el porqué de los sistemas de pago automáticos actuales.

Hoy en día se puede divisar los constantes avances en nuevas tecnologías informáticas y electrónicas, lo que ha impulsado la tendencia de automatización de procesos. Por este motivo, las empresas se esfuerzan por implementar sistemas aplicados a los puntos de venta, a fin de reducir costos y tiempos más cortos de estadía de los clientes en los locales. Las tendencias actuales en los sistemas de pago de los supermercados indican que se volverán más automatizados y requerirán menos empleados [22].

Detalles del sistema

En el sentido económico, la alta competitividad en el campo de establecimientos y cadenas de supermercados se ha convertido en un motor para mejorar en creatividad e innovación tecnológica a fin de diferenciarse de sus competidores. Las empresas requieren utilizar la tecnología para mejorar el servicio al cliente y asegurarse de que salgan de las instalaciones lo más satisfechos posible. Para ello, se debe considerar los aspectos que más valoran los usuarios en su experiencia de supermercado.

Un usuario de supermercado valora diferentes aspectos que se encuentran resumidos en la Tabla 3.1.

Tabla 3.1 Características valuadas por usuarios [23]

Área	Características
Instalación	Conveniencia Horario
Atención al público	Personal suficiente Poco tiempo de espera Confianza
Servicios adicionales	Pago con tarjeta Caja rápida

Una vez identificadas las características y aspectos que pueden implementarse en el prototipo de caja de autoservicio, se determinan las especificaciones del sistema a diseñar.

- Autónomo: un sistema que no requiera la intervención de los empleados. Los clientes tendrán el poder de realizar todo el proceso de compra por sí mismos.
- Rápido: el sistema de autoservicio busca agilizar la experiencia de compra y evitar las filas.
- Facilidad de uso: al diseñar el sistema, se debe considerar la sencillez para el consumidor final; ya que un análisis muestra la preferencia hacia las cajas de autoservicio alrededor del mundo, donde la complejidad del sistema determina si un usuario lo utilizará o no [24]. Por lo tanto, no debe introducir por error un sistema que, en última instancia, haga que los clientes dediquen más tiempo a realizar compras debido a su complejidad.
- Rentable a largo plazo: desde la perspectiva de los supermercados, supone una inversión inicial que será recuperada a largo plazo ya que se trata de un monto alto, pero supone una gestión más eficiente del personal o reducción del mismo.

Diseño propuesto

El funcionamiento del sistema se basa en torno a dos pilares: uno es el sistema de lectura de códigos de barras y otro es el sistema de control de peso. Considerando esas dos pautas, los clientes son responsables de leer los códigos de barras de los productos cuando realizan una compra. La lectura del producto puede realizarse mediante una pistola láser de escaneo de códigos de barras.

El control de peso es un sistema que permitirá validar la concordancia entre el producto escaneado por el cliente y el peso del mismo, para este propósito se puede utilizar una báscula o balanza con una precisión y carga máxima adecuada.

Cada producto debe, obligatoriamente, contar con un peso, de esta manera se cumplirá con la concordancia producto – peso, para este fin se puede utilizar una base de datos que almacene esta información; además, el responsable de la tienda o cadena de supermercados deberá añadir, eliminar o modificar estos productos a la base de datos de acuerdo al *stock* de los mismos.

El cliente debe visualizar cada una de las instrucciones y pasos para usar satisfactoriamente el prototipo, una pantalla es la solución en donde se podrá ver la lista de productos escaneados, valor total a pagar, valor individual de cada producto e interfaz de pago. No es necesario que la pantalla o monitor sea de tipo táctil, con botones físicos se pueden ejecutar acciones dentro de la interfaz gráfica.

Existen distintos sistemas de pago que permiten realizar transacciones instantáneas, pero requieren afiliación o subscripción a los mismos, por ello, se puede simular un sistema de pago utilizando un sensor que permita validar que la transacción se realizó satisfactoriamente. El diagrama de bloques de la Figura 3.1 simplifica el entendimiento del diseño propuesto.

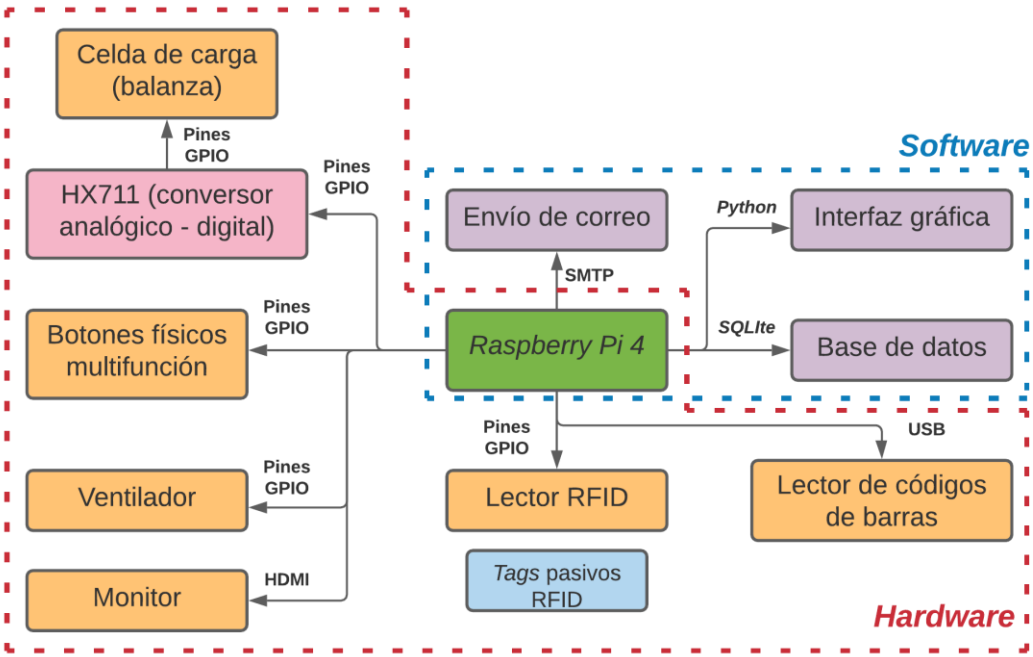


Figura 3.1 Diagrama de bloques de *hardware* y *software*

Seguridades adicionales

De manera obligatoria, se debe acotar que este sistema presenta restricciones tales como: la cantidad máxima de productos que puede soportar la balanza, productos con pesos muy similares, productos muy ligeros que podrían evadir el control de peso.

Para saldar estas restricciones se pueden añadir sistemas de seguridad externos que complementen el sistema de autoservicio como: cámaras de seguridad, sensores antirrobo ubicados en las entradas y salidas de la tienda, y personal reducido que vele por el uso correcto de las estaciones de autoservicio por parte de los clientes.

3.2 Análisis del *hardware* adecuado según los requerimientos necesarios para el prototipo

La selección del equipo se realizó considerando siempre la premisa de un prototipo sencillo de utilizar para el usuario, interfaces intuitivas y de fácil manipulación.

El usuario debe interactuar con un lector de códigos de barras, una balanza, un simulador de pago y 3 botones físicos para ejecutar las posibles acciones que se muestren en una pantalla.

Selección del sensor de lectura de códigos de barras

Por medio de un lector de código de barras, es posible recuperar el código de un producto tomado por el cliente y, posteriormente, colocar el artículo en un sensor de peso; estos 2 sensores deben guardar concordancia entre ellos, pues cada artículo escaneado guarda relación con un peso determinado, este peso debe ser validado por la balanza para lograr esa concordancia.

Para la selección del escáner de códigos de barras se consideró varias opciones ofrecidas en el mercado, donde se identificaron dos grandes grupos de sensores: lectores unidireccionales y lectores omnidireccionales, la Tabla 3.2 resume las características más relevantes de estos grupos:

Tabla 3.2 Comparativa de lectores unidireccionales y bidireccionales [18] [17]

Parámetros	Lectores unidireccionales	Lectores Omnidireccionales
Principio de funcionamiento	Su haz de luz debe estar relativamente alineado con el código de barras para ser leído	Su haz de luz no requiere alinearse, solo necesita la presentación de la etiqueta del código de barras para su lectura
Forma de obtener la lectura de código	Generalmente, requieren de la activación de un botón o gatillo para realizar la lectura	No poseen gatillo o botón a ser pulsado para realizar la lectura
Rendimiento	Escanea una pequeña cantidad de códigos en un intervalo de tiempo	Escanea una gran cantidad de códigos en un intervalo de tiempo
Costo	Precios accesibles	Precios altos, tienden a costar dos o tres veces más que un lector unidireccional

Hay que destacar la ventaja del lector omnidireccional de no requerir que se presione un botón o gatillo para realizar la lectura de un código, sin embargo, los lectores unidireccionales modernos pueden ser configurados en modo manos libres (*hands-free*), de esta manera, se evita pulsar el gatillo y se realiza la misma función por un menor precio.

Además, el prototipo presente se enfoca a clientes con pocos productos a comprar, entre 1 y 15 artículos es el rango adecuado ya que escanear más ítems puede resultar una tarea tediosa para el cliente y deterioraría la experiencia de compra. Adicionalmente, la velocidad de lectura del lector tiene bajo impacto en el prototipo porque el diseño propuesto exige un intervalo de tiempo para validar el peso después que se ha escaneado un producto. Por estas razones, se ha escogido un lector unidireccional, el cual será mucho más económico y podrá ser configurado posteriormente acorde al prototipo [18].

Los lectores de códigos de barras llevan determinadas configuraciones de fábrica, pero estas son modificables a través de los códigos comandos, un código comando es un código de barras reservado para cumplir funciones de configuración [18]; el presente prototipo requiere que la pistola lectora cumpla con las siguientes características:

- Al terminar de leer un código, se debe presionar automáticamente el botón *Enter* para hacer un salto de línea, es primordial que esto suceda ya que el *Enter* es la señal para que el programa escrito en *Python* entienda que se concluyó de leer el código.
- Ya se seleccionó anteriormente el tipo de código lineal que se usará, el *code-128,* entonces se debe verificar que la pistola sea capaz de leer este tipo de código.
- El lector no debe ser manipulado físicamente; es decir, el cliente no puede tomar la pistola, presionar el gatillo y escanear; sino que el lector debe estar fijo y en lectura continua para que el cliente solo presente el producto delante del lector y el código sea leído. A este modo se denomina *hands-free* o lectura en modo continuo.
- Entre la lectura de un código y el siguiente código, debe existir un retraso o *delay* para evitar que el lector realice lecturas no deseadas, este puede ser de 1 a 2 segundos.

Selección de la balanza

La única alternativa ampliamente utilizada para recuperar el peso de objetos son las celdas de carga acompañadas de su conversor y amplificador HX711, estas celdas de carga son compatibles para *Arduino* y *Raspberry,* se comercializan de acuerdo con el peso máximo que soportan; para aplicaciones de poco peso, existen de 1, 5 y 10 (kg), y para aplicaciones de pesos grandes: 20, 50, 100 (kg). Pero se debe considerar que, si la celda soporta una carga mayor, su precisión será menor.

La balanza del prototipo está destinada a ser usada por clientes con pocos productos, por ello, la celda más conveniente es de 10 (kg), esta celda es el equilibrio entre una buena precisión y un soporte máximo de peso para un usuario con una cantidad moderada o baja de productos.

Selección de la base de datos

La base de datos se encargará de almacenar la información de cada uno de los productos, cada registro de producto debe contar con: código del producto, nombre del producto, precio, peso y nombre de una imagen en formato *.png, .jpeg* o *.jpg*; debe permitir recuperar estos datos cuando un producto sea escaneado, al igual que ser modificados, eliminados o añadir productos nuevos si así lo requiere el administrador del prototipo.

Para esta selección, se han considerado dos conocidas bases de datos compatibles con *Python* pero que presentan diferencias marcadas en la Tabla 3.3:

Tabla 3.3 Comparativa de bases de datos *MySQL* y *SQLite3* [25]

Parámetros	*MySQL*	*SQLite3*
Modelo de conexión	Cliente – servidor, es decir, requiere acceso a internet para recuperar los datos desde un servidor	Integrado en la propia aplicación, hace uso del almacenamiento del equipo donde fue instalado
Almacenamiento	Datos almacenados en un servidor	Datos almacenados de manera local, en los recursos de *hardware* que se disponga
Tipos de datos admitidos	Admite una gran variedad de archivos y tipos de datos, sean estos grandes o pequeños	Limitado a cinco tipos de datos: *integer, numeric, real, text* y *blob*
Enfoque	Para usos comerciales y empresariales	Enfocado a proyectos y comercios pequeños

SQLite3 es la mejor opción para el presente proyecto porque el almacenamiento local del prototipo así lo permite, en este caso, no es conveniente utilizar *MySQL* ya que realizar consultas de datos a un servidor externo toma un tiempo extra y existe una dependencia total de una conexión a internet; si no se cuenta con una conexión estable, no se podrá recuperar la información de las bases de datos adecuadamente.

Si bien *SQLite3* es limitado en los tipos de datos soportados, este prototipo solo requerirá datos del tipo texto, numéricos reales y almacenamiento de imágenes. Además, el prototipo cuenta con almacenamiento local suficiente para albergar la base de datos.

Selección de la placa de desarrollo

La placa de desarrollo debe contar con las capacidades para: conectar el lector de código de barras por conexión USB, acoplar la celda de carga con su amplificador a través de pines de propósito general GPIO, además de los 3 botones de interacción que también usarán estos pines y, finalmente, visualizar una interfaz gráfica a través de una pantalla que deberá conectarse a la placa de desarrollo.

La Tabla 3.4 muestra una comparación entre *Arduino* y *Raspberry* en función a las conexiones y pines que requiere el prototipo.

Tabla 3.4 Comparativa de las placas *Arduino Mega* y *Raspberry Pi* 4 [26] [4]

Parámetros	*Arduino Mega REV3*	*Raspberry Pi 4*
Procesador	*ATmega2560* de un solo núcleo	*Broadcom BCM2711* de cuatro núcleos, almacenamiento de memoria *RAM* a partir de 2 (GB)
Almacenamiento	Memoria interna de 256 (kB), expandible por memorias externas	Lo determina la memoria *microSD* que se coloque
Tipos de conexiones	54 pines *GPIO*, conexiones expandibles a través de módulos (*shields*)	40 pines *GPIO*, 4 entradas *USB,* entrada *microSD,* 2 salidas *microHDMI,* conexión inalámbrica por el estándar 802.11ac
Sistema operativo	No soportado	*Raspberry Pi OS*, entre otras distribuciones de *Linux*
Lenguaje de programación	Adaptación de *C++*	Cualquier lenguaje soportado en *Raspberry Pi OS* o *Linux*
Entorno de desarrollo (IDE)	*Arduino* IDE	Cualquier IDE soportada en *Raspberry Pi OS* o *Linux*
Precio	Precio moderado	Su precio es aproximadamente el doble que un *Arduino Mega*

Después de esta comparación, se concluye que la mejor opción es *Raspberry* con su placa de desarrollo versión 4. Este proyecto exige el uso de los GPIO, conexión USB, almacenamiento variable y conexión a un monitor por HDMI.

Arduino puede solventar estas conexiones de las que carece a través de sus *shields,* pero agrega mayor complejidad a la programación, además que el precio se incrementa y se vuelve equiparable a adquirir una placa *Raspberry.*

Arduino goza de poseer una gran cantidad de librerías para conectar cualquier sensor a través de sus pines, a diferencia de *Raspberry*, que solo en determinados lenguajes de programación se han desarrollado librerías para sensores, uno de esos lenguajes es *Python*. Ahora, una gran ventaja de *Raspberry* es que no se encajona en un solo

lenguaje de programación como ocurre en *Arduino*, *Arduino* debe usar de manera forzosa su lenguaje adaptado C++ en su entorno de desarrollo IDE *Arduino.*

Selección del *hardware* de simulación de pago

El prototipo debe validar que se ha realizado el pago de los productos tomados, para ello se ha optado por la simulación de un sistema de pago utilizando el módulo *Radio-Frequency Identification* (RFID) RC522. Se conoce que cada tarjeta RFID cuenta con un código único, se puede usar esta característica para simular un pago, en donde un *tag,* a través de su código único, sea habilitado para realizar el pago mientras que, otro *tag* con su código sea rechazado.

3.3 Desarrollo del *software* del prototipo

Selección del lenguaje de programación

El lenguaje de programación a escoger debe permitir desarrollar interfaces gráficas, ser compatible con la pistola de escáner y poseer librerías para controlar los periféricos como los botones físicos y la celda de carga con su amplificador.

Adicionalmente, el lenguaje debe manejar los requisitos anteriormente seleccionados que son: soportar *SQLite3* y ser compatible con la placa *Raspberry Pi* 4 y su sistema operativo.

En la

Tabla 3.5 se resumen los requisitos y la capacidad de 2 lenguajes de programación para solventarlos.

Tabla 3.5 Comparativa de lenguajes de programación *Java* y *Python* [27]

Parámetros	*Java*	*Python*
Compatibilidad con periféricos USB	Sí permite interactuar con dispositivos que utilizan comunicación USB	Sí permite interactuar con dispositivos que utilizan comunicación USB
Desarrollo de interfaces gráficas	Muy flexible y sencillo para crear interfaces gráficas muy llamativas	Las interfaces que se desarrollan son rudimentarias, sencillas y simples
Compatibilidad con *Raspberry Pi OS*	Compatible	Compatible
Compatibilidad con *SQLite3*	Compatible	Compatible
Manejo de pines GPIO	Limitado, no existen suficientes librerías o soporte para los pines multipropósito	Gran cantidad de librerías y módulos para controlar GPIO en *Raspberry Pi*

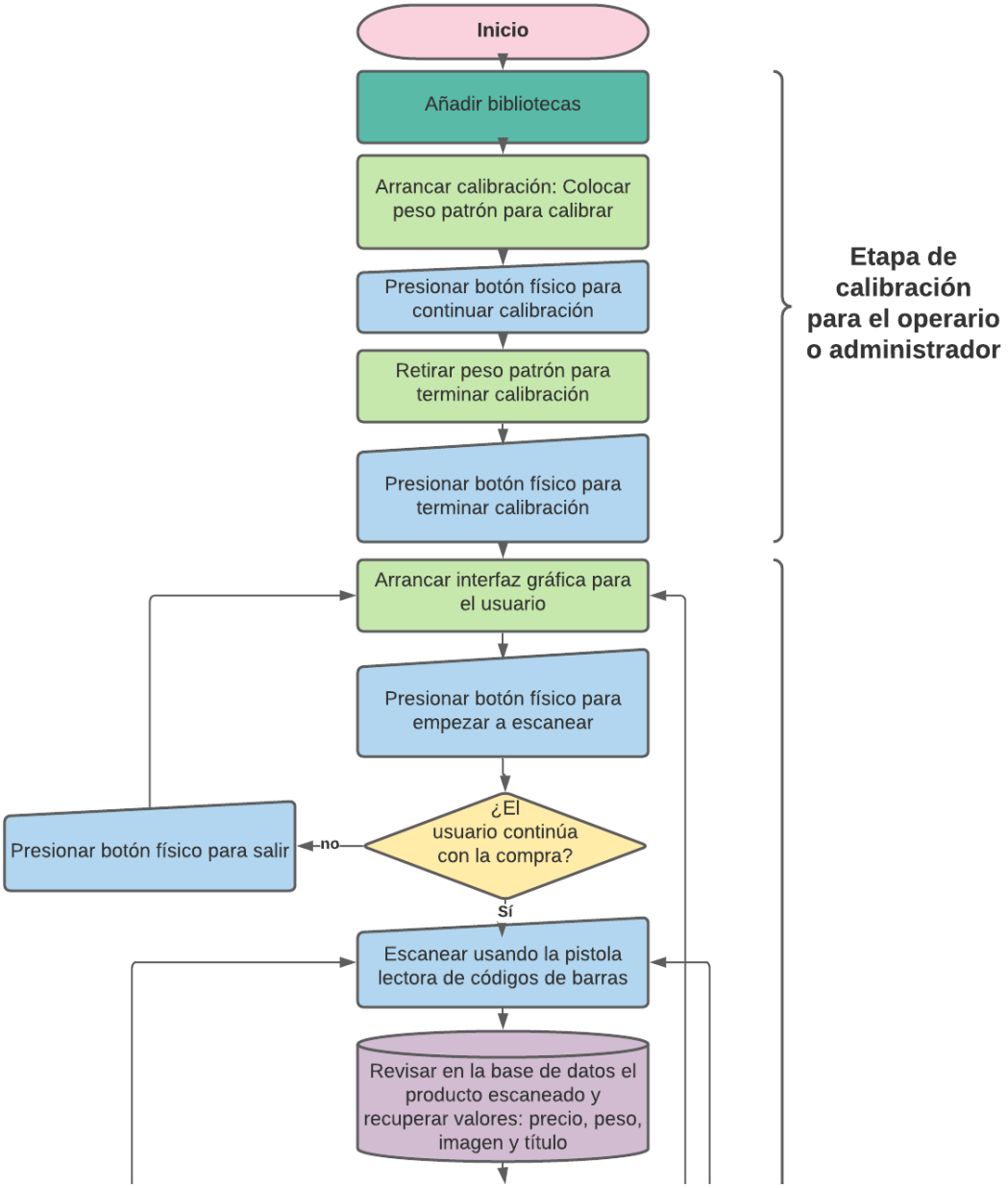
Si bien en *Python* no se programan interfaces gráficas tan atractivas como en *Java*, este prototipo requiere obligatoriamente explotar las funcionalidades de los pines GPIO; por ello, *Python* es la opción más adecuada, con una comunidad sólida aportando con librerías para manipular estos pines que permiten integrar varios sensores. En *Java* las librerías para manipular los pines son muy limitadas.

Diagrama de bloques

Para una fácil comprensión del comportamiento del programa principal se utilizarán diagramas de bloques, el código principal contiene aproximadamente 900 líneas, entonces se ha seccionado por partes y se explicará cada región del código. Además, se desarrollaron dos *scripts* adicionales, los cuales arrancan el código principal y permiten usar la balanza del prototipo de manera aislada, respectivamente. Los códigos que se explican a continuación se encuentran en la sección Código fuente.

Diagrama de bloques general del programa

En la Figura 3.2 se resume el código principal cuya función es ejecutar la interfaz gráfica que permite la interacción del cliente con el prototipo, este diagrama se basa en el código principal que se encuentra en la Figura 3.15 en la sección Código fuente.


Inicio
Añadir bibliotecas
Arrancar calibración: Colocar peso patrón para calibrar
Presionar botón físico para continuar calibración
Retirar peso patrón para terminar calibración
Presionar botón físico para terminar calibración
Etapa de calibración para el operario o administrador
Arrancar interfaz gráfica para el usuario
Presionar botón físico para empezar a escanear
¿El usuario continúa con la compra?
no
Presionar botón físico para salir
Sí
Escanear usando la pistola lectora de códigos de barras
Revisar en la base de datos el producto escaneado y recuperar valores: precio, peso, imagen y título

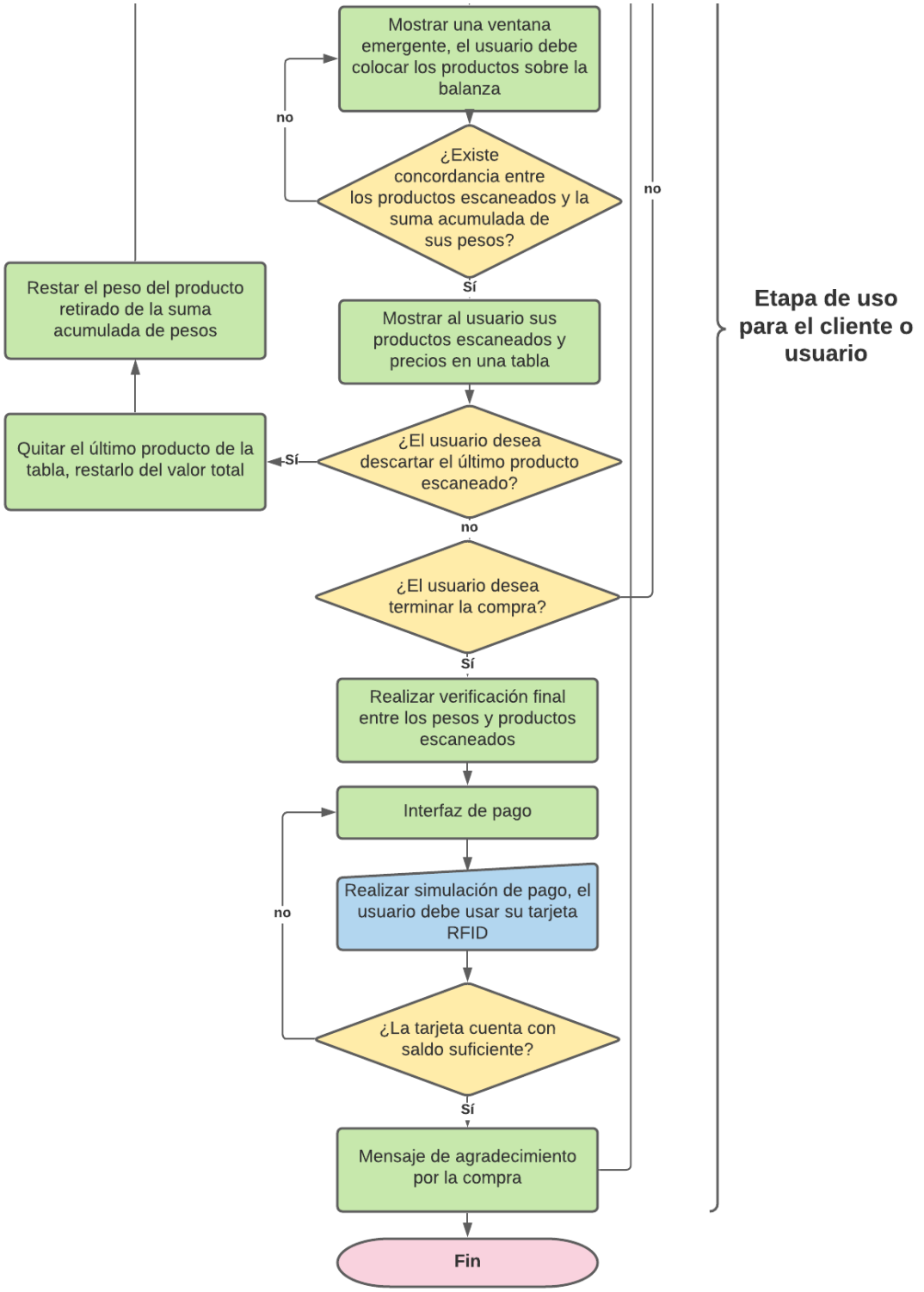


Figura 3.2 Diagrama de bloques del código desarrollado

Diagrama de bloques para la etapa de calibración

La balanza del prototipo requiere ser calibrada cada vez que se encienda, en esta sección de código se desarrollaron las líneas para que el amplificador HX711 y la celda

de carga recopilen información y la usen para realizar la calibración, estos datos y cálculos son posibles gracias a las leves deformaciones que sufren las resistencias ubicadas en la celda de carga. En la Figura 3.3 se detalla cómo funciona este proceso. Este diagrama se basa en el código de la balanza aislada que se encuentra en la Figura 3.14 en la sección Código fuente.

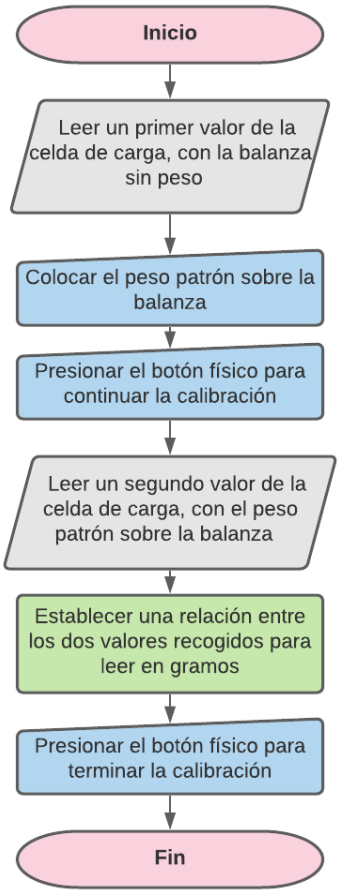


Figura 3.3 Diagrama de bloques de calibración del HX711 y la celda de carga

Diagrama de bloques para las transiciones de ventanas

En el módulo *tkinter*, las transiciones de ventanas son útiles para bloquear ciertas funcionalidades y habilitar otras. Para el prototipo, esta funcionalidad tiene mucha relevancia porque en determinados momentos es necesario que los botones físicos sean deshabilitados temporalmente, al igual que el lector de códigos de barras y el lector de tarjetas RFID. La Figura 3.4 indica de manera genérica cómo se aplica estas transiciones de ventanas. Este diagrama se basa en el código principal que se encuentra en la Figura 3.15 en la sección Código fuente.

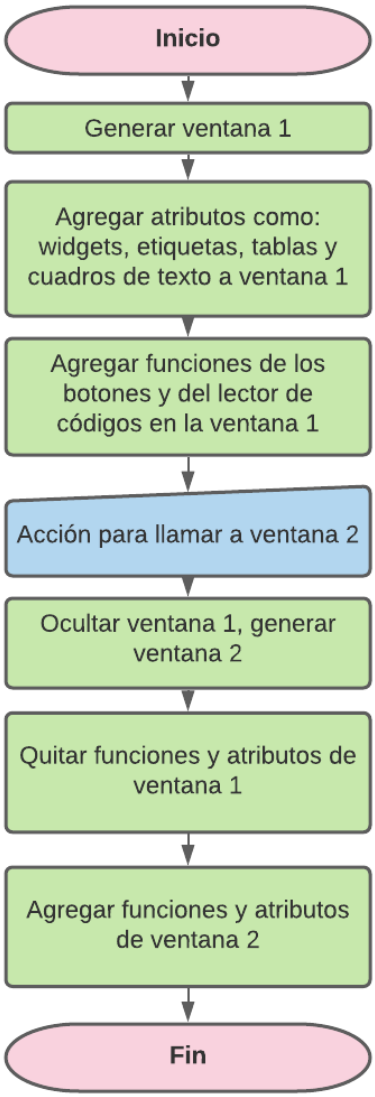


Figura 3.4 Diagrama de bloques de transiciones de ventanas a través de *tkinter*

Diagrama de bloques para la lectura de la base de datos

En la base de datos se almacenarán los datos de cada producto para ser llamados cuando se lea el código de un producto. Los datos que se deben almacenar de cada producto son: código del producto, nombre del producto, precio del producto en dólares americanos, peso del producto en gramos, nombre de la imagen del producto y existencias. La Figura 3.5 resume este proceso en diagrama de bloques. Este diagrama se basa en el código principal que se encuentra en la Figura 3.15 en la sección Código fuente.

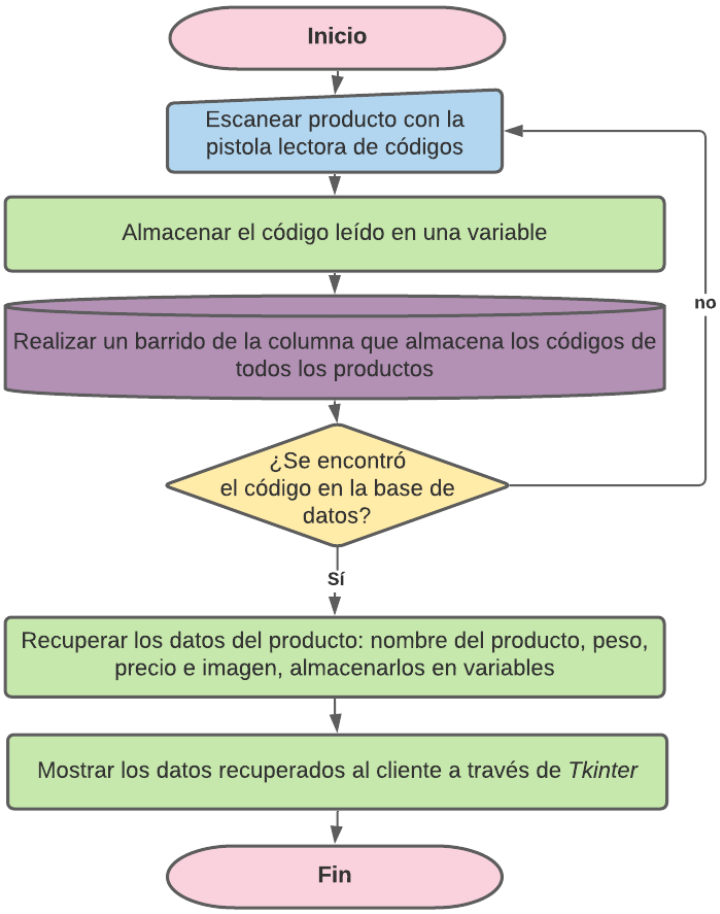


Figura 3.5 Diagrama de bloques del código leído y la base de datos

Diagrama de bloques para la validación de peso

La balanza puede leer un peso ligeramente diferente al que se ha almacenado en la base de datos; por ello, se ha determinado un margen o rango del 20% en el cual el producto será validado; la selección del ajuste del 20% está justificado en la precisión que maneja la celda de carga de 10 (kg), este sensor de peso desvía las lecturas ligeramente entre 2 y 6 (g). Adicionalmente, se debe considerar el peso de la funda donde se almacenarán los artículos la cual oscila entre 4 y 7 (g). A través del siguiente de ejemplo se asimila de mejor manera la validación de peso, si el peso de un producto es 100 (g) en la base de datos, el rango será entre 90 y 110 (g); es decir, cuando se coloque este producto de 100 (g) sobre la balanza, bastará que la balanza detecte un peso entre 90 y 100 (g) para validarlo. El diagrama de bloques de la Figura 3.6 resume este proceso.

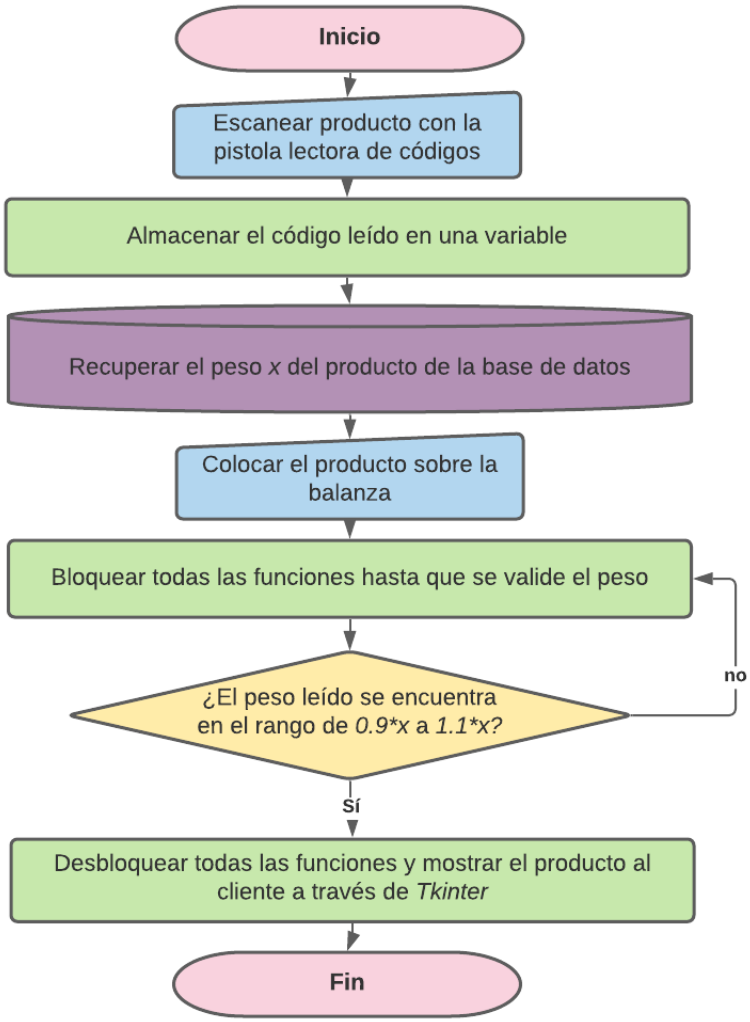


Figura 3.6 Diagrama de bloques de la validación de peso a través de un margen de tolerancia

Este diagrama está ligado al diagrama de la Figura 3.4 porque todas las funcionalidades, tanto de los botones como del lector de códigos, son bloqueadas mientras se valida el peso. Este diagrama se basa en el código principal que se encuentra en la Figura 3.15 en la sección Código fuente.

Diagrama de bloques para la validación de peso combinado

A medida que se escanean los productos, estos son apilados dentro de una funda en conjunto, por ello surgió la necesidad de realizar una última verificación de peso antes de pasar a la ventana de simulación de pago. Esta verificación es conjunta, es decir, se valida la suma de todos los pesos que se encuentran en la funda.

El diagrama de la Figura 3.7 es usado cuando la cantidad de productos de la funda actual es mayor a uno; entonces, una variable acumulativa denominada sumaPeso crecerá a medida que se escanean los productos, y la validación se realizará en base a la suma total de los pesos que están contenidos en la funda actual. Este diagrama se basa en el código principal que se encuentra en la Figura 3.15 en la sección Código fuente.

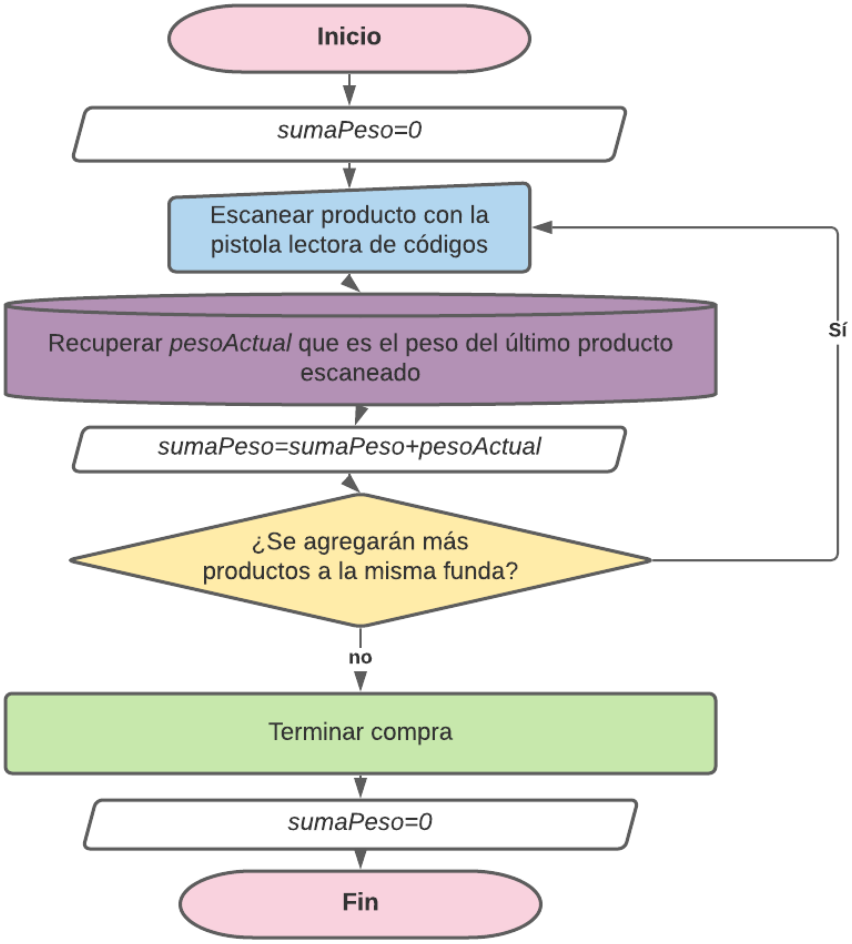


Figura 3.7 Diagrama de bloques de acumulación de peso a medida que se colocan más productos

Diagrama de bloques para una nueva funda

La opción nueva funda permite al usuario cambiar una funda llena de productos por una vacía; y de manera paralela, a nivel de código, una nueva funda implica encerar la balanza y cambiar el valor de la variable acumulativa sumaPeso a cero.

El diagrama de la Figura 3.8 muestra que, al seleccionar una nueva funda, la variable sumaPeso cambiará su valor a cero y nuevamente, este crecerá acorde a los productos

que son escaneados. Repitiendo tantas veces este ciclo como fundas necesite el usuario. Este diagrama se basa en el código principal que se encuentra en la Figura 3.15 en la sección Código fuente.

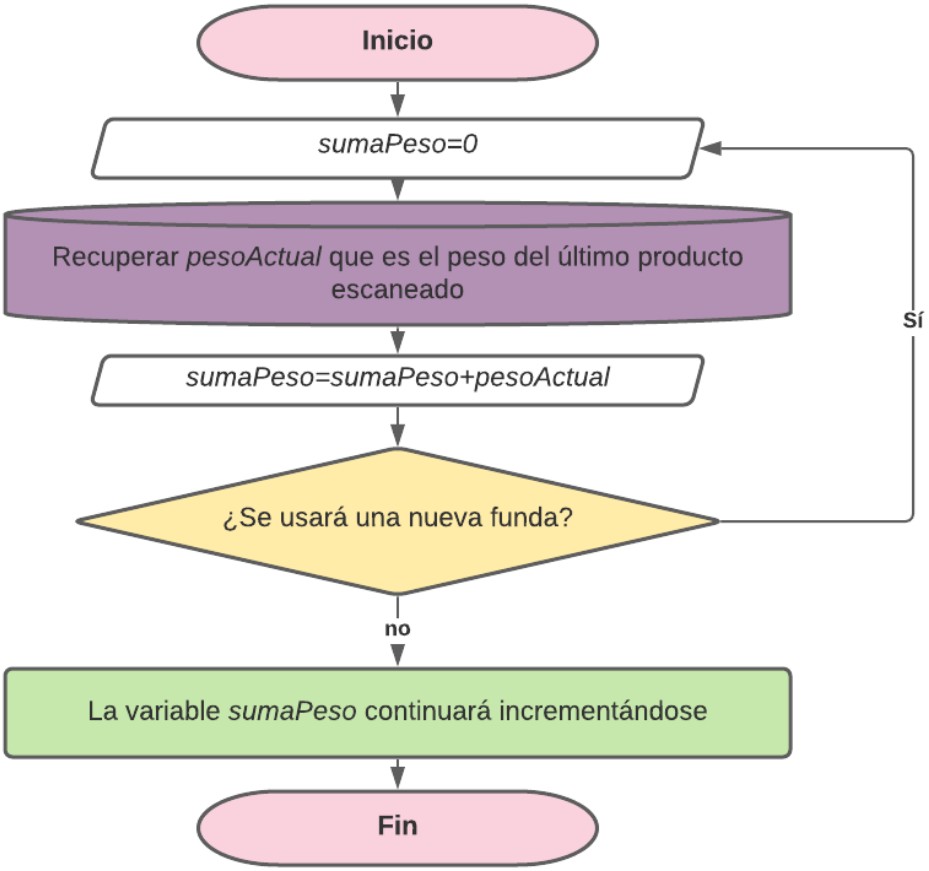


Figura 3.8 Diagrama de bloques para encerar la balanza y colocar una nueva funda

Diagrama de bloques para lector RFID

El lector RFID se encargará de realizar la simulación de pago, al contar con 2 *tags* o tarjetas pasivas, se simulará que una de ellas es una tarjeta válida para finalizar la compra, y otra simulará ser una tarjeta sin saldo y no permitirá terminar la compra.

El principio de funcionamiento del diagrama de la Figura 3.9 se basa en el número o ID único que posee cada *tag*, cuando se analiza el ID de cada una de ellas, una tarjeta será rechazada y la otra será aceptada. Este diagrama se basa en el código principal que se encuentra en la Figura 3.15 en la sección Código fuente.

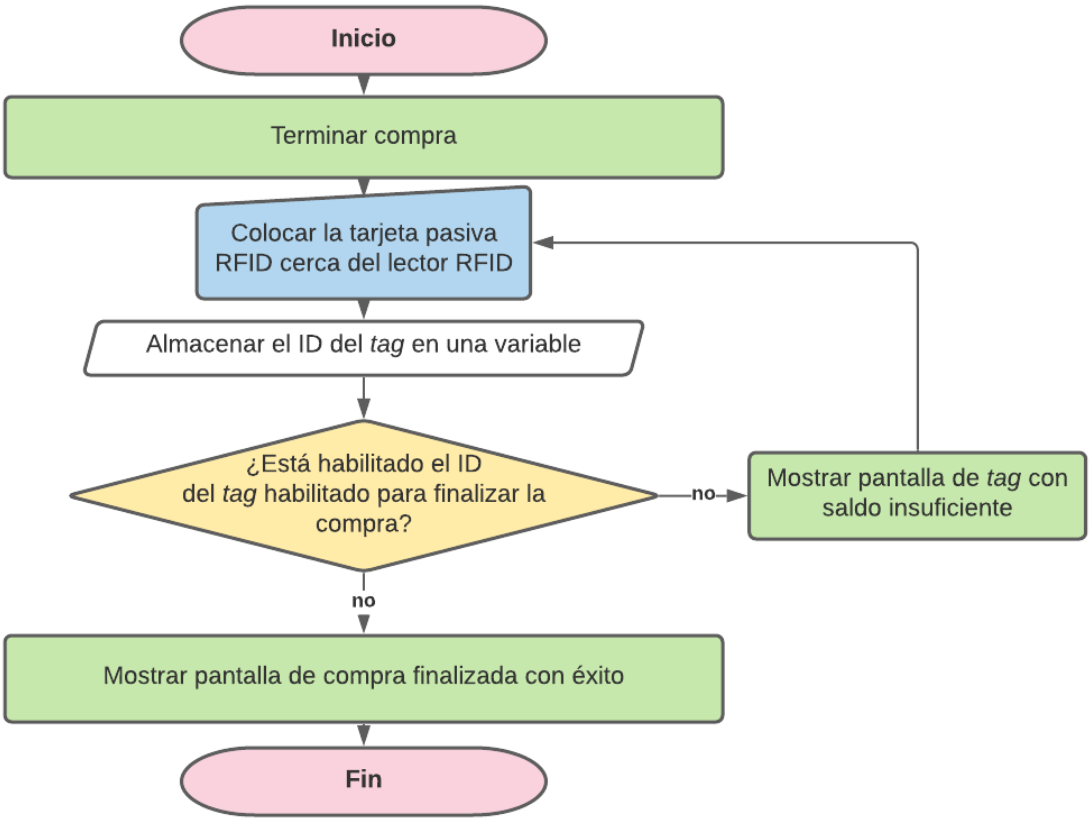


Figura 3.9 Diagrama de bloques para verificar la validez de un tag RFID a través de su ID

Diagrama de bloques para enviar correo

El envío de correo se utilizará para notificar que determinado producto se está agotando, el mensaje llegará al administrador de la caja de autoservicio.

En el diagrama de bloques de la Figura 3.10 se resume que, si la cantidad de existencias de un producto es menor o igual a 3, entonces se enviará un correo con el producto que se está agotando. Este diagrama se basa en el código principal que se encuentra en la Figura 3.15 en la sección Código fuente.

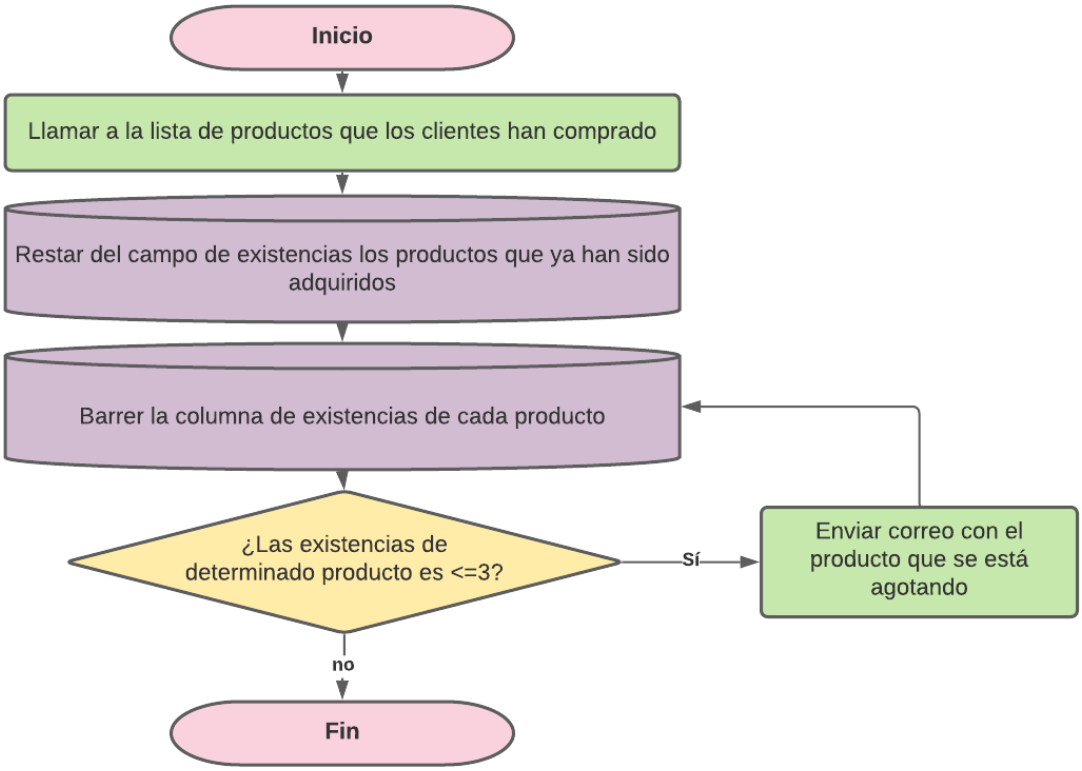


Figura 3.10 Diagrama de bloques de envío de correo en caso de que un producto se esté agotando

Diagrama de bloques para botón de apagado

Los únicos periféricos con los que cuenta el prototipo son: botones físicos para el cliente, lector RFID y pistola lectora de código de barras. Al momento de apagar la *Raspberry* se debería conectar un ratón o un teclado para realizar esta acción; por ello, se optó por implementar un pulsador físico al que solo tendrá acceso el operario del prototipo y realizar el proceso de apagado con solo pulsarlo.

El diagrama de bloques de la Figura 3.11 demuestra la prevalencia y omnipresencia del botón físico de apagado; es decir, no importa la línea de código que se encuentre ejecutando el programa, al pulsar el botón, la *Raspberry* cerrará el programa y se apagará. Este diagrama se basa en el código principal que se encuentra en la Figura 3.15 en la sección Código fuente.

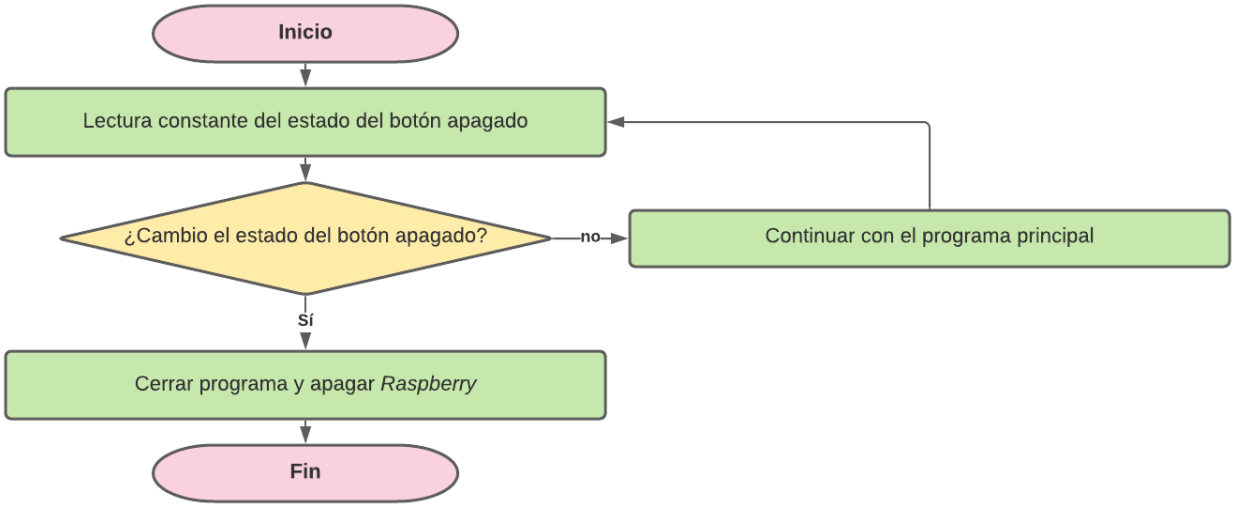


Figura 3.11 Diagrama de bloques del pulsador físico de apagado

Diagrama de bloques para el arranque del programa principal

En *Raspberry,* es posible arrancar determinados programas y ficheros al momento que esta se enciende; sin embargo, no se pudo iniciar el archivo con el *script* que contenía el programa, esto se debe a las restricciones que posee *tkinter* para ser arrancado automáticamente.

Para solventar el problema del arranque del *script* principal, se desarrolló un *script* auxiliar en *Python*, un código sencillo que sí es posible ejecutarlo en el arranque, este pequeño programa utiliza el módulo *pyautogui* que permite manipular un teclado y ratón de manera virtual, se usa esta librería para que después de conectar la alimentación de la *Raspberry* en un enchufe, el puntero se coloque sobre una terminal, ingrese a ella y escriba los comandos necesarios para arrancar al programa principal.

Gracias a este reducido programa de la Figura 3.12, se logró subsanar el problema de arranque del programa principal. Este diagrama se basa en el código para arrancar que se encuentra en la Figura 3.13 en la sección Código fuente.

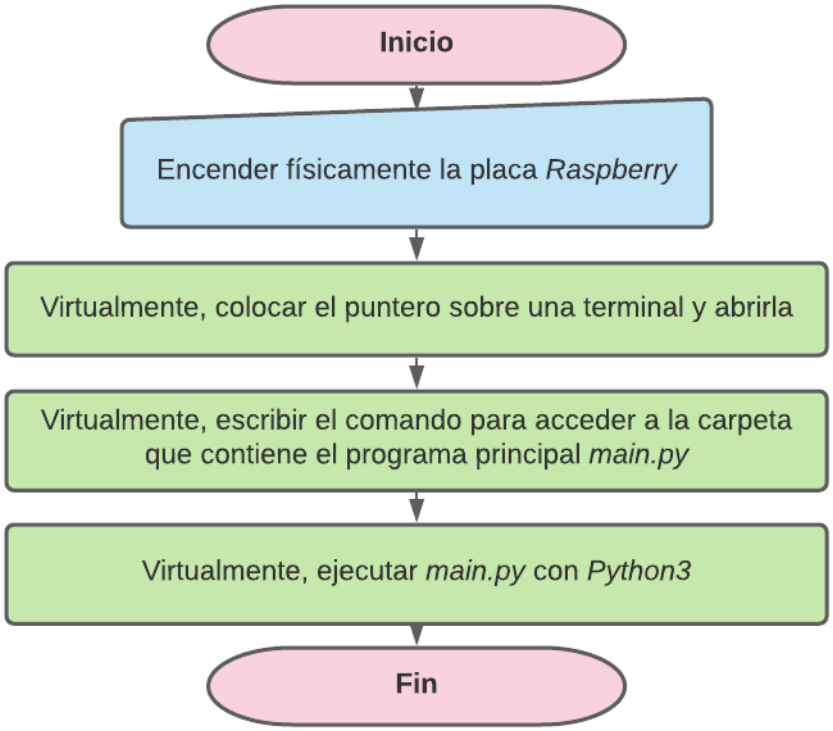


Figura 3.12 Diagrama de bloques para el arranque del programa principal

Edición de la base de datos

Modificar la base de datos es primordial para mantener al día el inventario de los productos; para ello, *SQLite3* se soporta en un editor de bases de datos de libre uso denominado *DB Browser for SQLite;* a través de este programa se puede buscar, añadir, quitar o modificar registros de una base de datos, o en sí, crear más base de datos y tablas. *DB Browser* permite realizar estos cambios a través de una interfaz gráfica fácil de usar [28].

Código fuente

Debido a la extensión del código, este se ha almacenado en un archivo de texto en la nube al que se puede acceder a través de códigos QR; además, el prototipo no se compone de un solo *script* de *Python,* sino que son 3 en total: uno de ellos es el programa principal (Figura 3.15), otro es el *script* que permite arrancar el programa principal apenas enciende la *Raspberry* (Figura 3.13) y otro es un *script* para utilizar la balanza de manera aislada (Figura 3.14).

Figura 3.13 Código para arrancar el programa principal

Figura 3.14 Código para usar la balanza de manera aislada

Figura 3.15 Código principal

Estructura de la base de datos del prototipo

Usando *DB Browser for SQLite* se pueden crear tantas bases de datos como la memoria local del dispositivo lo permita; y dentro de estas bases de datos, crear tantas tablas como se requieran. Para el presente prototipo, solo se necesitó de una base de datos con una única tabla. La Figura 3.16 es la tabla real que se ha utilizado, esta tabla se compone de 6 campos (*fields*) o columnas: código, producto, precio en dólares

americanos, peso en gramos, nombre de la imagen y las existencias del producto. Y se han agregado 10 productos o filas (*register*) para simular una compra.

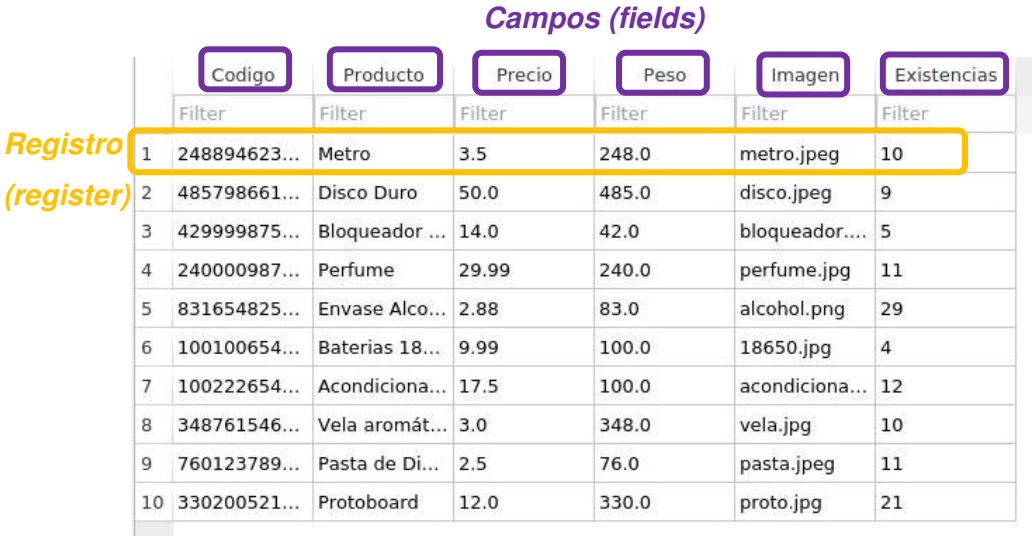


Figura 3.16 Tabla utilizada en el prototipo

Ahora, los códigos que se encuentran en la columna “código” no son los que se encuentran impresos por defecto en el producto, sino que se generaron intencionalmente en una página *web*, resultó el método más conveniente porque si bien algunos productos poseen su código impreso en el empaque, otros no; además que usando esta página *web* se puede controlar cuántos caracteres tendrá el código final.

Generación de códigos de barras a través de *barcode.tec-it.com*

A través de esta *web* es posible generar todo tipo de código, los clásicos códigos de barras, códigos QR y otras opciones más. Para los productos que escaneará la pistola del prototipo se ha optado por usar solo números y cada código debe tener una longitud fija de 12 números; al usar solo números se agiliza ligeramente la búsqueda en la base de datos; también, el uso de solo 12 números por cada código ayuda a realizar una búsqueda más rápida; por lo general, los códigos de barras comerciales 1D varían entre 20 y 25 caracteres [29]. Con estas características nombradas, la Figura 3.17 es un ejemplo de cómo lucen los códigos generados.

Figura 3.17 Ejemplo de código de barras generado a través de *barcode.tec-it.com*

Finalmente, se utilizó *code-128* que es uno de los varios tipos de códigos lineales existentes, es uno de los más usados ya que es compatible con la mayoría de lectores de códigos de barras [30].

Configuración de la pistola lectora de códigos *3nstar sc050*

Las configuraciones por defecto varían de acuerdo a la marca de cada lector; para el modelo que se ha utilizado en el prototipo, el *3nstar sc050,* se descubrió que ya cumplía con ciertas pautas que se necesitaban.

De fábrica, este lector presiona automáticamente *Enter* al finalizar de leer un código, también lee códigos *code-128* por defecto [31]. Cumple con 2 de las 4 pautas requeridas, resta por configurar el modo continuo (*hands-free*) y aplicar un retraso después de cada lectura.

Para habilitar el modo continuo, se debe escanear 3 códigos comandos que se encuentran en la Figura 3.18.

Figura 3.18 De izquierda a derecha, configuración del lector de códigos en modo continuo o *hands-free* [31]

El primer código comando se encarga de ingresar al modo configuración del lector, el segundo código de habilitar el modo lectura continua, y el tercer código de guardar y cerrar el modo configuración.

Finalmente, existen 8 valores a escoger para aplicar un retraso entre la lectura de un código y el siguiente código, entre estos valores se encuentra 2000 (ms) que es el más adecuado para el prototipo [31].

Al igual que en la Figura 3.18, los dos códigos de los extremos de la Figura 3.19 se encargan de abrir y cerrar el modo configuración, el código central es el que permite agregar el retraso de 2 segundos o 2000 (ms).

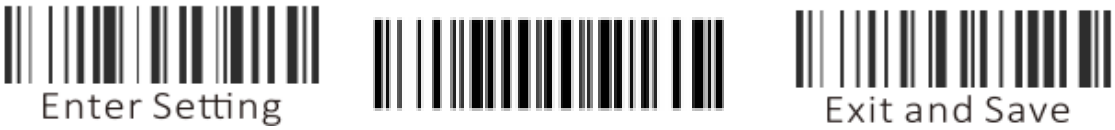


Figura 3.19 De izquierda a derecha, configuración para agregar dos segundos de retraso entre lecturas [31]

3.4 Construcción del prototipo de autoservicio

Circuito y distribución de pines en *Raspberry*

En la Figura 3.21 se muestra el diagrama esquemático con los pines utilizados en la *Raspberry,* los sensores y botones que se han integrado. Para facilitar la numeración de los pines, también se presenta la Tabla 3.6 que señala donde estará conectado cada pin en notación BOARD, la notación BOARD se refleja en la Figura 3.20 en donde se numera a cada pin en forma de zigzag, desde el número 1 hasta el número 40.

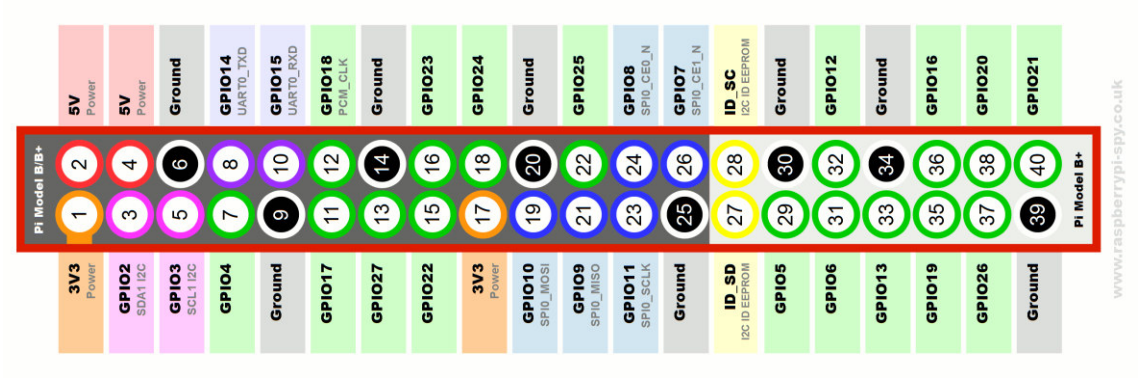


Figura 3.20 Numeración de pines tipo BOARD confinada en el rectángulo rojo [32]

Tabla 3.6 Conexiones realizadas entre el elemento o dispositivo, y el pin *BOARD* de *Raspberry*

Elemento	Pin del elemento	Pin en la *Raspberry* (*BOARD*)
Botón verde	Extremo 1	Pin digital 3
	Extremo 2	GND, pin 14
Botón rojo	Extremo 1	Pin digital 12
	Extremo 2	GND, pin 14
Botón azul	Extremo 1	Pin digital 36
	Extremo 2	GND, pin 14
Botón de apagado	Extremo 1	Pin digital 32
	Extremo 2	GND, pin 30
Ventilador	Vcc	+5 (V), pin 4
	Cable negativo/tierra	GND, pin 6
RFID	SDA	Pin digital 24
	SCK	Pin digital 23
	MOSI	Pin digital 19
	MISO	Pin digital 21
	GND	GND, pin 14
	RST	Pin digital 22
	3.3 (V)	+3.3 (V), pin 17
	IRQ	vacío
HX711	Vcc	+5 (V), pin 2
	DT	Pin digital 29
	CK	Pin digital 31
	Tierra	GND, pin 39

Para realizar la representación esquemática de los circuitos del prototipo se ha utilizado el *software Fritzing* que cuenta con toda clase de dispositivos electrónicos para ser simulados.

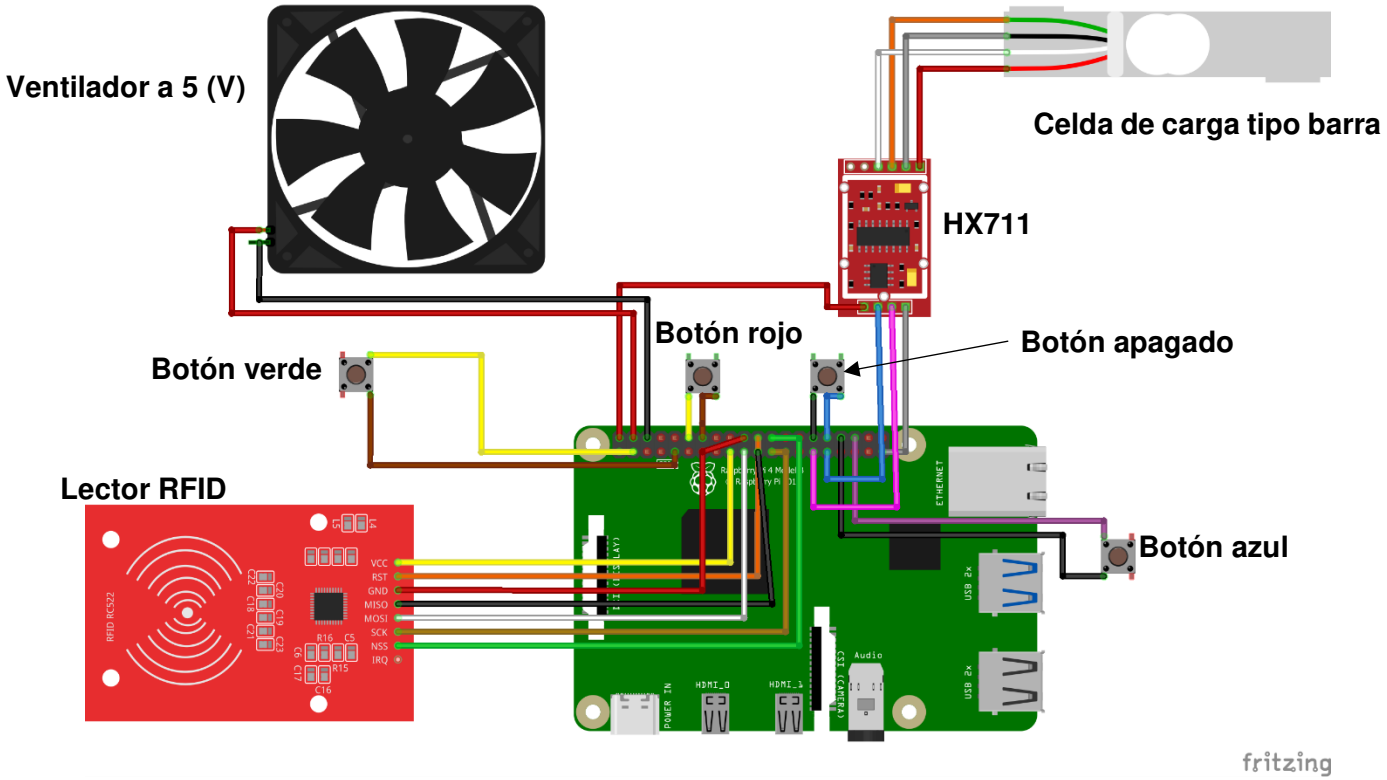


Figura 3.21 Diagrama esquemático del circuito general realizado en *Fritzing*

Construcción de la placa PCB

Una placa PCB permitió reducir la cantidad de cables que se usó, esto se logró unificando los pines de tierra en una sola salida, también se diseñó para intercambiar los dispositivos fácilmente en caso que uno de estos llegue a estropearse.

Para lograr este objetivo, se ha utilizado el programa de simulación *Proteus* en el cual se ha diseñado un circuito, en la Figura 3.22 se representa esquemáticamente las borneras y los pines que conectarán los cables de los botones físicos y el lector RFID.

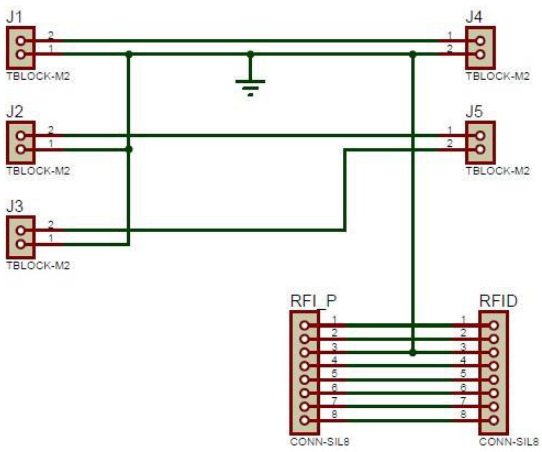


Figura 3.22 Circuito esquemático de la placa PCB a implementar

Con el circuito desarrollado en *Proteus,* este debe ser trasladado a *Ares* que es una herramienta incluida en *Proteus* utilizada para la fabricación de placas de circuito impreso [33]. La Figura 3.23 muestra el resultado obtenido con *Ares;* además de un diseño 3D de cómo lucirá la placa una vez colocados los pines y borneras.

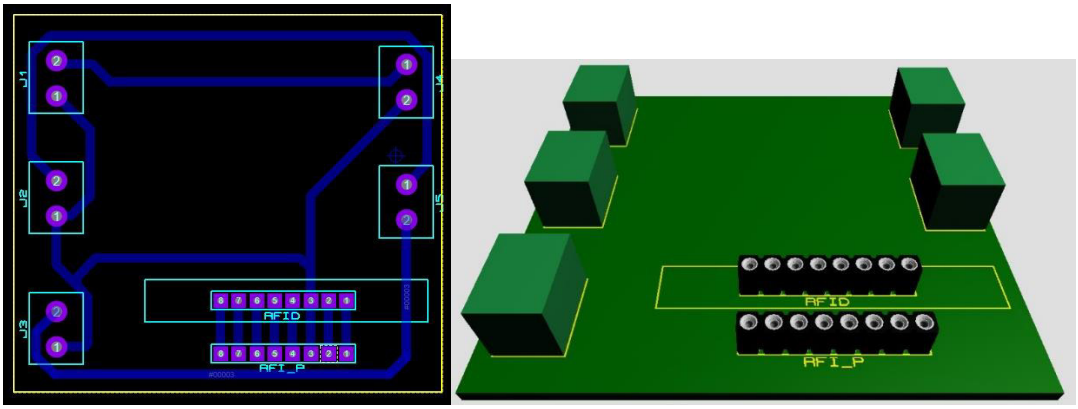


Figura 3.23 Resultado de la placa impresa obtenida en *Ares* y su simulación en 3D

El resultado práctico de la placa PCB con sus puntos soldados se encuentra en la Figura 3.24.

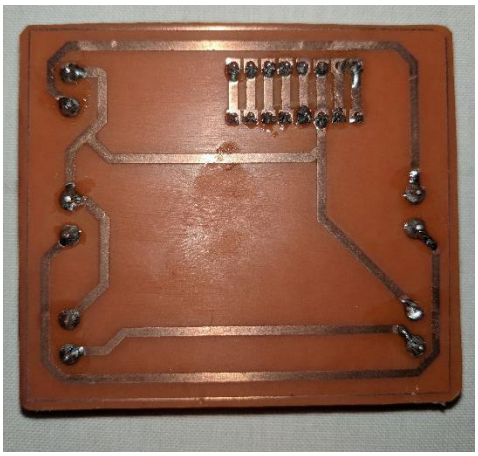


Figura 3.24 Parte posterior de la placa PCB realizada

En la Figura 3.25 se muestra a la placa con sus elementos ya soldados, estas borneras y pines albergarán los cables provenientes de los botones físicos y del lector RFID.

Figura 3.25 Parte frontal de la placa PCB realizada

Esta placa logra reducir la cantidad de cables usados en los 3 botones físicos del cliente ya que une todas las tierras en un solo punto que se une al pin BOARD 14 de la *Raspberry* como se señaló en las conexiones de la Tabla 3.6, entonces, con un extremo del cable a tierra, a cada botón le resta un cable por el otro extremo, este cable restante es guiado a su respectivo pin digital para cumplir con las funcionalidades del código desarrollado.

La Figura 3.26 esquematiza gráficamente como se realizó la tierra en común.

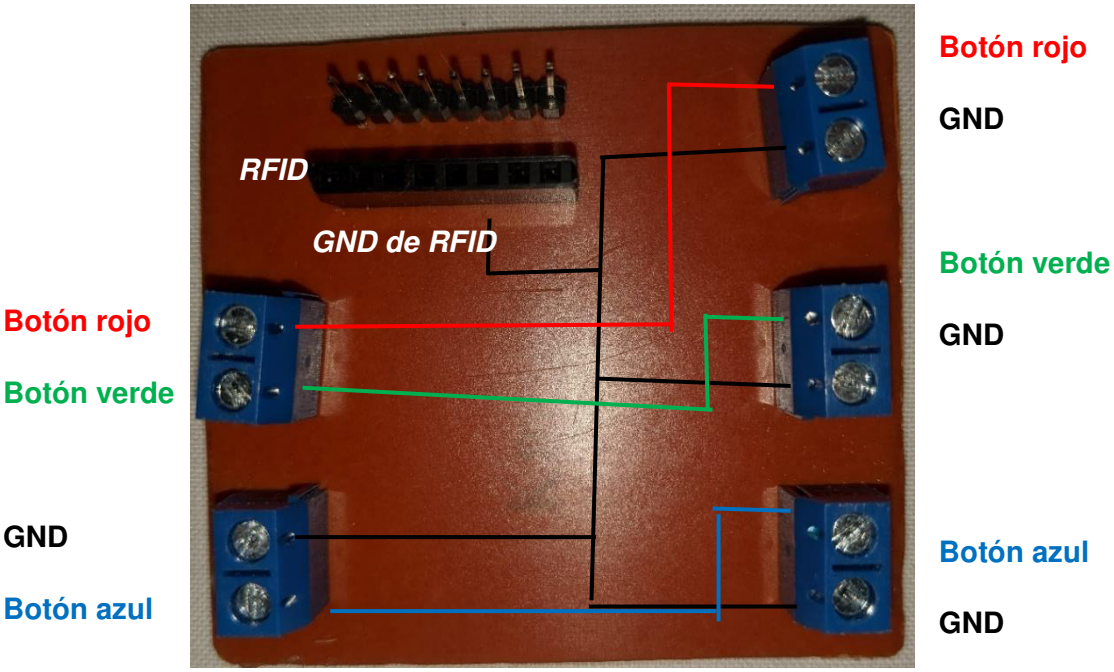


Figura 3.26 Elementos a conectar en las borneras y pines de la placa PCB

Diseño y construcción de estructura

A partir de un bosquejo de la estructura hecho en papel y lápiz, se logró determinar los factores más relevantes de la estructura: sus dimensiones y el material a usar. Además, surgieron varios diseños tentativos de los cuales se tomaron los 2 posibles mejores para el prototipo.

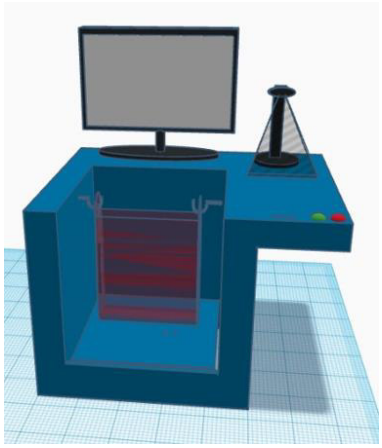
Una vez realizado el bosquejo, se procedió a realizar un diseño en 3D para visualizar de mejor manera las medidas o dimensiones. Para el diseño 3D se empleó el *software web Tinkercad*. Con este programa *web,* se plasmó el bosquejo realizado a lápiz en un proyecto 3D, esto permitió tener una mejor noción de cómo luciría la estructura y realizar los cambios de medidas que eran necesarios.

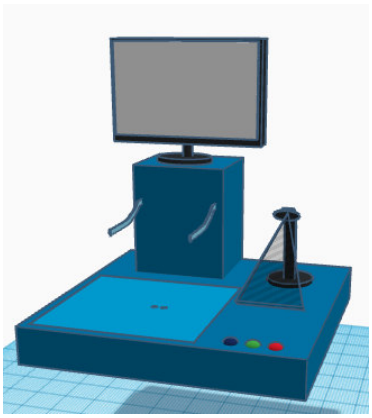
De manera paralela al diseño de la estructura, se identificaron requerimientos que debieron considerarse para cumplir con los requisitos del prototipo, el diseño debe ser de un material capaz de soportar el peso del monitor que visualizará el cliente, el cual pesa aproximadamente 2.5 (kg); debe ser resistente a la deformación ya que la mayoría de cables y dispositivos electrónicos se ocultan en los compartimientos huecos de la estructura; y debe ser de un material que se pueda perforar y cortar de manera casera en caso de que se requiera hacer modificaciones.

Los dos diseños se observan en la Figura 3.27 cumplen con los requerimientos ya mencionados. Sin embargo, el primer diseño resultaba de difícil manipulación para el cliente que usará esta caja de autoservicio; la funda se encuentra colgando entre tres paredes, esto implicaba que el usuario debe introducir su mano y brazo completamente para depositar los productos en la funda, otro punto en contra es la posición de la pistola lectora la cual está demasiado arriba para el cliente. También, la implementación del primer diseño implicaba más material y más costo respecto al segundo diseño.

El segundo diseño es el más adecuado porque es más fácil de acceder a la funda, además que todos los elementos se encuentran a la misma altura; tanto la balanza como la pistola lectora son más fáciles de manipular, y el monitor es el único dispositivo que se ha dado una elevación para que el usuario pueda visualizar de mejor manera, se requiere menos material para fabricar el segundo diseño, consecuentemente, su precio de producción también será más bajo.

Diseño 1





Diseño 2

Figura 3.27 Diseños propuestos para la estructura realizados en *Tinkercad*

Para la selección del material que usará la estructura se realizó un estudio de materiales donde se tomó en cuenta especificaciones como: grosor, proceso de fabricación, costo, rigidez, peso, complejidad para modificar, cortar y perforar sobre el mismo.

Se compararon 3 candidatos a usar en la Tabla 3.7: acero cortado, fibropanel de densidad media (MDF) y filamento plástico a base nailon o policarbonato, siendo este último material el usado para impresión 3D. Con estos materiales se realizó una comparativa que permita elegir a uno de ellos.

Tabla 3.7 Comparativa de materiales para la construcción del armazón

Factor	Acero Cortado	MDF	Impresión 3D
Peso	Altamente Pesado	Ligeramente Pesado	Ligero
Corte de piezas	Láser, herramientas especiales	Láser, herramientas especiales	Ninguno
Costo	Intermedio	Bajo	Alto
Tiempo de fabricación	144 horas	72 horas	16 horas
Tiempo de diseño	48 horas	24 horas	72 horas
Acabados	Soldadura	Bordes quemados debido al corte láser	Compacto al ser integrado todo en una sola pieza
Resistencia o rigidez	Alta	Intermedia	Baja
Calidad	Intermedia	Alta	Intermedia
Grado de personalización sobre el diseño final	Bajo	Alto	Bajo

Finalizada la comparativa, se concluyó que el material más adecuado es MDF porque su costo es accesible; es resistente, lo suficiente para soportar el peso del monitor; es fácil de realizar modificaciones sobre este material, y el tiempo para armar e integrar las piezas es bajo.

Seleccionado el material, se realizó un diseño con cada una de las piezas a cortar en el *software AutoCAD,* esto se indica en la Figura 3.28 que es el diseño final que se usó para formar estas planchas en MDF usando corte láser. Cada plancha posee pestañas que encajan con su pieza adyacente, realizar estas pestañas son muy útiles para brindar mayor rigidez a la estructura.

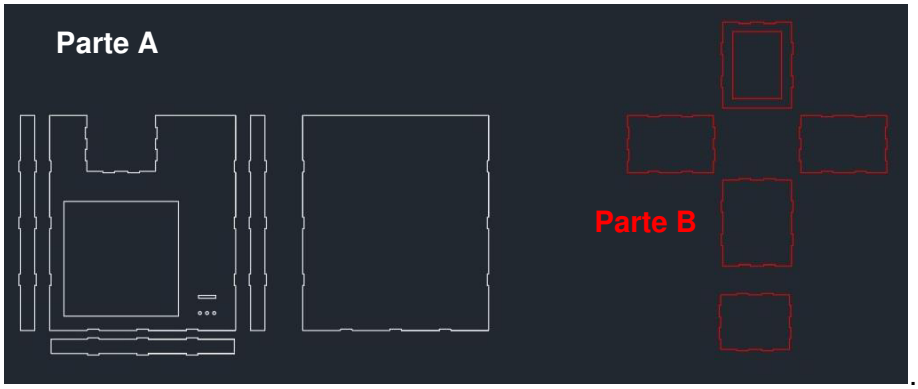


Figura 3.28 Diseños de piezas realizado en *AutoCAD*

En la Figura 3.28 parte A, se encuentra el diseño de la base inferior la cual es totalmente llana y es donde se integran la celda de carga, pulsadores, placa PCB, Circuito RFID y cables. Mientras que en la Figura 3.28 parte B, se encuentra el diseño de la cámara hueca principal que soporta el peso del monitor y a su vez, dentro de esta cámara se alberga la *Raspberry*, el amplificador HX711*,* el pulsador para apagar el prototipo y sobrantes de cables.

En la Figura 3.29 se visualiza la estructura final realizada en MDF, cada pieza ha sido unida con pegamento para madera y el grosor de cada plancha es de 6 (mm).

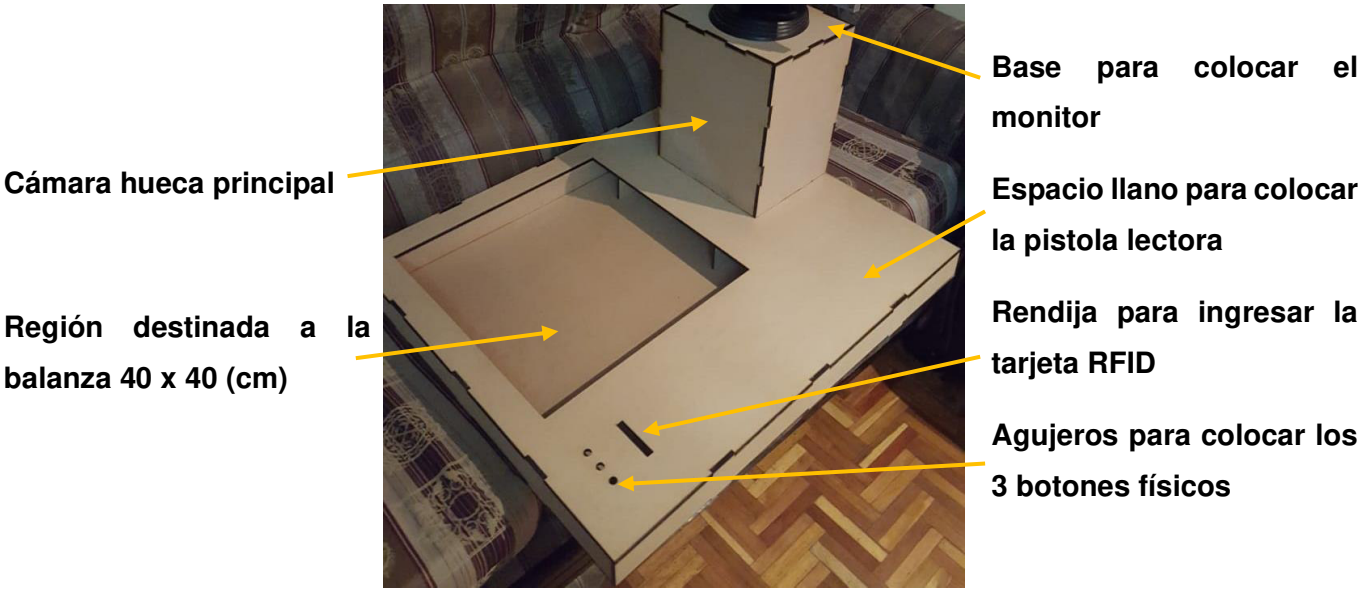


Figura 3.29 Armazón final

La Figura 3.30 es un duplicado de la Figura 3.29 en donde se han colocado las dimensiones finales de la estructura, esto a fin de evitar sobrecargar de datos a la Figura 3.29.

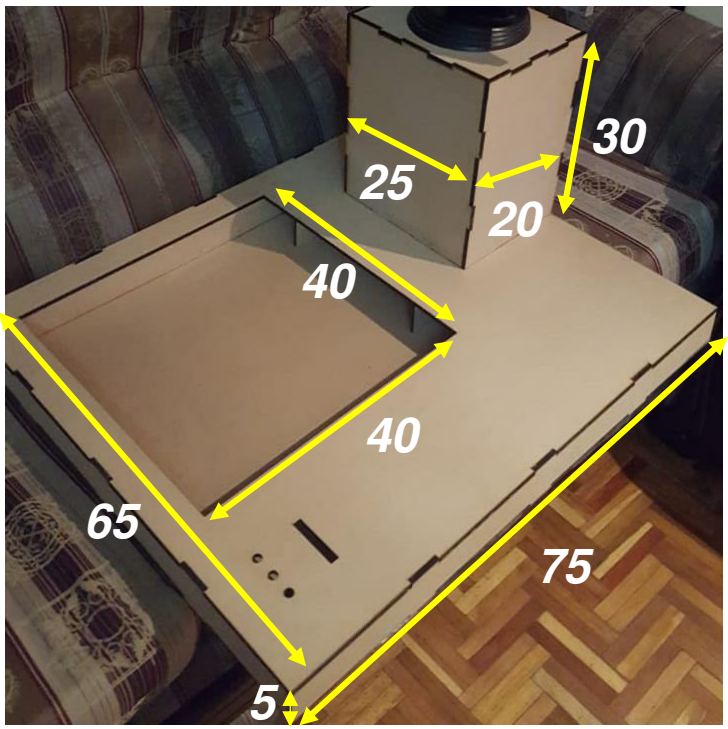


Figura 3.30 Dimensiones de la estructura en (cm)

Integración de cables y dispositivos dentro de la estructura

Los cables y dispositivos han sido colocados debajo de la estructura y en la cámara hueca principal de la Figura 3.29; las conexiones de la placa PCB se aprecian en la Figura 3.31.

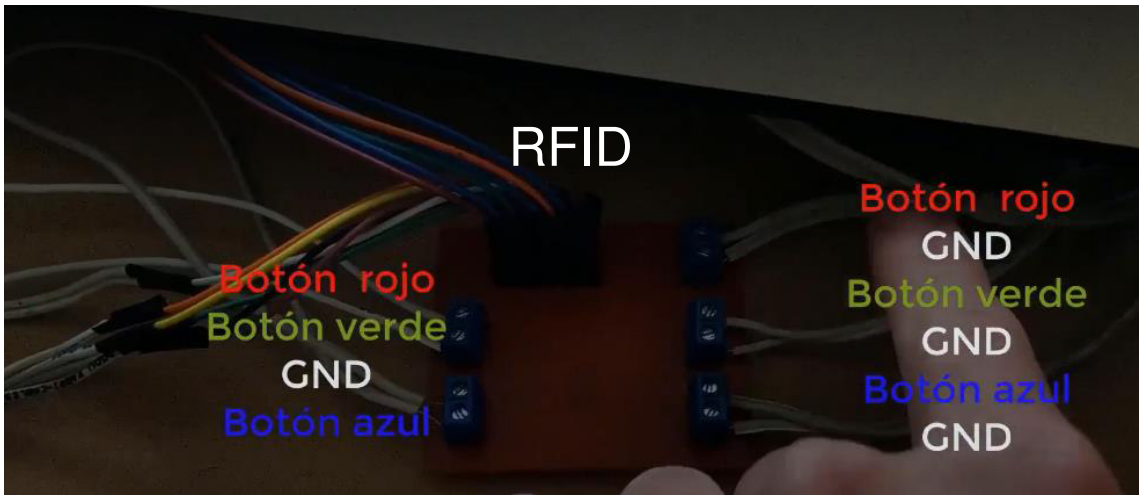


Figura 3.31 Placa PCB y su cableado implementado

Ahora, debido a la cantidad de cables, tanto en la parte frontal como en la parte posterior del prototipo, el cableado se apreciaba antiestético en la Figura 3.32 y Figura 3.33, se observa que se utilizó cinta adhesiva temporalmente.

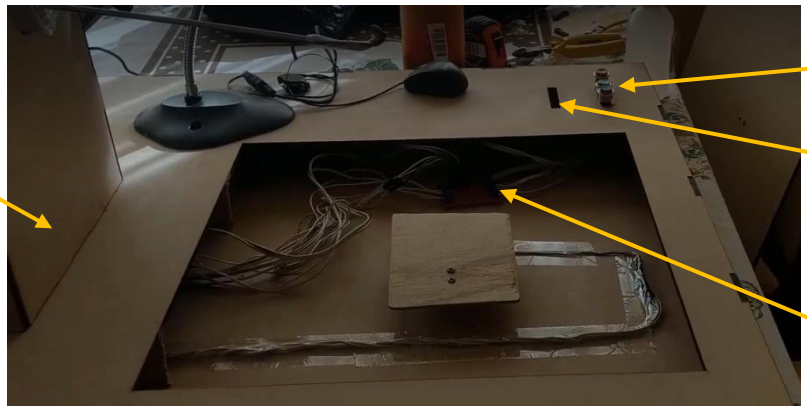


Figura 3.32 Cableado delantero sin canaletas

Figura 3.33 Cableado posterior sin recoger y componentes sin sujeción

Para ordenar el cableado frontal, se ha utilizado canaleta flexible espiral, ideal para la gran cantidad de cables que han generado los botones y el lector RFID, después se han fijado con ganchos metálicos atornillables. Y para el cableado de la celda de carga se ha usado canaleta rígida rectangular. La Figura 3.34 muestra el resultado después de aplicar las diferentes canaletas.

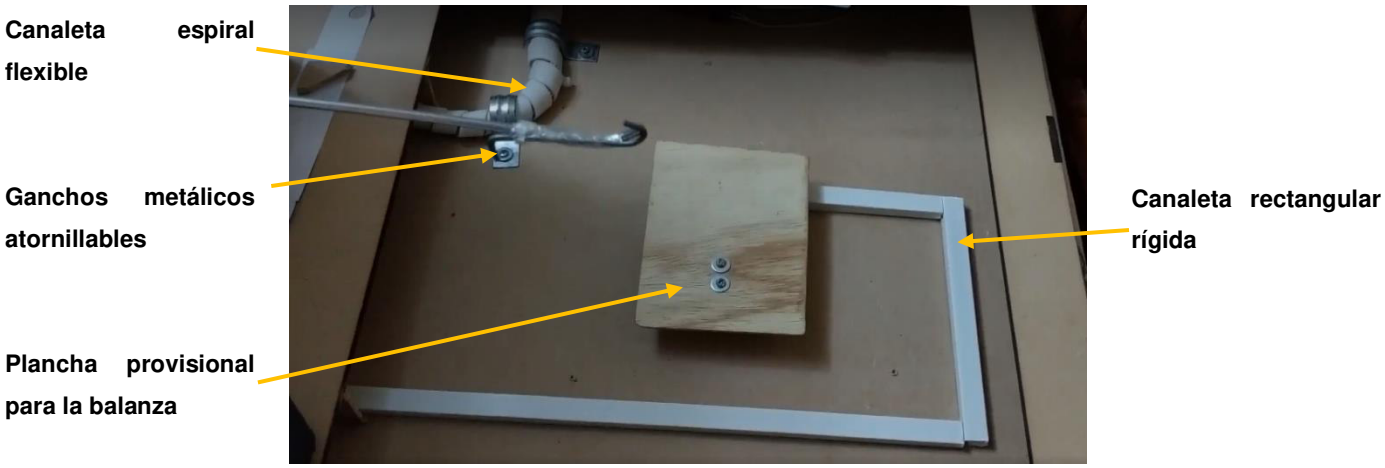


Figura 3.34 Cableado delantero con canaleta espiral y canaleta rectangular

El cableado posterior también fue recogido con amarras plásticas. El pulsador de apagado de la Figura 3.36 ha sido fijado con silicona caliente, al igual que el amplificador HX711. La *Raspberry* ha sido confinada usando ángulos metálicos como se ve en la Figura 3.35.

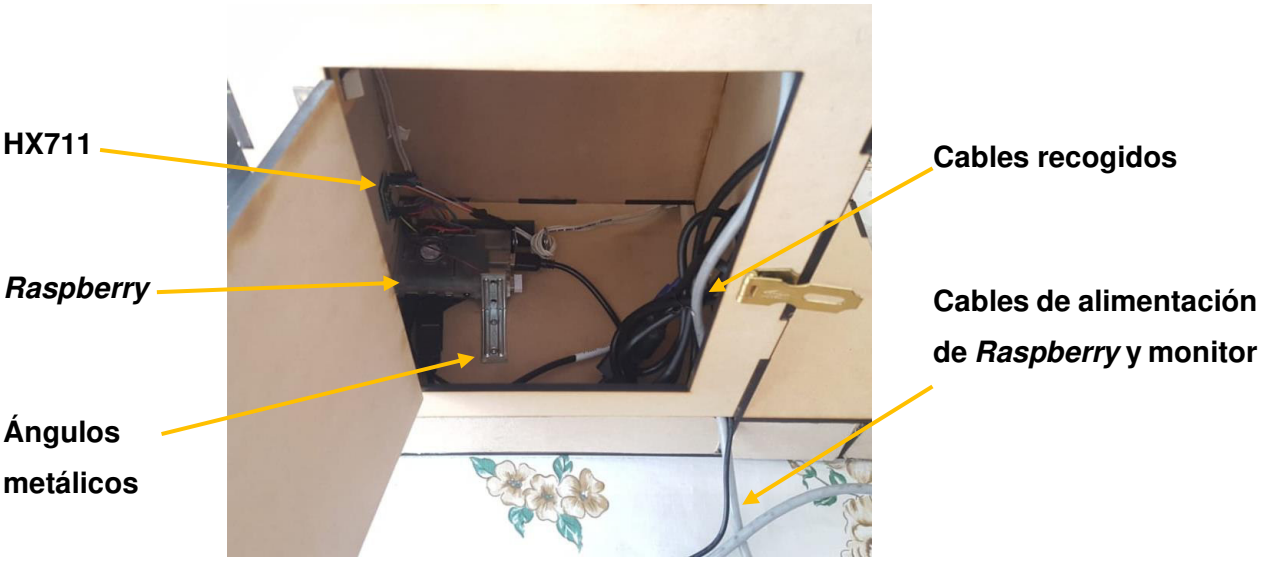


Figura 3.35 Cableado posterior recogido, *Raspberry* y dispositivos restantes fijados

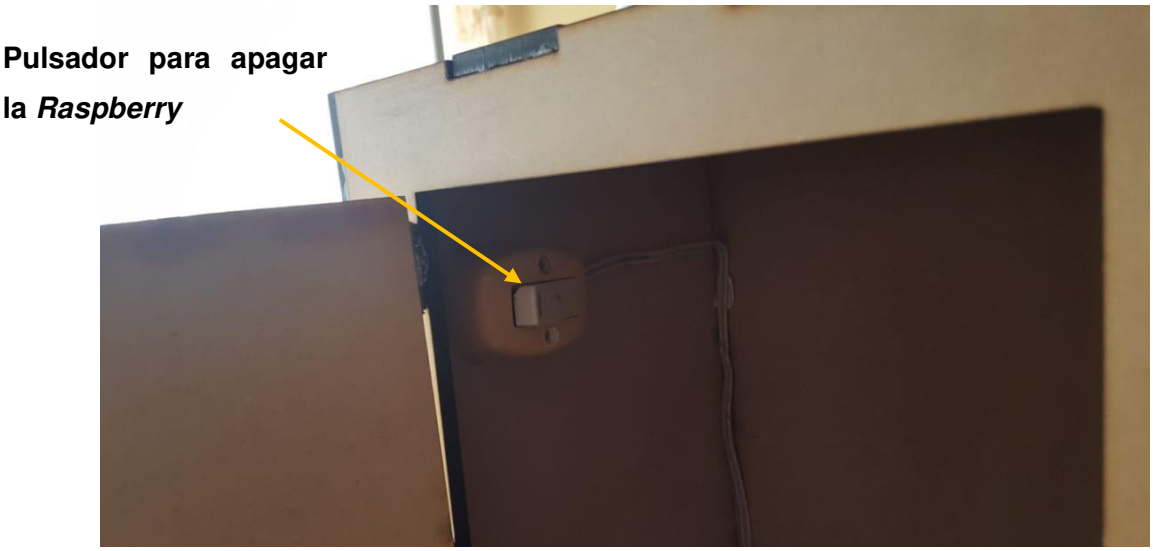


Figura 3.36 Ubicación del pulsador de apagado dentro de la cámara principal

Finalmente, la Figura 3.37 muestra que se ha implementado un candado para que solo el administrador del prototipo pueda acceder a la cámara principal; y la Figura 3.38 la colocación de la plancha final que censará el peso de los productos.

Figura 3.37 Resultado final de la parte posterior del prototipo

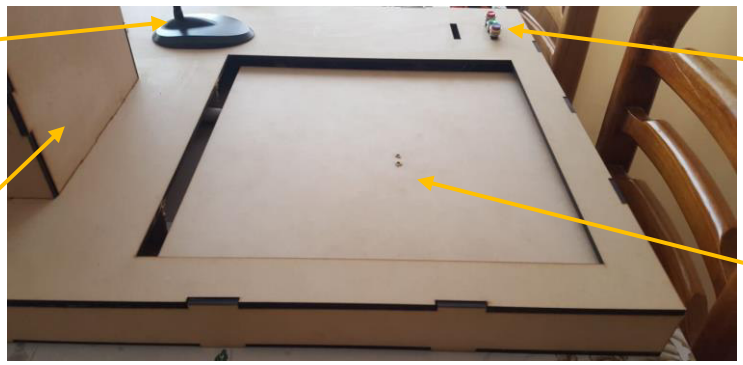


Figura 3.38 Implementación de la plancha definitiva para pesar los artículos

Costo del proyecto

El costo del proyecto de la Tabla 3.8 contempla los materiales adquiridos y la mano de obra para integrarlos. Los materiales principales se adquirieron por Mercado Libre [34]; y la mano de obra abarca la programación en *Python*, armado de la estructura, soldaduras, organización de cables y la integración de los dispositivos con el armazón.

Tabla 3.8 Costo de implementación

Cantidad	Detalle	Precio unitario	Precio total
100	Mano de obra, aproximadamente 100 horas	$10.00	$1 000.00
1	*Raspberry Pi 4* versión 4 (GB) RAM	$70.00	$70.00
1	Estructura realizada en MDF	$67.00	$67.00
1	Pistola lectora de códigos de barras 1D	$40.00	$40.00
1	Monitor 17”	$40.00	$40.00
1	Cargador original *Raspberry Pi 4*	$15.00	$15.00
1	Memoria micro SD 32 (GB) clase 10	$15.00	$15.00
1	Carcasa con ventilador *Raspberry Pi 4*	$10.00	$10.00
1	Cable de imagen HDMI a VGA	$7.00	$7.00
1	Baquelita	$7.00	$7.00
1	Amplificador y conversor analógico – digital HX711	$5.50	$5.50
1	Celda de carga tipo barra de 10 (kg)	$5.00	$5.00
1	Lector RFID RC522	$3.00	$3.00
2	Ganchos demostrativos o ganchos para perchas	$1.20	$2.40
10	10 metros de cable de timbre	$0.20	$2.00
40	*Jumpers* variados, tanto machos como hembras	$0.05	$2.00
3	Pulsadores redondos de colores	$0.55	$1.65
1	Pulsador grande de timbre	$1.00	$1.00
4	Pernos con arandelas para celda de carga	$0.25	$1.00
3	Ángulos metálicos pequeños	$0.30	$0.90
COSTO TOTAL			**$1 295.45**

3.5 Realización de pruebas de funcionamiento del prototipo

Etapa de arranque del prototipo

En primera instancia, el prototipo no cuenta con teclado o ratón para navegar por su interfaz gráfica, por lo que es necesario un arranque automático del programa *Python*, ya se explicó cómo se consigue el arranque del programa en el diagrama de la Figura

3.12. Y cómo luce el arranque en la práctica se aprecia en la captura de pantalla de la Figura 3.39.

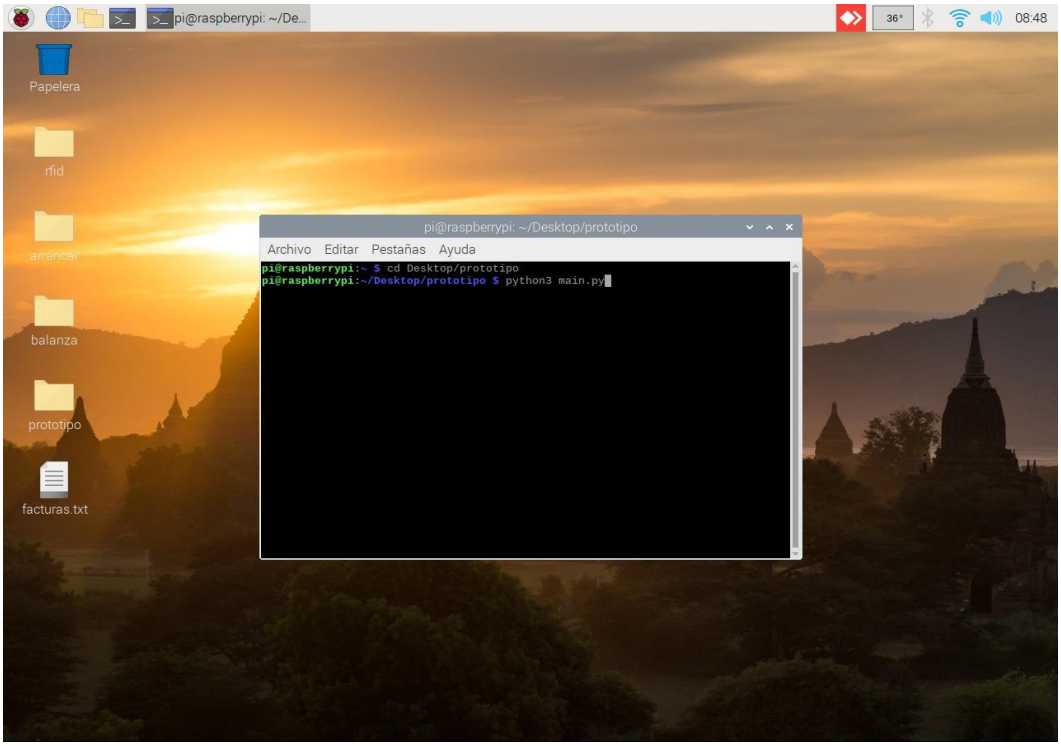


Figura 3.39 Arranque automático del programa *main.py*

Tal como se mencionó en el diagrama de la Figura 3.12, una vez se haya energizado la placa *Raspberry*, el puntero se posicionará sobre el ícono de la terminal, la abrirá, escribirá y ejecutará los dos comandos que se ven en la Figura 3.39.

A través del comando *cd Desktop/prototipo* se accede al directorio prototipo donde se encuentra el ejecutable *.py*, y el segundo comando *python3 main.py* ejecuta el archivo *main.py* a través de *Python* en su versión 3.

Etapa de calibración del prototipo

La calibración del prototipo debe ser realizada cada vez que se encienda el prototipo, para ello, se ha utilizado un flexómetro o cinta métrica como peso patrón.

Para convertir a este flexómetro en el peso patrón, se necesitó conocer el peso del mismo, para este propósito se usó una balanza externa, el peso que arrojó se muestra en la Figura 3.40; este valor fue ingresado en el código del programa principal para realizar la calibración correctamente.

Figura 3.40 Peso en gramos del objeto patrón para calibrar la balanza

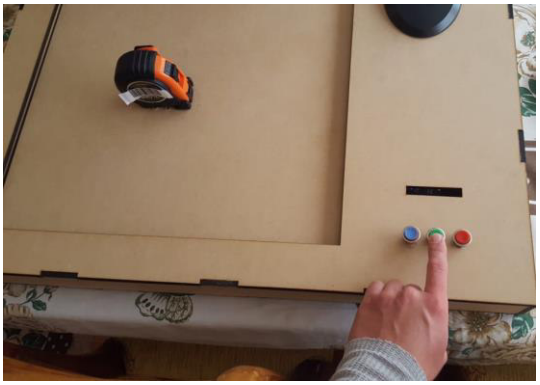


Figura 3.41 Etapa de calibración

La primera ventana que revela el programa al iniciar es la indicada en la Figura 3.42, esta ventana es la que dará la instrucción de colocar el peso patrón sobre la balanza y presionar el botón físico tal como lo muestra la Figura 3.41.

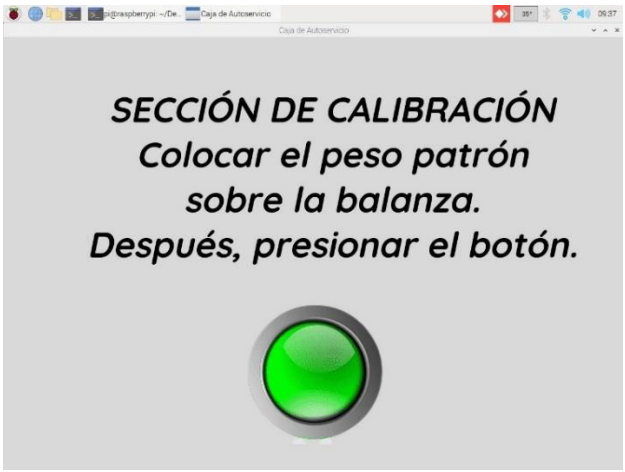


Figura 3.42 Primera ventana de calibración

Después, la segunda ventana de la Figura 3.43 indicará que se ha calibrado correctamente y que el prototipo ya puede ser usado por los clientes.

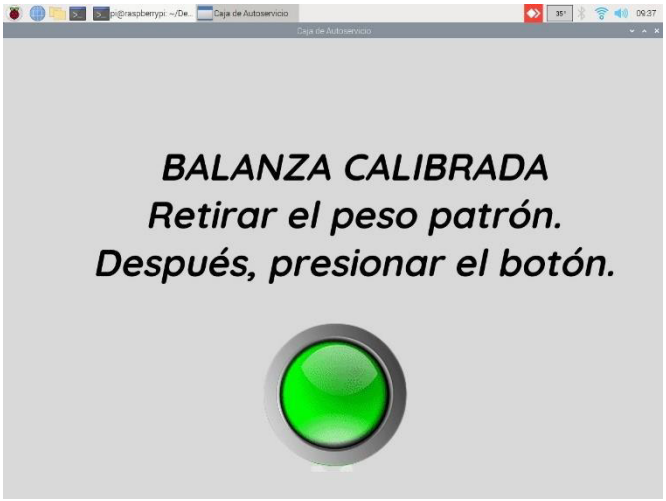


Figura 3.43 Ventana final de calibración

Etapa de uso por parte del cliente

En esta etapa existen varias ventanas con funciones que cambian de acuerdo a la acción que ejecute el cliente. Un principio primordial del *script* es activar solo los botones y periféricos que se muestran en pantalla; por ejemplo, la Figura 3.44 indica que se debe presionar el botón físico verde para proceder a escanear; consecuentemente, los 2 botones restantes, azul y rojo, no cumplen ninguna función dentro de esta ventana; por ende, deben ser bloqueados por el programa de tal manera que si el cliente presiona cualquier botón que no es el verde, no se ejecutará ninguna acción. Del mismo modo, si el cliente usa la pistola lectora o la balanza, estas no desencadenarán ninguna acción porque deben estar intencionalmente bloqueadas durante esta ventana.

Esta lógica debe ser aplicada en cada ventana, en donde se activarán solo los dispositivos que se muestran en pantalla, y los restantes serán bloqueados.

La Figura 3.44 es la ventana de espera e invitación al cliente, esta permanecerá de manera ininterrumpida hasta que el usuario presione el botón verde. En ella se despliega un aviso del límite de peso que soporta la balanza, este límite ya fue explicado y desarrollado en la selección de balanza.

Figura 3.44 Ventana de invitación a usar el prototipo de autoservicio

La Figura 3.45 es la ventana que se despliega cuando se presiona el botón físico verde, en ella se muestra una tabla vacía en donde se mostrarán los productos escaneados por el cliente; además de un botón de retorno en caso de no continuar con la compra, este botón llama de regreso a la ventana de la Figura 3.44.

Figura 3.45 Ventana para empezar a escanear los productos

Después de escanear un producto, de manera inmediata, se despliega la ventana de verificación de peso de la Figura 3.46 en donde se indica que se coloque el producto sobre la balanza.

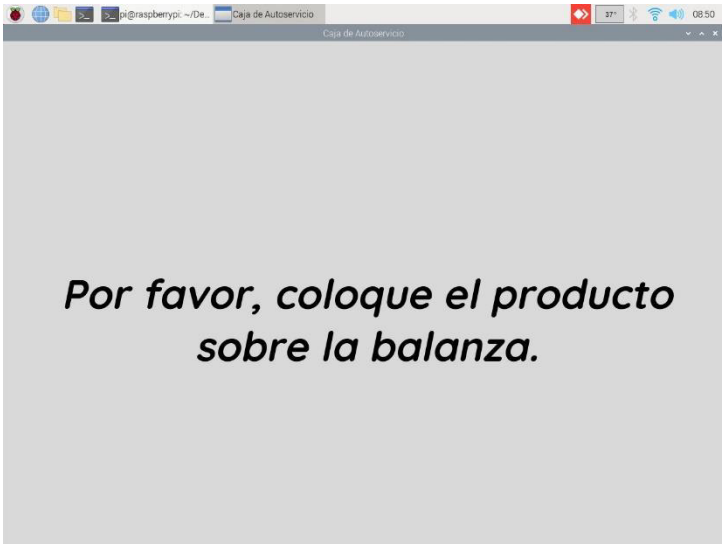


Figura 3.46 Validación de peso al agregar peso

Cuando se coloca el producto correcto sobre la balanza, se cierra la ventana de la Figura 3.46 para dar paso a la ventana de la Figura 3.47, en donde se muestra el producto escaneado con su nombre, precio e imagen.

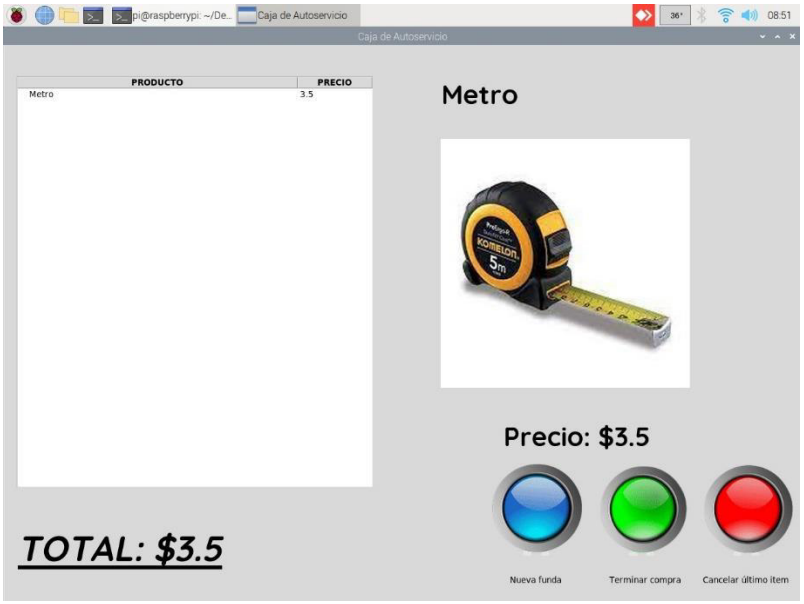


Figura 3.47 Ventana de escaneo con nuevas funcionalidades

También, esta nueva ventana muestra 3 botones que permiten: cancelar el último producto escaneado, terminar la compra o seleccionar una nueva funda.

Las ventanas de la Figura 3.45, Figura 3.46 y Figura 3.47 se repetirán tantas veces como productos tengan que ser escaneados y verificados. De esta manera, la tabla y el valor total a pagar se volverá más grande como se muestra en la Figura 3.48.

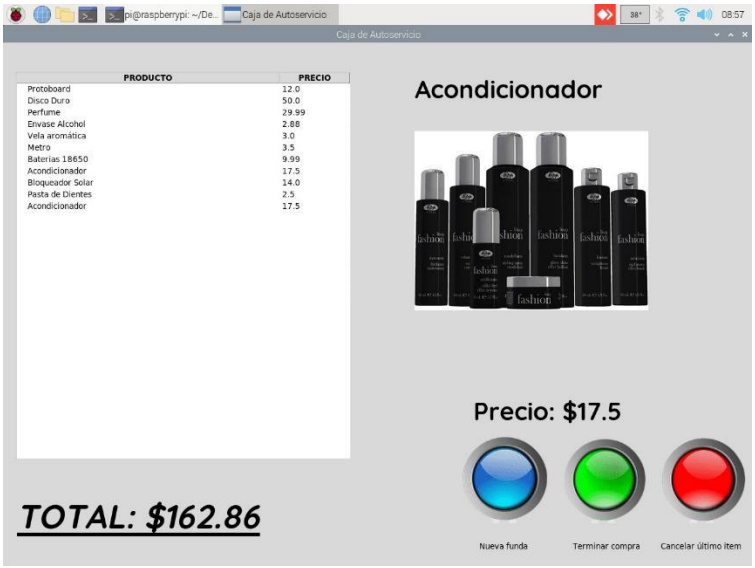


Figura 3.48 Ventana con varios productos escaneados

Ahora, existen 3 opciones en la Figura 3.48, la opción “Cancelar último ítem” permite retirar el último artículo escaneado; al presionarlo, se despliega la ventana emergente de la Figura 3.49; una vez se haya retirado el producto correcto, se retorna a la ventana de la Figura 3.48, pero con el último producto eliminado de la tabla y del total a pagar.

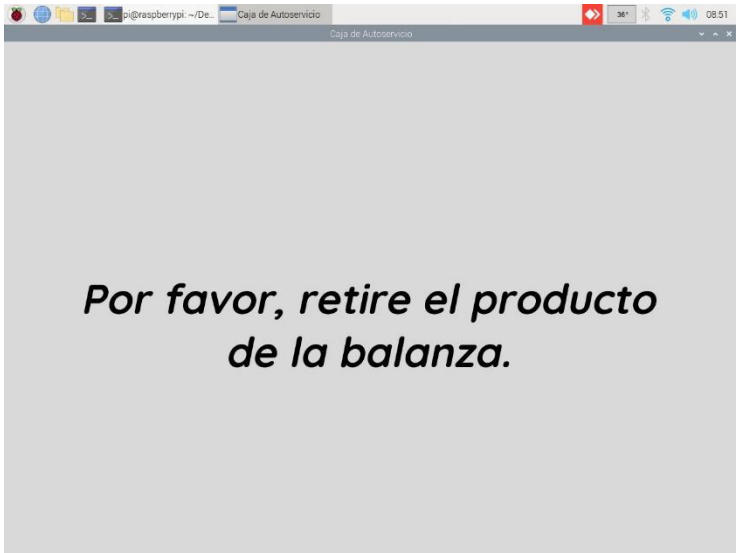


Figura 3.49 Validación de peso al retirar peso

La opción “Nueva funda” capacita al cliente a cambiar su funda actual con productos por una vacía, la opción debe ser usada si la funda actual se encuentra llena de productos, si se encuentra cerca de superar el límite de carga de la balanza o si el cliente considera necesario realizar un cambio de funda.

Al seleccionar la opción de “Nueva funda”, se despliega la ventana emergente de la Figura 3.50 y advertirá al usuario que, si cambia de funda, ya no podrá retirar los artículos de las anteriores fundas. Si se presiona el botón de la opción “Si”, la balanza es encerada y asume que se tomó una nueva funda. Si se presiona el botón de la opción “No”, se asume el cliente continuará depositando productos en su funda actual.

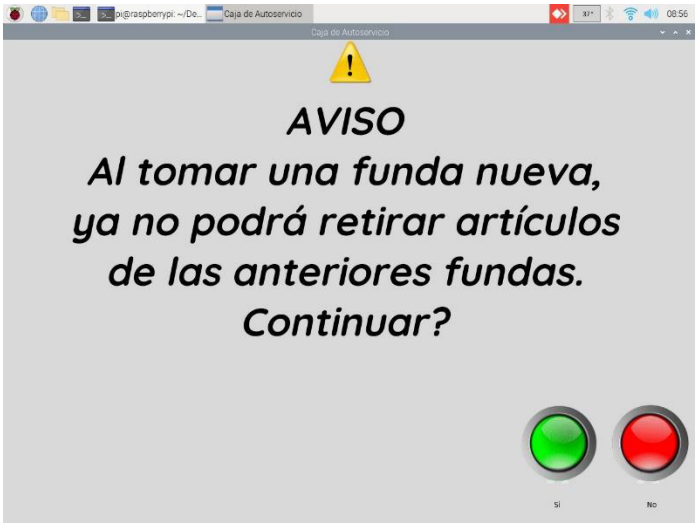


Figura 3.50 Ventana de advertencia al solicitar una nueva funda

De todas maneras, si ya se optó por una nueva funda y el cliente, de manera persistente, desea retirar productos de anteriores fundas, se mostrará la ventana de la Figura 3.51 en donde se indica que no es posible deshacer los productos de fundas anteriores. Si se presiona “Continuar”, se asume que el usuario continuará la compra y se retornará a la Figura 3.48, caso contrario, al presionar “Cancelar todo” se cancelará toda la compra y se retornará a la ventana inicial de la Figura 3.44.

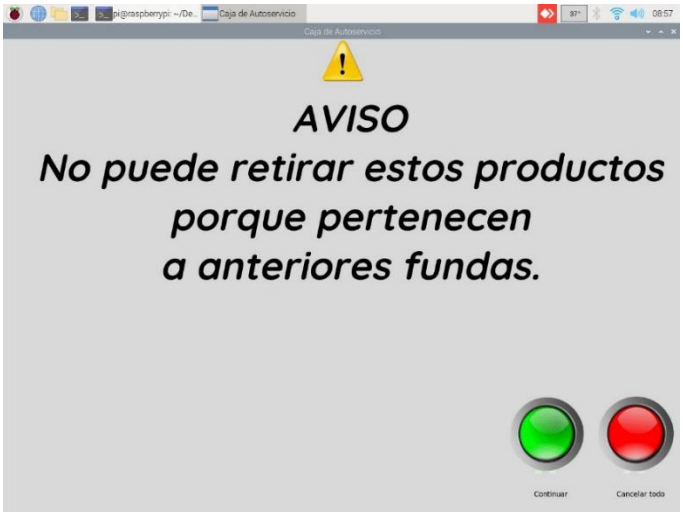


Figura 3.51 Ventana de advertencia al intentar retirar artículos de fundas anteriores

Finalmente, la opción “Terminar compra” indica que se ha concluido la etapa de escaneo de productos y se procederá a realizar la simulación de pago que se muestra en la Figura 3.52. Pero antes de esto, se realiza una verificación de peso final, en donde los pesos de la última funda son sumados y deben ser validados para pasar a la ventana de pago, caso contrario, se mostrará el mensaje emergente de la Figura 3.53.

Figura 3.52 Ventana para realizar simulación de pago

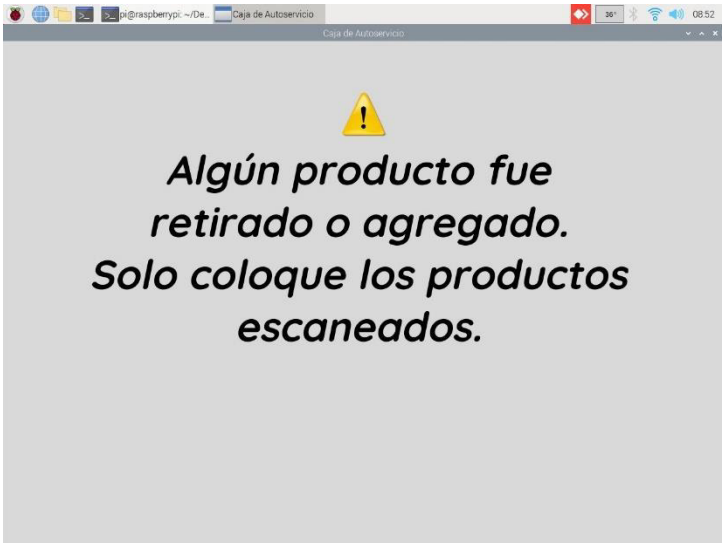


Figura 3.53 Ventana de validación final antes de proceder al pago

Ahora, existen dos eventos posibles al realizar la simulación de pago: que sea exitosa y se concrete la compra, o que el saldo sea insuficiente.

En el caso de un pago exitoso, se muestra la ventana emergente de la Figura 3.54 por un breve momento, para después retornar a la primera ventana del programa de la

Figura 3.44. Y de manera paralela, el programa abre un archivo con extensión *.txt* en donde se guarda un detalle de los artículos comprados con su fecha y hora, este archivo se encuentra en el escritorio de la *Raspberry* y se visualiza en la Figura 3.55.

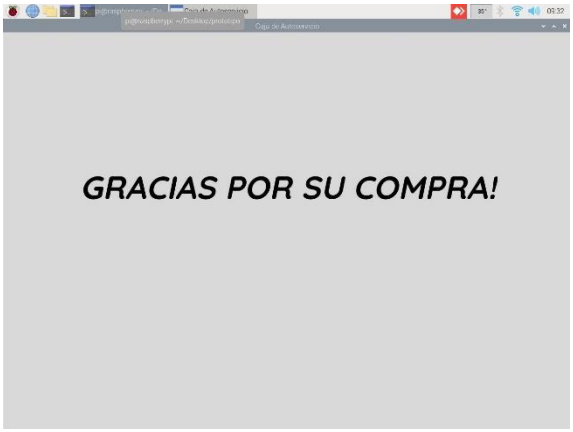


Figura 3.54 Ventana emergente de compra finalizada exitosamente

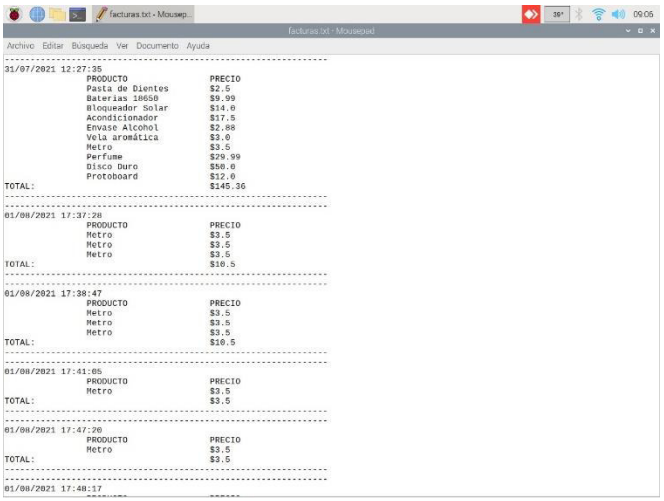


Figura 3.55 Archivo de texto que registra las compras realizadas

Caso contrario, un pago no exitoso lanza una ventana de saldo insuficiente como se muestra en la Figura 3.56, después de unos segundos, se visualiza la nueva ventana de pago de la Figura 3.57, la cual añade un nuevo botón respecto a la primera ventana de pago de la Figura 3.52, en donde se ofrece la opción de “Quitar productos” y “Cancelar todo”.

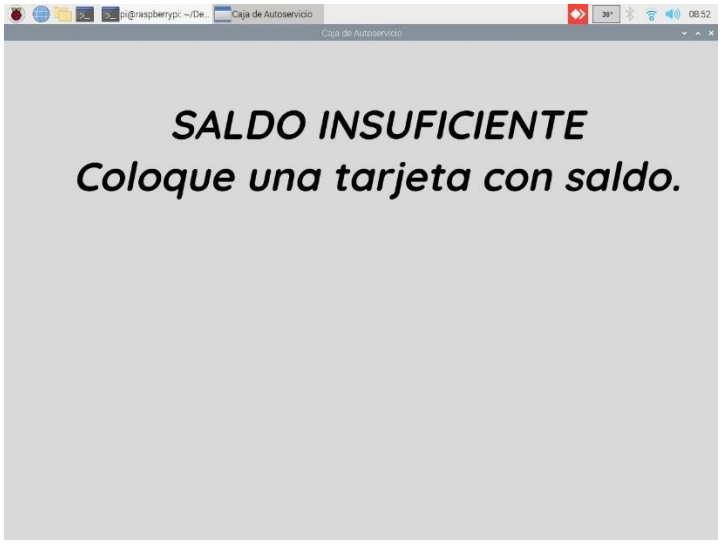


Figura 3.56 Ventana de pago no exitoso

Figura 3.57 Nueva ventana de pago

La opción “Quitar productos” permite retornar a la ventana de la Figura 3.48, y “Cancelar todo” anula la compra en su totalidad y retorna a la pantalla inicial de la Figura 3.44.

Impresión de códigos de barras

Como se mencionó, los códigos de barras se desarrollaron en la página *web https://barcode.tec-it.com/es.* La misma página permite descargar los códigos de barras en formato *png*.

Ahora, en la Figura 3.58 se encontró que los códigos cortados de esta manera tienden a generar conflictos con la pistola lectora tales como lecturas erróneas o retrasos en las lecturas.

Figura 3.58 Código cortado sin sobrantes a los extremos laterales

Más tarde, se descubrió que estos errores de lectura y retraso en leer el código por parte de la pistola lectora se producían debido a poco espacio que existen en los laterales; es decir, dejar espacios a los extremos izquierdo y derecho del código permite realizar una lectura correcta. El código de la Figura 3.59 respecto al código de la Figura 3.58 fue menos propenso a generar conflictos de lectura.

Figura 3.59 Código cortado con sobrantes a los extremos laterales

3.6 Manual de uso y mantenimiento

Para una mejor explicación, se han grabado y editado 2 videos a los que se puede acceder a través de códigos QR; la Figura 3.60 redirige al manual de uso y la Figura 3.61 redirige al manual de mantenimiento.

Figura 3.60 Video de manual de uso

Figura 3.61 Video de manual de mantenimiento

4 CONCLUSIONES Y RECOMENDACIONES

4.1 Conclusiones

- Se concluye que el prototipo de autoservicio implementado como parte de este proyecto es un concepto que permite a los usuarios emplear de mejor manera su tiempo ya que es un sistema rápido, siempre y cuando la cantidad de productos a adquirir sea pequeña, se considera entre 1 y 15 artículos la cantidad adecuada ya que escanear cantidades más grandes a estas puede resultar tedioso para el cliente repercutiendo en una mala experiencia. También se debe acotar que una de las restricciones del prototipo es el peso máximo soportado por la celda de carga que es de 10 (kg), entonces se dedujo que entre menos artículos tengan que ser adquiridos por el cliente, menor es la probabilidad que la capacidad máxima de la celda de carga sea superada.
- En la selección de los dispositivos electrónicos tales como sensores y placas de desarrollo se infiere que no es suficiente analizar los sensores que requiere el prototipo, si no que los mismos sean compatibles con la placa de desarrollo escogida; y más importante aún, se debe investigar previamente que existan las librerías para incorporar estos sensores en la *Raspberry* y que estas librerías hayan sido escritas en un lenguaje de programación que soporte la placa de desarrollo. De manera resumida, la compatibilidad de dispositivos debe ser total para evitar inconvenientes al integrarlos, tanto a nivel de *hardware* como de *software*.
- Se infiere que el *software* más adecuado para esquematizar las conexiones entre la placa *Raspberry* y los sensores fue *Fritzing*. Un potencial candidato a usar fue *Tinkercad* el cual genera circuitos muy llamativos pero su librería es muy limitada, en *Tinkercad* no fue posible encontrar muchos de los sensores que se usaron en el prototipo. Por otro lado, *Proteus* cuenta con muchas librerías, pero en contraste, la manera cómo presenta los circuitos no es atractiva. *Fritzing* fue la opción más equilibrada porque se encontraron todos los sensores y dispositivos electrónicos y la forma cómo los presenta recuerda mucho a *Tinkercad.*
- A nivel de *software* y programación, se concluye que *Python* es un lenguaje de programación sencillo de aprender y que cuenta con una comunidad enorme detrás del mismo. Además, *Python* facilitó la integración de los sensores del prototipo ya que cuenta con un prolijo catálogo de librerías y módulos

almacenados en repositorios como *GitHub*. Sin embargo, existen tantas librerías que es posible encontrar algunas obsoletas o no compatibles con los sensores o con la propia *Raspberry.* Programar en *Python* ha sido fácil; pero la búsqueda de las librerías para los sensores ha sido un ensayo recursivo de prueba y error.

- Se concluye que existen varias maneras de arrancar automáticamente un programa o un comando en *Raspberry Pi OS;* pero algunos programas debido a sus restricciones no permiten ser arrancados. Tal caso se presentó al intentar arrancar el *script* principal y se descubrió que el problema se debía a la librería *tkinter* de *Python*. Para saldar esta problemática, se creó un *script* auxiliar el cual sí permitía ser ejecutado al iniciar la *Raspberry*, este pequeño código ejecuta las sentencias necesarias para arrancar al programa principal.
- Se concluye que el material MDF usado en la estructura fue el más adecuado ya que tiene buena rigidez, es estable, soporta pesos grandes y se requieren altos esfuerzos para deformarla; definitivamente cumplió con las características planteadas en el objetivo. No obstante, se descubrió una debilidad atribuida al material, el mismo es susceptible a sufrir cambios al contacto con el agua o en lugares con altos niveles de humedad, a tal punto que puede tornarse endeble.
- Se infiere la importancia de utilizar memorias de almacenamiento de alto rendimiento. Para desarrollar esta conclusión, se debe decir que la placa *Raspberry* usa memorias micro SD como elemento principal de almacenamiento del sistema operativo y de los datos del usuario. En un principio, se utilizó una memoria de clase 4; las clases se asignan a las memorias para referirse a su velocidad de escritura y de lectura; a mayor sea la clase, mayores velocidades. Se optó por clase 4 ya que son muy asequibles, fáciles de conseguir y con velocidades de 4 (MB/s) en lectura y escritura lo cual era suficiente para la placa *Raspberry* en un uso básico. Sin embargo, a medida que se desarrolló la interfaz gráfica y se implementó la base de datos, el prototipo exigía cada vez más recursos de *hardware*. Es cuando se identificó que esta memoria de clase 4 estaba generando un cuello de botella, el sistema operativo se tornó lento y entorpecía la experiencia de uso. Se logró solventar este problema colocando una memoria de clase 10 con velocidades de 10 (MB/s) en escritura y lectura la cual se desempeña mejor con la *Raspberry Pi 4*.
- En las pruebas de funcionamiento se infiere que la etapa más crítica del prototipo es la calibración; si la balanza no se calibra correctamente, el sistema no funcionará porque leerá pesos erróneos. Se ultimó que mientras la celda de carga toma muestras para calibrar, la balanza debe permanecer totalmente

inmóvil y el sistema en general debe encontrarse en una superficie plana; de esta manera se garantiza que la balanza arrojará lecturas correctas.

- Se concluye que este prototipo está formado por dos componentes: uno de *hardware* y otro de *software;* teniendo más relevancia el conocimiento en el área de *software*, esto se pudo evidenciar en la investigación a fondo que se realizó acerca de la placa *Raspberry*, el estudio del paradigma de programación orientada a objetos, el estudio del lenguaje de programación *Python,* el uso de base de datos locales *SQLite,* la investigación de librerías y módulos para *Python* a través del repositorio *Github*. Además, la realización de este prototipo validó que la investigación y la autoeducación fueron los pilares para solventar inconvenientes, tanto a nivel de *software* como de *hardware*, y conseguir la implementación satisfactoria del proyecto.

4.2 Recomendaciones

- En el presente prototipo se usó una celda de carga tipo barra, la cual funcionó correctamente, pero el área de contacto de la plancha superior era muy grande, de 40 (cm) por 40 (cm), por lo que la placa superior presentó oscilaciones y esto puede comprometer las lecturas de la balanza. Por ello, se recomienda colocar cuatro celdas de carga de tipo cuadradas en cada esquina en lugar de colocar solo una en el medio de la plancha, de esta manera se pueden minimizar las oscilaciones.
- Para evitar errores de lectura por parte de la pistola lectora se recomienda cortar los códigos de barra impresos dejando espacios en blanco a los laterales (derecha e izquierda). La Figura 3.58 ejemplifica la manera incorrecta de cortar un código de barras; en contraste, la Figura 3.59 la manera correcta.
- Implementar un sistema de pago real utilizando una *Application Programming Interface* (API) ofertada por compañías que ofrecen servicios de pagos electrónicos. Si se aplican estas APIs al *script* desarrollado, será posible realizar pagos reales a través de códigos QR y aplicaciones móviles. Y para complementar esta característica, se puede agregar una impresora de impacto o una impresora térmica, también conocidas como impresoras de punto de pago o facturadoras *Point Of Sale* (POS) para generar un recibo como constancia y evidencia de los productos que se han adquirido.
- Reemplazar el monitor por otro con capacidades *touch* o táctiles. Si bien el contexto pandémico actual insta a evitar tocar superficies que son manipuladas por cientos de personas, también es verdad que el uso de pantallas táctiles está

muy masificado en comparación a usar botones físicos. Además, se pueden agregar más funcionalidades y más botones en pantalla en contraste a los tres botones físicos que se usaron en este prototipo.

- Para mejorar la experiencia de uso se recomienda utilizar librerías que permitan enviar correos más llamativos a la vista. El presente prototipo usó una de las librerías por defecto de *Python* para enviar correos electrónicos, estos correos son muy sencillos y solo pueden manejar texto. Se recomienda emplear una librería que permita enviar correos con imágenes, plantillas y gráficos.
- Reemplazar la celda de carga de acuerdo a los requerimientos, el presente prototipo consideró adecuado una celda con carga máxima de 10 (kg), pero puede ser intercambiada por una de mayor o de menor capacidad, siempre recordando que una capacidad mayor conlleva una precisión menor.
- Reemplazar el lector de códigos de barras tipo pistola por un lector de códigos tipo omnidireccional. El lector tipo pistola requiere que el código a escanear se encuentre alineado con el haz de láser de la pistola; al usar un lector omnidireccional el usuario es libre de colocar el código rotado, de lado o no alineado con el lector, con este cambio se puede mejorar la experiencia de uso del cliente.
- Evitar usar memorias de clases bajas para las placas *Raspberry*, apuntar a memorias de clase 7 o mayores para obtener el mejor rendimiento posible de esta placa de desarrollo. Adicionalmente, evitar adquirir memorias clonadas cuyos rendimientos son dudosos.
- Impermeabilizar la estructura en caso de que el prototipo sea usado en zonas con alta humedad; ya se ha descrito que el agua y humedad comprometen la integridad de la estructura hecha en MDF, por ello se recomienda utilizar algún método como la aplicación de barniz para proteger el material.
- Cambiar la base de datos utilizada. Si se tuvieran varias cajas de autoservicio en un mismo supermercado, se esperaría que las bases de datos de todas ellas se mantengan actualizadas; eso no es posible con *SQLite* ya que es una base de datos de uso local, se recomienda usar *MySQL* o *MariaDB* para manejar un modelo cliente – servidor y que las cajas de autoservicio puedan conectarse a una base de datos común alojada en un servidor.

5 REFERENCIAS BIBLIOGRÁFICAS

[1] Raspberry.org, «Raspberry Pi 4,» [En línea]. Available: https://www.raspberrypi.org/products/raspberry-pi-4-model-b/.

[2] Raspberry.org, «GPIO Interface,» de *Peripherals*, 2019, pp. 9-10.

[3] Raspberry.org, «Usage,» [En línea]. Available: https://www.raspberrypi.org/documentation/usage/.

[4] Arduino.cc, «Arduino Mega 2560 REV3,» [En línea]. Available: https://store.arduino.cc/usa/mega-2560-r3.

[5] BeJob, «Qué es la programación con Arduino y para qué sirve,» 14 Febrero 2017. [En línea]. Available: https://www.bejob.com/que-es-la-programacion-con-arduino-y-para-que-sirve/#:~:text=Lenguaje%20de%20la%20programaci%C3%B3n%20con%20Arduino%3A%20C%2B%2B&text=La%20plataforma%20Arduino%20se%20programa,es%20similar%20a%20C%2B%2B..

[6] P. TH, «Begginers Guide,» 18 9 2019. [En línea]. Available: https://wiki.python.org/moin/BeginnersGuide/Overview.

[7] I. Pérez, Y. Díaz y R. Becerra, «El lenguaje de programación Python,» junio 2014. [En línea]. Available: https://www.redalyc.org/pdf/1815/181531232001.pdf.

[8] ComputerWeekly, «Write once, run anywhere?,» 2 Mayo 2002. [En línea]. Available: https://www.computerweekly.com/feature/Write-once-run-anywhere.

[9] Oracle, «Introduction to Java TM Technology,» [En línea]. Available: https://www.oracle.com/java/technologies/introduction-to-java.html.

[10] CodersLegacy, «Python GUI with Tkinter,» [En línea]. Available: https://coderslegacy.com/python/python-gui/.

[11] P. TH, «Tkinter,» 17 2 2021. [En línea]. Available: https://wiki.python.org/moin/TkInter.

[12] Debian.org, «¿Qué es Debian GNU/Linux?,» [En línea]. Available: https://www.debian.org/releases/jessie/mips/ch01s03.html.es.

[13] Raspberry.org, «Raspberry Pi OS,» [En línea]. Available: https://www.raspberrypi.org/documentation/raspbian/.

[14] SQLite.org, «What Is SQLite?,» [En línea]. Available: https://www.sqlite.org/index.html.

[15] SQLite.org, «About SQLite,» [En línea]. Available: https://www.sqlite.org/about.html.

[16] MySQL, «What Is MySQL?,» [En línea]. Available: https://dev.mysql.com/doc/refman/8.0/en/what-is-mysql.html.

[17] Zebra, «Escanér de mano LS2208,» 2015. [En línea]. Available: https://www.zebra.com/la/es/products/spec-sheets/scanners/general-purpose-scanners/ls2208.html.

[18] Zebra, «LS228: Product Reference Guide,» Junio 2017. [En línea]. Available: https://www.zebra.com/content/dam/zebra_new_ia/en-us/manuals/barcode-scanners/ls2208-product-reference-guide-en-us.pdf.

[19] SparkFun, «Getting Started with Load Cells,» 22 Julio 2016. [En línea]. Available: http://tet.pub.ro/pages/altele/Documentatie/Magnetormetru%20MAG3110/Getting%20Started%20with%20Load%20Cells%20-%20learn.sparkfun.pdf.

[20] SparkFun, «SparkFun Load Cell Amplifier - HX711,» [En línea]. Available: https://www.sparkfun.com/products/13879.

[21] A. Semiconductor, «24-Bit Analog-to-Digital Converter (ADC) for Weigh Scales,» [En línea]. Available: https://cdn.sparkfun.com/datasheets/Sensors/ForceFlex/hx711_english.pdf.

[22] A. Saunders, «El impacto de la tecnología en el crecimiento y el empleo,» [En línea]. Available: https://www.bbvaopenmind.com/articulos/el-impacto-de-la-tecnologia-en-el-crecimiento-y-el-empleo/.

[23] S. López Salas, «Atención al cliente, consumidor y usuario,» [En línea]. Available: https://www.mheducation.es/bcv/guide/capitulo/8448175840.pdf.

[24] C. García, N. Sánchez y H. Pérez, «Aceptación de las Cajas Amigas en el hipermercado Alcampo en la provincia de Tenerife,» Julio 2018. [En línea]. Available: https://riull.ull.es/xmlui/bitstream/handle/915/11455/Aceptacion%20de%20las%20Cajas%20Amigas%20en%20el%20hipermercado%20Alcampo%20en%20la%20provincia%20de%20Santa%20Cruz%20de%20Tenerife.pdf?sequence=1.

[25] E. S, «SQLite vs MySQL – What's the Difference,» 27 Marzo 2019. [En línea]. Available: https://www.hostinger.com/tutorials/sqlite-vs-mysql-whats-the-difference/.

[26] Raspberry.org, «Raspberry Pi 4 Tech Specs,» [En línea]. Available: https://www.raspberrypi.org/products/raspberry-pi-4-model-b/specifications/.

[27] SeedStudio, «Which Raspberry Pi Programming Language should you use in 2021? Comparison Guide,» 2020. [En línea]. Available: https://www.seeedstudio.com/blog/2020/02/25/which-raspberry-pi-programming-language-should-you-use-comparison-guide/.

[28] DigitalOcean, «DB Browser for SQLite,» [En línea]. Available: https://sqlitebrowser.org/.

[29] BarcodesINC, «Barcoding Frequently Asked Questions,» [En línea]. Available: https://www.barcodesinc.com/faq/#:~:text=Depending%20on%20the%20specific%20barcode,go%20up%20to%202%2C000%20characters..

[30] TopBarcodes, «Most Popular 1D and 2D Barcodes,» [En línea]. Available: https://topbarcodes.com/.

[31] 3nstar, «Soporte y Descargables: Escánner Manual de código de barras 1D (SC050),» [En línea]. Available: https://3nstar.com/producto/1d-handheld-barcode-scanner-sc050/?lang=es.

[32] M. Macharla, «Difference between BCM and BOARD pin numbering in Raspberry Pi,» 31 Marzo 2020. [En línea]. Available: https://iot4beginners.com/difference-between-bcm-and-board-pin-numbering-in-raspberry-pi/.

[33] Enerxia, «ELECTRONICA: PROTEUS (ARES e ISIS) simulador digital y analógico,» [En línea]. Available:

https://www.enerxia.net/portal/index.php?option=com_content&view=article&id=406:electronica-proteus-simulador-digital-y-analogico&catid=61&Itemid=142.

[34] Mercado Libre, «Mercado Libre,» [En línea]. Available: https://listado.mercadolibre.com.ec/_CustId_114810563.

[35] SparkFun, «HX711 Datasheet,» [En línea]. Available: https://cdn.sparkfun.com/datasheets/Sensors/ForceFlex/hx711_english.pdf.

[36] SparkFun, «Load Cell Datasheet,» 2015. [En línea]. Available: https://cdn.sparkfun.com/datasheets/Sensors/ForceFlex/TAL220M4M5Update.pdf.

[37] NXP, «MFRC522 Datasheet,» [En línea]. Available: https://www.nxp.com/docs/en/data-sheet/MFRC522.pdf.

ANEXOS

Anexo 1: certificado de funcionamiento

ESCUELA POLITECNICA NACIONAL
Campus Politécnico "J. Rubén Orellana R

Quito, 15 de septiembre de 2021

CERTIFICADO DE FUNCIONAMIENTO DE PROYECTO DE TITULACIÓN

Yo, *Leandro Antonio Pazmiño Ortiz,* docente ocasional a tiempo completo de la Escuela Politécnica Nacional y como director de este trabajo de titulación, certifico que he constatado el correcto funcionamiento de este, ya que cumple de manera satisfactoria con todos los objetivos planteados. Este proyecto fue desarrollado por los estudiantes Johan Burbano y Juniel Incerri.

DIRECTOR
Ing. Leandro Antonio Pazmiño Ortiz., Msc.

Ladrón de Guevara E11-253, Escuela de Formación de Tecnólogos. EXT: 2743
email: leandro.pazmino@epn.edu.ec Quito-Ecuador

Anexo 2: *Datasheet* HX711 [35]

海芯科技 AVIA Semiconductor

HX711

24-Bit Analog-to-Digital Converter (ADC) for Weigh Scales

DESCRIPTION

Based on Avia Semiconductor's patented technology, HX711 is a precision 24-bit analog-to-digital converter (ADC) designed for weigh scales and industrial control applications to interface directly with a bridge sensor.

The input multiplexer selects either Channel A or B differential input to the low-noise programmable gain amplifier (PGA). Channel A can be programmed with a gain of 128 or 64, corresponding to a full-scale differential input voltage of ±20mV or ±40mV respectively, when a 5V supply is connected to AVDD analog power supply pin. Channel B has a fixed gain of 32. On-chip power supply regulator eliminates the need for an external supply regulator to provide analog power for the ADC and the sensor. Clock input is flexible. It can be from an external clock source, a crystal, or the on-chip oscillator that does not require any external component. On-chip power-on-reset circuitry simplifies digital interface initialization.

There is no programming needed for the internal registers. All controls to the HX711 are through the pins.

FEATURES

- **Two selectable differential input channels**
- **On-chip active low noise PGA with selectable gain of 32, 64 and 128**
- **On-chip power supply regulator for load-cell and ADC analog power supply**
- **On-chip oscillator requiring no external component with optional external crystal**
- **On-chip power-on-reset**
- **Simple digital control and serial interface: pin-driven controls, no programming needed**
- **Selectable 10SPS or 80SPS output data rate**
- **Simultaneous 50 and 60Hz supply rejection**
- **Current consumption including on-chip analog power supply regulator: normal operation < 1.5mA, power down < 1uA**
- **Operation supply voltage range: 2.6 ~ 5.5V**
- **Operation temperature range: -40 ~ +85℃**
- **16 pin SOP-16 package**

APPLICATIONS

- **Weigh Scales**
- **Industrial Process Control**

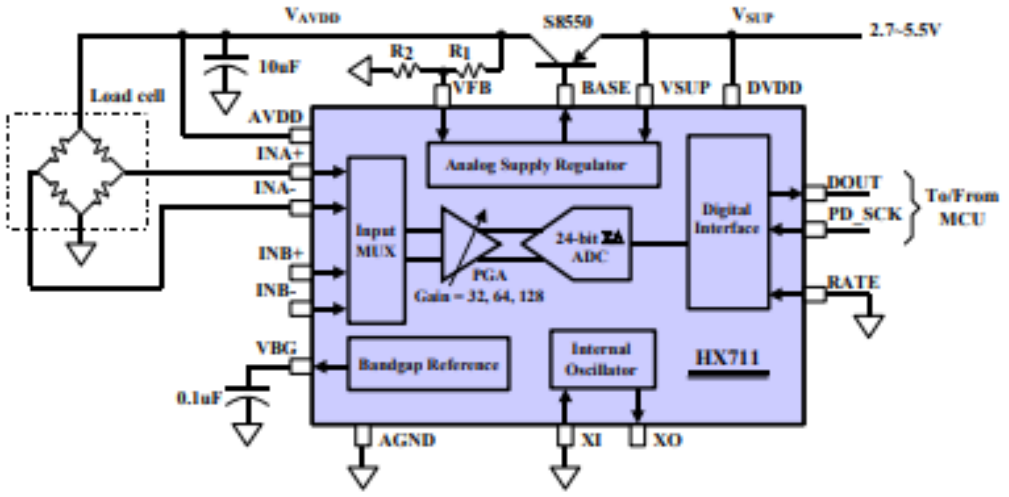
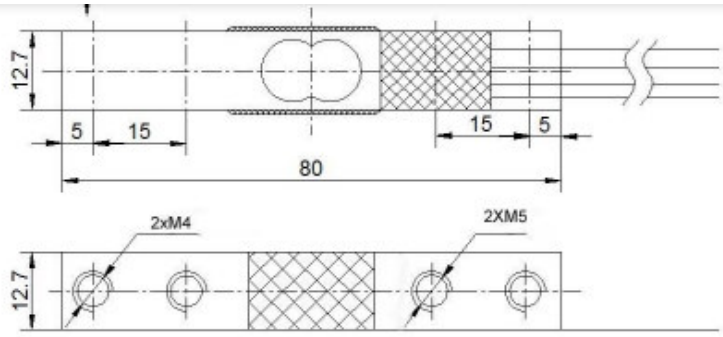
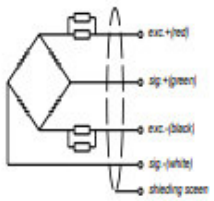


Fig. 1 Typical weigh scale application block diagram

Anexo 3: *Datasheet* celda de carga [36]





Specifications:		
capacity	kg	3,5,10,20,25,30,50(aluminum); 80,100,120,200(alloy steel)
safe overload	%FS	120
ultimate overload	%FS	150
rated output	mV/V	1.0 ± 0.15
excitation voltage	Vdc	5 – 10
combined error	%FS	± 0.05
zero unbalance	%FS	± 0.1
non-linearity	%FS	± 0.05
hysteresis	%FS	± 0.05
repeatability	%FS	± 0.03
creep	%FS/3min	± 0.05
input resistance	Ω	1000 ± 15
output resistance	Ω	1000 ± 10
insulation resistance	M Ω	≥ 2000
operating temperature range	℃	-10 – +55
compensated temperature range	℃	-10 – +40
temperature coefficient of SPAN	%FS/10℃	± 0.05
temperature coefficient of ZERO	%FS/10℃	± 0.05
Electrical connection	cable	4 color wire (standard)or 4 shielded PVC cable, Ø0.8 × 220 mm

Anexo 4: *Datasheet* RFID RC522 [37]

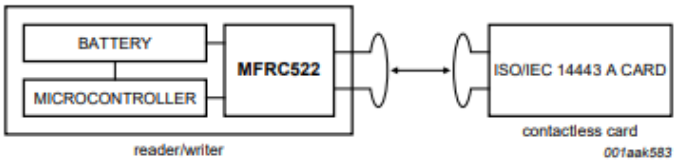
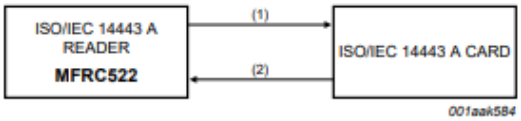


Fig 4. MFRC522 Read/Write mode

The physical level communication is shown in Figure 5.



(1) Reader to card 100 % ASK, Miller encoded, transfer speed 106 kBd to 848 kBd.

(2) Card to reader subcarrier load modulation, Manchester encoded or BPSK, transfer speed 106 kBd to 848 kBd.

Fig 5. ISO/IEC 14443 A/MIFARE Read/Write mode communication diagram

The physical parameters are described in Table 4.

Table 4. Communication overview for ISO/IEC 14443 A/MIFARE reader/writer

Communication direction	Signal type	Transfer speed			
		106 kBd	212 kBd	424 kBd	848 kBd
Reader to card (send data from the MFRC522 to a card)	reader side modulation	100 % ASK	100 % ASK	100 % ASK	100 % ASK
	bit encoding	modified Miller encoding	modified Miller encoding	modified Miller encoding	modified Miller encoding
	bit length	128 (13.56 µs)	64 (13.56 µs)	32 (13.56 µs)	16 (13.56 µs)
Card to reader (MFRC522 receives data from a card)	card side modulation	subcarrier load modulation	subcarrier load modulation	subcarrier load modulation	subcarrier load modulation
	subcarrier frequency	13.56 MHz / 16	13.56 MHz / 16	13.56 MHz / 16	13.56 MHz / 16
	bit encoding	Manchester encoding	BPSK	BPSK	BPSK