Desarrollo de prototipo clasificador y lector de códigos QR basado en Python / MySQL / Zbar

Tesis para optar al grado “Licenciado en Ciencias de la Ingeniería” y título de “Ingeniero en Automatización y Robótica”.

Autor: Rodolfo Antonio Ogino Anriquez
Profesor Guía: Néstor Palominos González

Santiago de Chile, Julio del 2018.
Yo, Rodolfo Antonio Ogino Anríquez, declaro que este documento no incorpora material de otros autores sin identificar debidamente la fuente.

Santiago, julio de 2018.
Dedicatoria

A mis padres quienes hubiesen estado felices de verme capitalizar este esfuerzo, a mis hijos, para quienes quiero ser un buen ejemplo.

Agradecimientos

Gracias a quienes me han apoyado en toda esta marcha por el desierto. A los profesores que con gran voluntad aclararon mis dudas, a mis compañeros que marcharon junto a mí.
<table>
<thead>
<tr>
<th>Contenido</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Resumen</td>
<td>9</td>
</tr>
<tr>
<td>Abstract</td>
<td>10</td>
</tr>
<tr>
<td>1 Antecedentes generales</td>
<td>11</td>
</tr>
<tr>
<td>1.1 Introducción</td>
<td>11</td>
</tr>
<tr>
<td>1.2 Objetivos</td>
<td>13</td>
</tr>
<tr>
<td>1.2.1 Objetivo general</td>
<td>13</td>
</tr>
<tr>
<td>1.2.2 Objetivos específicos</td>
<td>13</td>
</tr>
<tr>
<td>1.3 Carta Gantt</td>
<td>13</td>
</tr>
<tr>
<td>1.4 Organización y presentación del trabajo</td>
<td>14</td>
</tr>
<tr>
<td>1.5 Estado del Arte</td>
<td>14</td>
</tr>
<tr>
<td>2. Marco Teórico</td>
<td>15</td>
</tr>
<tr>
<td>2.1 Códigos QR</td>
<td>15</td>
</tr>
<tr>
<td>2.2 Tipos de códigos QR</td>
<td>17</td>
</tr>
<tr>
<td>2.3 Codificación y decodificación QR</td>
<td>19</td>
</tr>
<tr>
<td>2.3.1 Codificación de la información</td>
<td>20</td>
</tr>
<tr>
<td>2.3.2 Códigos de corrección de errores</td>
<td>22</td>
</tr>
<tr>
<td>2.3.3 División Polinómica</td>
<td>23</td>
</tr>
<tr>
<td>2.3.4 Patrones de búsqueda</td>
<td>23</td>
</tr>
<tr>
<td>2.3.5 Separadores</td>
<td>24</td>
</tr>
<tr>
<td>2.3.6 Patrón de alineamiento</td>
<td>24</td>
</tr>
<tr>
<td>2.3.7 Patrones de temporizado</td>
<td>25</td>
</tr>
<tr>
<td>2.3.8 Áreas reservadas</td>
<td>25</td>
</tr>
<tr>
<td>2.3.9 Ubicación de la información en el código</td>
<td>26</td>
</tr>
<tr>
<td>2.3.10 Enmascarado de Información</td>
<td>28</td>
</tr>
<tr>
<td>2.3.11 Reglas para seleccionar la Máscara</td>
<td>29</td>
</tr>
<tr>
<td>3 Desarrollo</td>
<td>31</td>
</tr>
<tr>
<td>3.1 Costos y presupuestos</td>
<td>31</td>
</tr>
<tr>
<td>3.2 Hardware</td>
<td>32</td>
</tr>
<tr>
<td>3.2.1 Raspberry pi</td>
<td>32</td>
</tr>
<tr>
<td>3.2.2 Cámara RaspiCam</td>
<td>33</td>
</tr>
<tr>
<td>3.2.3 Puente H L298</td>
<td>33</td>
</tr>
</tbody>
</table>
Índice de Tablas.

Tabla 1: Modos de codificación ................................................................. 19
Tabla 2: Nivel de corrección de errores .................................................. 20
Tabla 3: Indicador de largo. ................................................................... 21
Tabla 4: Caracteres alfanuméricos y sus respectivos valores. ................. 22
Tabla 5: Tabla de verdad puente H ............................................................ 40

Índice de figuras:

Fig. 1: Carta Gantt ................................................................................... 13
Fig. 2: Códigos Modelo 1 y 2 ................................................................. 17
Fig. 3: Micro QR ..................................................................................... 17
Fig. 4: IQR. ............................................................................................. 18
Fig. 5: QR Marco ................................................................................... 19
Fig. 6: Patrones de búsqueda ............................................................... 24
Fig. 7: Separadores ............................................................................... 24
Fig. 8: Patrón de alineamiento ............................................................... 25
Fig. 9: Patrones de temporizado ............................................................ 25
Fig. 10: Áreas Reservadas ..................................................................... 26
Fig. 11: Áreas reservadas versión 7 ....................................................... 26
Fig. 12: Llenado descendente y ascendente .......................................... 27
Fig. 13: Llenado del código .................................................................. 27
Fig. 14: Salto de obstáculo Vertical ...................................................... 28
Fig. 15: Salto de obstáculo Horizontal ................................................ 28
Fig. 16: Tipos de mascara y binario identificador .................................. 29
Fig. 17: Distribución de la máscara dentro del código. ......................... 30
Fig. 18: Materiales y presupuesto ......................................................... 31
Fig. 19: Raspberry pi 3 y pinout ............................................................. 32
Fig. 20: Especificaciones Raspberry pi 3 ............................................. 32
Fig. 21: Cámara Raspberry Pi ................................................................. 33
Fig. 22: Modulo puente H L298 2A ....................................................... 34
Fig. 23: Vnc viewer............................................................................... 35
Fig. 24: SSH ......................................................................................... 35
Fig. 25: ZBar ......................................................................................... 36
Fig. 26: Diseño correa transportadora .................................................. 38
Fig. 27: Conexión Raspberry-Relé y sus dispositivos controlados ......... 39
Fig. 28: Conexiones entre Raspberry y Puente H ................................ 40
Fig. 29: Esquema de conexión del sensor IR ........................................ 41
Fig. 30: Esquema sistema clasificador de artículos mediante códigos QR 41
Fig. 31: Diagrama de bloques con la relación software-hardware .................. 43
Fig. 32: Diagrama de flujo del programa Python ............................................. 44
Fig. 33: Esquemático del proceso de toma de imagen y decodificación QR ...... 45
Fig. 34: Diseño en Autocad Palanca ............................................................ 47
Fig. 35: Hipotenusa en un cuadrado. ............................................................. 47
Fig. 36: Sentido de giro de motor según polarización ..................................... 47
Fig. 37: Captura de código QR por parte de la cámara del sistema .................. 48
Fig. 38: Red de prueba .................................................................................. 49
Fig. 39: Tabla de notas de venta ..................................................................... 49
Fig. 40: Tabla de comunas ............................................................................. 50
Fig. 41: Registro tabla Notas (de venta) de BD despachos ............................. 51
Fig. 42: Frecuencias por campo "estado". ....................................................... 52
Fig. 43: Tabla de frecuencias campo "Comunas". ............................................. 52
Fig. 44: Tabla de resultados de las pruebas. .................................................. 53
Fig. 45: Tabla de frecuencias ......................................................................... 54
Fig. 46: Tabla de frecuencia de errores. ......................................................... 54
Fig. 47: Versión seleccionada. ....................................................................... 73
Fig. 48: Modo seleccionado. ......................................................................... 74
Fig. 49: Tabla de división y reducción de términos de x .................................. 76
Fig. 50: Resultado aplicación criterio n°1 horizontal. ..................................... 77
Fig. 51: Resultado aplicación criterio n°1, vertical ........................................... 78
Fig. 52: Resultado aplicación criterio n°2, bloques 2x2. ................................. 78
Fig. 53: Resultado aplicación criterio n°3, bloques 2x2. ................................. 79
Fig. 54: Tabla de distribución de la máscara dentro del código ..................... 82
Fig. 55: Código resultante ............................................................................. 82
Resumen

El control de un inventario es un problema que existe desde los comienzos de la producción en serie. Largos turnos de trabajadores fueron usados en un comienzo generando inventarios de forma manual, sin embargo, con el avance de la tecnología, esta tarea se ha ido automatizando, disminuyendo el tiempo ocupado y teniendo la ventaja de obtener resúmenes y estadísticos de los objetos clasificados en menos tiempo y con mayor precisión (Osorio, 2013).

En el presente proyecto se propone el diseño de un prototipo de clasificador basado en tecnología QR implementado sobre Raspberry-pi 3 utilizando librería Zbar para Python 2.7

Actualmente, existen diversas opciones de clasificadores, pero la mayoría son de código propietario, impidiendo al usuario hacer modificaciones acorde a sus necesidades.

Se propone un sistema de bajo costo, con herramientas open source, que permitirán al usuario final modificar e implementar nuevas funcionalidades.

Se incluye también la implementación de una base de datos MySQL, la cual registra cada ingreso de productos, además del accionamiento de una banda transportadora y actuadores lineales conectados a pines de propósito general, los cuales permitirán el desplazamiento del producto a un destino determinado.

Se realizaron pruebas que demostraron una eficacia del 90% respecto a la identificación del código y posterior movimiento de este dentro de la banda transportadora a su destino final.
Abstract

The control of an inventory is a problem that exists since the beginning of serial production. Long shifts of workers were used at the beginning, however, with the advancement of technology, this task has been automated, reducing the time spent and having the advantage of obtaining summaries and statistics of classified objects.

In this project we present the design of a classifier prototype based on QR technology implemented on Raspberry-pi 3 using Zbar library for Python 2.7

Currently, there are several options for classifiers, but most are proprietary code, preventing the user from making modifications according to their needs.

In this project, a low-cost system is proposed, with open source tools, which will allow the end user to modify and implement new functionalities.

It also includes the implementation of a MySQL database, which records each product entry, in addition to the drive of a conveyor belt and linear actuators connected to the general purpose pins, which will allow the product to be moved to a specific destination.

Tests were carried out that demonstrated an efficiency of 90% with respect to the identification of the code and its subsequent movement within the conveyor belt to its final destination.

Keywords: Automatic sorter, zbar, QR Codes.
1 Antecedentes generales

1.1 Introducción

Desde comienzos de la revolución industrial y automatización de tareas repetitivas, se ha buscado la forma óptima de clasificar objetos y disminuir los tiempos de conteo de inventario y resumen de estadísticos, tarea nada de fácil teniendo en cuenta la naturaleza cambiante de estos (Osorio, 2013).

Con el avance de la tecnología los tiempos en que se podía contar con las diversas mercancías se fue haciendo cada vez más corto, pero con el desarrollo de la sociedad y las actividades de sus integrantes, el tiempo se fue haciendo un recurso cada vez más limitado, así mismo la necesidad de saber con qué artículos se contaba y que faltaba en corto tiempo cobró gran relevancia.

Desde el principio el tiempo fue una barrera a derribar, buscando rutas más directas y rápidas hacia los mercados, mejorando los medios de transporte y comunicación.

Con el crecimiento de las ciudades, la necesidad de atender rápidamente al público en las grandes tiendas y supermercados, supuso un nuevo reto para la tecnología. Buscando hacer más rápido el proceso de facturación en las cajas, en 1932 Wallace Flint, un estudiante de negocios de Harvard en su tesis proponía tarjetas perforadas con las preferencias de compras del cliente que luego serían despachadas automáticamente. Pero lamentablemente la idea no resultó económicamente viable para la época, era la Gran Depresión.

Posteriormente, en 1948 un estudiante de posgrado llamado Bernard Silver fue influenciado por una investigación conducida por la universidad sobre un sistema automatizado para recibir información de los productos en caja, a idear un código,
para ello, junto a su amigo Norman Joseph Woodland, desarrolló un sistema de patrones de tinta susceptibles de ser leídos con una luz ultravioleta pero la idea no prosperó. Woodland continuó solo hasta lograr un código de barras basado en el código morse y junto a Flint quien también había continuado su investigación por separado presentaron una solicitud de patente el 20 de octubre de 1949. Este sería el nacimiento del código de barras.

La necesidad de contener más información impulsó la creación de códigos bidimensionales, entre ellos el código QR. Los códigos QR (Quick Response Code) en que se representa la información en una matriz de puntos bidimensional, fueron creados en Japón por la compañía DENSO Wave.

Los códigos QR son rápidos y económicos de producir, y pueden ser leídos con cualquier cámara. Esto puede ser muy conveniente para desarrollar sistemas de bajo costo para clasificar y derivar productos ya sea para completar pedidos o para ordenarlos en las góndolas de almacenamiento correspondientes, o como código de caja registradora, como el de barras.

En la actualidad, que el comercio aumenta sus ventas incorporando el despacho a domicilio, la logística de las entregas debe ser tratada de un modo mucho más ágil, he aquí una propuesta para la pequeña y mediana empresa, donde los recursos no son tan abundantes.
1.2 Objetivos

1.2.1 Objetivo general

Diseñar un prototipo de clasificador basado en lecturas y decodificación de códigos QR utilizando zbar en Python, sobre una placa Raspberry PI, tal que almacene registros en una base de datos MySQL y que tenga la capacidad de mostrar estadísticos y accionar actuadores que deriven cada paquete a una ubicación específica según su identificador.

1.2.2 Objetivos específicos

- Desarrollar programa que pueda interpretar códigos QR
- Configurar base de datos MySQL utilizando Raspberry-pi
- Almacenar datos capturados
- Obtener estadísticos de interés
- Medir el porcentaje de acierto o error del dispositivo

1.3 Carta Gantt

<table>
<thead>
<tr>
<th>TAREAS</th>
<th>ago-17</th>
<th>sep-17</th>
<th>oct-17</th>
<th>nov-17</th>
<th>dic-17</th>
<th>ene-18</th>
<th>feb-18</th>
<th>mar-18</th>
<th>abr-18</th>
<th>may-18</th>
<th>jun-18</th>
</tr>
</thead>
<tbody>
<tr>
<td>Diseño de la idea</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Selección de componentes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Instalación de prequisitos</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Decodificacion QR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Electrónica del clasificador</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Programación del clasificador</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Implementación base de datos</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Desarrollo de estadísticas</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Puesta en marcha y validacion</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Fig. 1: Carta Gantt
1.4 Organización y presentación del trabajo

En el primer capítulo, se trata de hacer claridad sobre los códigos QR, que son, para que se inventaron, como funcionan.

El segundo capítulo, está dedicado a la visión computacional, aquí se intentará hacer un poco de claridad respecto a este tópico.

Aclarados en los primeros capítulos los temas más destacados de este proyecto, se describirá éste en profundidad en el capítulo 3°, donde se detallarán las otras tecnologías utilizadas y se presentaran resultados de su implementación experimental.

1.5 Estado del Arte

El primer uso que se les dio a los códigos QR, fue el de seguimiento de piezas y partes en la industria automotriz. Sus creadores, Denso Wave son parte del grupo Nippon Denso que son grandes proveedores de piezas para la industria automotriz.

Los códigos QR gracias a su gran capacidad de almacenamiento, en la actualidad son utilizados principalmente con propósitos promocionales e informativos.

Los códigos QR gracias a su gran capacidad de almacenamiento, en la actualidad son utilizados principalmente con propósitos promocionales e informativos, no obstante se ha experimentado con ellos para la encriptación y des encriptación óptica libre de ruidos de imágenes en escala de grises (Shuming Jiao, et al 2010).
Otros estudiantes chinos abordaron el tema de la seguridad en la encriptación y la desencriptación de código QR e imagen en escala de grises con esquemas basados en interferencia con recuperación de alta calidad y resolución del problema de siluetas (Yi, 2016), este último como un sistema seguro de encriptación de imágenes a través de internet, dado que con los sistemas tradicionales de encriptado basados en interferencia aún permiten que se distinga una silueta de la imagen.

En la universidad de Sevilla se han desarrollado programas de innovación en la formación de docentes para llevar la tecnología a las aulas. Los resultados reflejan que las actividades planteadas con códigos QR “nos ofrecen una nueva oportunidad de innovar en el aula desde distintas áreas y entornos de aprendizaje, haciendo más atractivas las asignaturas” (R.Graván, et al 2013).

Según un estudio de alumnos de la Universidad Católica de Chile, el público que visita muestras como museos y otros prefieren métodos directos para obtener información sobre la exhibición como son información en un panel o videos en un monitor pero a la vez descubrieron que después de los medios tradicionales los códigos QR son la forma más económica de entregar contenido digital en este tipo de lugares, en especial para jóvenes en edad escolar (Nussbaum M, et al 2016).

2. Marco Teórico

2.1 Códigos QR

En lo que respecta a clasificación de inventario, hace no mucho, los códigos de barra eran la principal opción, hasta que la industria se encontró con su limitante de 20 caracteres, lo cual significó un problema al agregar descripciones detalladas de productos, “entonces surgieron las inquietudes de: en un mismo
código quisiera que tuviera más información, o, quisiera que se pudieran expresar Kanji y Kana (caracteres japoneses)” (Denso Wave, 2018).

Los códigos QR a diferencia de los códigos de barra, son capaces de almacenar decenas o incluso miles de datos en un espacio reducido.

Por otro lado, los códigos QR pueden almacenar distintos tipos de información tales como texto, caracteres, números o incluso archivos enteros, a diferencia de los códigos de barra, que solo pueden almacenar números.

Los códigos QR almacenan información en sus dos dimensiones (alto y ancho), por lo que la misma información que podría contener un código de barras, el QR la contiene en una enésima parte de espacio.

Aunque una parte del código sea destruido o manchado es posible recuperar la información contenida en este gracias a su función de corrección de errores, con la cual se puede recuperar un aproximado de 30% de código destruido.

Gracias a sus tres marcas características, el código QR puede ser leído a alta velocidad en cualquier posición.

Un código QR muy grande puede ser dividido en hasta 16 partes, gracias a esta funcionalidad se puede imprimir códigos en áreas estrechas, luego el código dividido en varios códigos pequeños puede ser vuelto a unificar.
2.2 Tipos de códigos QR

2.2.1 Código QR modelo 1 y modelo 2.

El modelo 1 es el primer modelo creado. Su versión más grande es la 14 (73x73 celdas) y puede contener hasta 1167 caracteres alfanuméricos.

El modelo 2 es la mejora del modelo 1, su versión mayor es la 40 (177x177 celdas) contiene hasta 7089 caracteres alfanuméricos. Actualmente al hablar de códigos QR, normalmente se está hablando del modelo 2.

2.2.2 Micro QR.

Su versión mayor es la M4 (17x17 celdas) y puede manipular 35 caracteres alfanuméricos. Posee solo una de sus marcas características, es un código que se puede imprimir en espacios más pequeños. Su margen también se reduce a
solo dos celdas. Los códigos QR requieren un margen de 4 celdas en torno al código.

2.2.3 Código iQR.

Fig. 4: iQR.

Dentro de sus posibles formas esta la cuadrada (normal) o alargada, puede ser invertida (vista de atrás) o con los colores de celda invertidos. En teoría su máxima versión es la 61, 422x422 celdas, en las que se podrían contener cerca de 40 mil caracteres alfanuméricos.

2.2.4 SQRC

Este código tiene privilegios de lectura incorporados, sirven para administrar comunicaciones privadas o información interna de las empresas. Su apariencia es de un código normal.

2.2.5 Código QR marco.

Dentro del código existe un “lienzo” que se puede usar con toda libertad. En la porción correspondiente al lienzo se puede imprimir fotos, caracteres, dibujos, etc.
2.3 Codificación y decodificación QR

El código QR codifica la información en cadenas de texto en cinco modos. En todos los modos la información se codifica en bits, pero cada modo utiliza un método diferente para convertir la información en bits. Cada modo está optimizado para generar la menor cadena de caracteres posible. Los modos de conversión de la información son los siguientes:

<table>
<thead>
<tr>
<th>Modo</th>
<th>Indicador</th>
<th>Contenido</th>
</tr>
</thead>
<tbody>
<tr>
<td>Numérico</td>
<td>0001</td>
<td>Dígitos del 0 – 9</td>
</tr>
<tr>
<td>Alfanumérico</td>
<td>0010</td>
<td>Dígitos y las letras mayúsculas, y los símbolos $, %, *, +, -, /, .</td>
</tr>
<tr>
<td>Byte</td>
<td>0100</td>
<td>Conjunto de caracteres de la ISO-8859-1</td>
</tr>
<tr>
<td>Kanji</td>
<td>1000</td>
<td>Kanji Escritura japonesa, conjunto de caracteres Shift JIS.</td>
</tr>
<tr>
<td>ECI</td>
<td>0111</td>
<td>Variante modo Byte.</td>
</tr>
</tbody>
</table>

Tabla 1: Tabla 1 Modos de codificación

Para elegir el modo más eficiente de codificación, se debe observar la cadena de
caracteres a codificar y de acuerdo a su largo y tipo de información seleccionar uno de los modos especificados en el párrafo anterior.

2.3.1 Codificación de la información.

La primera parte de la codificación de la información es seleccionar el nivel de corrección de errores. Éste está basado en el sistema de corrección Reed-Solomon. Este proceso crea palabras de corrección (bytes) en función de la información codificada. El lector de QR puede determinar por medio de estas palabras si la información está correcta y/o como corregirla. Los 4 niveles de corrección de errores son:

<table>
<thead>
<tr>
<th>Nivel</th>
<th>Capacidad de Corrección</th>
</tr>
</thead>
<tbody>
<tr>
<td>L</td>
<td>Recupera 7% de información</td>
</tr>
<tr>
<td>M</td>
<td>Recupera 15% de información</td>
</tr>
<tr>
<td>Q</td>
<td>Recupera 25% de información</td>
</tr>
<tr>
<td>H</td>
<td>Recupera 30% de información</td>
</tr>
</tbody>
</table>

Tabla 2: Nivel de corrección de errores.

En segundo lugar se debe seleccionar el código con el tamaño (cantidad de caracteres) apropiado para contener la información a codificar.

Cada tamaño está tabulado y recibe el nombre de “Versión”. Cada versión es 4 píxeles mayor que el anterior, así tenemos que la versión 1 es de 21x21 píxeles y la versión 2 es 25x25 píxeles, actualmente se tabula hasta la versión 40 de 177x177 píxeles.

Cada versión posee una capacidad específica según el tipo de información (modo), esta capacidad varía de acuerdo al nivel de corrección de errores. Esto quiere decir que mientras mayor es el nivel de corrección de errores menor
la cantidad de información codificable dentro del código QR.

Para mostrar el procedimiento, se codificará: “U. ANDRES BELLO” (Calculo disponible en la sección de anexos).

Se selecciona el modo de la tabla de modos de codificación, en este caso 0010, modo alfanumérico

Se determina el largo de la cadena de caracteres a codificar.

<table>
<thead>
<tr>
<th>Versiones</th>
<th>1 a 9</th>
<th>10 a 26</th>
<th>27 a 40</th>
</tr>
</thead>
<tbody>
<tr>
<td>Numérico</td>
<td>10 bits</td>
<td>12 bits</td>
<td>14 bits</td>
</tr>
<tr>
<td>Alfanumérico</td>
<td>9 bits</td>
<td>11 bits</td>
<td>13 bits</td>
</tr>
<tr>
<td>Byte</td>
<td>8 bits</td>
<td>16 bits</td>
<td>16 bits</td>
</tr>
</tbody>
</table>

Tabla 3: Indicador de largo.

Separa la cadena de caracteres seleccionada en pares y se le asigna su número de carácter según la tabla de caracteres alfanuméricos presentada a continuación.
2.3.2 Códigos de corrección de errores.

Se utilizan estos códigos para detectar y corregir errores la lectura. Los errores se pueden producir por ejemplo, debido a la destrucción del código o ruidos producidos por suciedad.

Cabe mencionar que dependiendo del largo de la información a codificar, esta se divide en grupos, cada grupo se divide en bloques y por ultimo estos en palabras (bytes), de acuerdo con la tabla de palabras de corrección de errores e información de bloques.

Para crear el código de corrección de errores, se utiliza el método Reed-Solomon que consiste en aplicar una serie de operaciones a cada palabra de datos y campos de Galois. No es parte de esta tesis explicar este método a fondo, solo se mencionará que la principal operación que se realizará es la división polinómica.

Como se mencionó anteriormente se utilizara Campos de Galois (CG) en
específico CG(256), en rango de números del 0 al 255 que es el rango representable por un byte de 8 bits (11111111 que es el equivalente binario de 255). Toda operación realizada sobre un número del campo debe resultar en un número dentro del campo, por eso si el número resultante es mayor que 255, se le aplica el módulo 285, que es un XOR con el número 285 (100011101 en binario).

2.3.3 División Polinómica.

Este procedimiento es muy importante, porque de él resultaran las palabras de corrección de errores. Se sabe que para dividir polinomios, se necesitan 2 polinomios. El primer polinomio se genera a partir de la información codificada en bits y dividida en palabras de 1 byte (8 bits), este polinomio se llama “Polinomio de Mensaje”.

El segundo polinomio se denomina “Polinomio Generador” y se crea mediante la operación:

\[(x - \alpha^0) \cdots (x - \alpha^{n-1})\].

Donde “n” es el número de palabras de corrección de errores y \(\alpha\) (alfa) es igual a 2 y si sabemos que el coeficiente de X es 1 por lo tanto es lo mismo que decir:

\[(\alpha^0 x - \alpha^0)(\alpha^0 x - \alpha^{n-1})\]

2.3.4 Patrones de búsqueda

Son los tres bloques posicionados en las esquinas superiores izquierda y derecha, y en la esquina inferior izquierda del código, los cuales tienen como función
cuadrar la lectura del código en caso de estar en posición oblicua o recta.

2.3.5 Separadores

Franjas blancas en torno al ángulo interior de los Patrones de búsqueda que separan a éstos del resto del código.

2.3.6 Patrón de alineamiento

Similar al Patrón de búsqueda, menor en tamaño, posicionados dentro de los códigos versión 2 en adelante. Su punto central coincide con el punto correspondiente al punto de la esquina interior de los Patrones de búsqueda superior izquierdo e inferior, la cantidad de estos patrones depende de la versión del código.

Cuando se da el caso de más de un Patrón de alineamiento, la ubicación de éstos es el reflejo horizontal y vertical del patrón ya descrito, utilizando como ejes de
simetría los ejes del código, y los puntos medios de los ejes de los propios patrones, eliminándose los que se superponen a los patrones de búsqueda.

2.3.7 Patrones de temporizado.

Líneas punteadas que unen los Patrones de búsqueda por el punto correspondiente a las esquinas interiores de éstos.

2.3.8 Áreas reservadas.

Ciertas franjas del código son reservadas para la información de formato y versión, las áreas de formato reservadas son las siguientes:
El contorno adyacente al separador superior izquierdo, la línea bajo el separador superior derecho y la columna contigua al separador inferior, bajo el modulo oscuro.

Fig. 10: Áreas Reservadas.

Para los códigos versión 7 y superior, las áreas de versión reservadas son:

El bloque de 3x6 anterior al separador superior derecho y el bloque de 6x3 sobre el separador superior inferior.

Fig. 11: Áreas reservadas versión 7

2.3.9 Ubicación de la información en el código.

El llenado de la matriz con la información debe seguir un patrón determinado, que se ve interrumpido cada vez que se encuentra con un área reservada o de función
del código.

La matriz se comienza a llenar a partir de la esquina inferior derecha, un punto hacia la izquierda, luego un punto en diagonal hacia arriba a la derecha, un punto a la izquierda, para luego repetir en forma cíclica este procedimiento, hasta terminar la columna.

Una vez alcanzado el límite superior de la columna, se continúa los dos siguientes puntos hacia la izquierda y se desciende uno en diagonal a la derecha, un punto a la izquierda y se repite este patrón descendente de uno hacia abajo a la derecha y un punto a la izquierda hasta el límite inferior.

Así las doble columnas se van sucediendo en zigzag hasta arriba, a la izquierda hasta abajo a la izquierda y vuelve a repetir hasta llenar el código.
Cuando se encuentra en el camino con un patrón de función, se continúa en línea recta por el lado del patrón hasta el punto en que se puede volver al patrón en zigzag.

![Fig. 14: Salto de obstáculo Vertical](image)

Cuando se alcanza la línea temporizadora vertical, esta se salta y se continua adyacente a esta en forma normal.

![Fig. 15: Salto de obstáculo Horizontal](image)

2.3.10 Enmascarado de Información.

Esto, básicamente, consiste en sumarle un 1 a cada bit, sin acarreo, con lo que cada bit es cambiado a su estado contrario, si es 0 se vuelve 1 y viceversa.

En el estándar QR existen 8 patrones de enmascarado los que se aplican solo a los módulos de información y de corrección de errores, no a los patrones de función ni a las áreas reservadas.
Los patrones de enmascarado se pueden observar en la siguiente imagen:

![Imagen de patrones de máscara](image)

Fig. 16: Tipos de mascara y binario identificador

Con el patrón de mascara se llena toda el área del código, restándose o recortándose las zonas reservadas, luego se realiza la suma binaria (sin acarreo) con el código obtenido en los pasos anteriores. Como se mencionó anteriormente la máscara no afecta las zonas reservadas.

Si bien existen reglas para seleccionar las mascara, esta podría ser seleccionada arbitrariamente por el codificando, ya que no hay una confirmación si la máscara es la que corresponde o no a ese código en particular.

2.3.11 Reglas para seleccionar la Máscara.

Para seleccionar la máscara de acuerdo al protocolo QR, el código obtenido hasta aquí, es evaluado aplicándole las 8 mascaras ya mencionadas anteriormente (numeradas del 0 al 7).

Una vez aplicadas las máscaras se aplican 4 criterios de evaluación, cada uno asigna un puntaje al código enmascarado. La máscara que sume el puntaje más
bajo de los cuatro criterios, será la máscara a utilizar.
La evaluación se aplica también a las zonas reservadas y de función.

2.3.12 Distribución dentro del código.

Fig. 17: Distribución de la máscara dentro del código.
3 Desarrollo

3.1 Costos y presupuestos

<table>
<thead>
<tr>
<th>DESCRIPCION</th>
<th>COSTO</th>
</tr>
</thead>
<tbody>
<tr>
<td>RASPBERRY PI3</td>
<td>40000</td>
</tr>
<tr>
<td>PANTALLA HDMI 7&quot;</td>
<td>30000</td>
</tr>
<tr>
<td>PLACAS PUENTE H</td>
<td>10000</td>
</tr>
<tr>
<td>TRANSFORMADOR 2A</td>
<td>4000</td>
</tr>
<tr>
<td>CABLE HDMI</td>
<td>5000</td>
</tr>
<tr>
<td>ADAPTADOR PINES RASPBERRY</td>
<td>5000</td>
</tr>
<tr>
<td>MOTORES</td>
<td>20000</td>
</tr>
<tr>
<td>MAQUETA MAQUINA</td>
<td>10000</td>
</tr>
</tbody>
</table>

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>124000</td>
</tr>
</tbody>
</table>

Fig. 18: Materiales y presupuesto
3.2 Hardware

3.2.1 Raspberry pi

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>PROCESADOR</strong></td>
<td>Broadcom BCM2837, Cortex-A53 (ARMv8) 64-bit</td>
</tr>
<tr>
<td><strong>FRECUENCIA DE RELOJ</strong></td>
<td>1.2 GHz</td>
</tr>
<tr>
<td><strong>GRÁFICOS</strong></td>
<td>VideoCore IV 400 MHz</td>
</tr>
<tr>
<td><strong>MEMORIA</strong></td>
<td>1GB LPDDR2 SDRAM</td>
</tr>
<tr>
<td><strong>CONECTIVIDAD</strong></td>
<td>2.4 GHz</td>
</tr>
<tr>
<td><strong>INALÁMBRICA</strong></td>
<td>IEEE 802.11 b/g/n</td>
</tr>
<tr>
<td></td>
<td>Bluetooth 4.1</td>
</tr>
<tr>
<td><strong>CONECTIVIDAD DE RED</strong></td>
<td>Fast Ethernet 10/100 Mbps</td>
</tr>
<tr>
<td><strong>PUERTOS</strong></td>
<td>GPIO-40 pines</td>
</tr>
<tr>
<td></td>
<td>HDMI</td>
</tr>
<tr>
<td></td>
<td>4x USB 2.0</td>
</tr>
<tr>
<td></td>
<td>CSI (diseno Raspberry Pi)</td>
</tr>
<tr>
<td></td>
<td>DSI (pantalla LCD)</td>
</tr>
<tr>
<td></td>
<td>Toma auriculares / Video compuesto</td>
</tr>
<tr>
<td></td>
<td>Micro SD</td>
</tr>
<tr>
<td></td>
<td>Micro USB (alimentación)</td>
</tr>
</tbody>
</table>

![Fig. 19: Raspberry pi 3 y pinout](image1)

![Fig. 20: Especificaciones Raspberry pi 3](image2)
3.2.2 Cámara RaspiCam

![Cámara Raspberry Pi](image)

Fig. 21: Cámara Raspberry Pi.

- Lente de distancia focal fija
- Resolución nativa de 8MP.
- Resolución fotos de 3260x2464
- Resolución video 1080p30, 720p60, 640x480p90
- Tamaño: 25mm x 23mm x 9mm
- Peso: 3g
- Se conecta a la Raspberry Pi a través de un pequeño cable plano proporcionado
- Compatible con la última versión de Raspbian

3.2.3 Puente H L298

Este módulo tiene por finalidad accionar y brindar la potencia suficiente a los motores que se encargaran de la transmisión y clasificación de los paquetes. La placa cuenta con jumpers de selección para habilitar cada una de las salidas del módulo (A y B). La salida A está conformada por OUT1 y OUT2 y la salida B por OUT3 y OUT4. Los pines de habilitación son ENA y ENB respectivamente.
Fig. 22: Modulo puente H L298 2A

Características:

• Doble Puente H
• Circuito Integrado L298N
• Voltaje lógico: 5V
• Voltaje de operación: 5 – 35V
• Corriente control: 0 – 36mA
• Máxima corriente de operación: 2A (por motor)
• Temperatura de operación: -20 to +135 C
• Máxima potencia a disipar: 25W
3.3 Software

3.3.1 VNC Viewer

Fue utilizado para tener acceso en forma remota a la Raspberry y poder programar desde un computador externo.

![Vnc viewer](image1)

Fig. 23: Vnc viewer.

3.3.2 SSH

Al igual que VNC, sirve para acceder a un servidor remoto y ejecutar comandos en él solo con la diferencia que es un protocolo de acceso a consola y no a escritorio remoto.

![SSH](image2)

Fig. 24: SSH
3.3.3 Librerías utilizadas

La librería ZBar es una librería orientada a la lectura y decodificación de códigos de Barra y QR.

Comandos e instrucciones ZBar

zbar.ImageScanner(): Esta función crea un objeto scanner para explorar una imagen en busca de códigos QR.

zbar.Image(anch, alto, 'Y800', imagen): Convierte una imagen a un formato compatible con ZBar, de medidas ancho*alto en formato “Y800”.
El “Y800” es un formato de 8 bits, monocromático.

Objeto.scan(imagen): Esta función explora y encuentra los códigos QR en la imagen y los almacena en imagen. “Objeto” es el nombre de un objeto creado con la función zbar.ImageScanner().
4. Desarrollo

4.1 Metodología

Se comenzará creando una maqueta de banda transportadora, que es indispensable para la prueba del proyecto propuesto.

Luego, se desarrollará el sistema electrónico y mecánico que dotará de movimiento a la banda y a los sistemas de accionamiento de los clasificadores.

Tras esto se implementará el sistema de captura y decodificación basada en Zbar en python 2.7

Posteriormente, se creara la base de datos MySQL y se enlazara al código anterior.

Finalmente, se implementara un sistema en PHP y HTML el cual entregue los estadísticos de interés

4.2 Maqueta

Si bien es cierto, el diseño y fabricación de la correa transportadora, no tiene que ver con el tema planteado en esta tesis, si es importante para la puesta en marcha del experimento y la obtención de resultados, y como se desarrolló especialmente para este proyecto además de ser una automatización más, se merece una pequeña mención.

Este no pretende ser un tratado sobre la fabricación de bandas transportadoras, pero dejó algunos conocimientos obtenidos empíricamente de esta manualidad.
Se escogió un largo de un metro por estimarse una longitud suficiente para que los componentes cupieran y hubiese espacio suficiente para alimentar de artículos de prueba al sistema.

Fue fabricada a partir de 2 trozos de madera y dotado con la cantidad de rodillos suficientes para sustentar los paquetes que se le montarían sin sufrir demasiada flexión en la banda, y esta fue cortada a partir de un yoga mat (tapete para hacer yoga) que es un material similar a la goma eva bastante elástica.

Es importante que los ejes estén en 90° respecto a los soportes laterales para que la banda no se desvíe y se salga de los rodillos conductores, que son los rodillos extremos, de ellos depende que la banda funcione sin problemas.

La banda no debe quedar muy tensa para no sobrecargar el eje del motor y así mejorar su desempeño y vida útil.

De todas formas, la banda transportadora fue diseñada en AutoCAD con la finalidad de eventualmente ser replicada en un futuro para trabajos próximos.

![Fig. 26: Diseño correa transportadora.](image)
4.3 Hardware

Para este proyecto se necesita controlar 3 motores:

El motor principal, conectado a un relé, que es el motor que mueve la banda transportadora, es controlado por PWM. Para conseguirlo se declara el pin GPIO24 como salida y se define como PWM, con un ciclo de 10 Hz, y un ancho de pulso de 20%. En realidad el ancho de pulso del PWM es de 80%, pero como el relé acciona en baja, el pulso para el vale 20%.

En la figura las líneas verdes representan señales de control, ambas de salida de la Raspberry, GPIO23 y 24 están configurados de salida.

Los relés se enumeran de arriba hacia abajo, se utilizan los relés 2(illuminación) y 3 (motor de la banda transportadora).

Los dos motores restantes controlan 2 pistones eléctricos, que realizan el
accionamiento del mecanismo de expulsión de los paquetes desde la banda transportadora. Estos pistones deben realizar la acción de extensión y retracción por lo que se hace necesario cambiar su sentido de giro, por eso se eligió el puente H.

Fig. 28: Conexiones entre Raspberry y Puente H

Las líneas verdes son señales de control se necesitan 2 por cada motor (del pistón), una por cada sentido de giro.

Tenemos entonces en este esquema que los GPIO 5, 6, 13 y 19 están configurados como salida.

<table>
<thead>
<tr>
<th>Salidas</th>
<th>Valor de la salida</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPIO (5 y 13)</td>
<td>1 0 1 0</td>
</tr>
<tr>
<td>GPIO (6 y 19)</td>
<td>0 1 1 0</td>
</tr>
<tr>
<td>Giro horario</td>
<td>antihorario</td>
</tr>
<tr>
<td>no</td>
<td>no</td>
</tr>
</tbody>
</table>

Tabla 5: Tabla de verdad puente H.

Por último, el sensor infrarrojo que cumple una función de obturación. Al detectar una caja llama a la cámara que toma una fotografía para posteriormente
decodificar el QR contenido en ella.

En la imagen se puede apreciar la conexión a 5v, el neutro y la salida (del sensor) al pin GPIO4 de la Raspberry configurado como entrada.

4.4 Clasificador

A continuación, se presenta un esquema del clasificador y sus componentes principales:

1. Banda transportadora
2. Cámara
3. Pistón de eyeción derecho
4. Pistón de eyección izquierdo
5. Almacenamiento temporal empaques separados.
6. Paquete

Para clasificar objetos se recurre a una de sus características, como: forma, color, tamaño, etc. En este proyecto se pone el foco en un código impreso en un empaque. Este código no es cualquier código, es un código QR, el que puede ser leído con cualquier cámara o celular con el software adecuado.

La tarea de la cámara es detectar el código QR del empaque sobre la banda transportadora, en el cual se entiende, contiene información tal como: dirección, cliente, n° de pedido, etc. En este caso la información que nos interesa es la comuna.

Se ha dividido el gran Santiago en 3 zonas (poniente, centro y oriente) y sus comunas cuentan con la asociación a su respectivo código de zona (1, 2 y 3 respectivamente)
4.5 Software

La cámara toma una imagen del código QR sobre la banda transportadora, la Raspberry Pi 3, gracias al programa Python instalado en su memoria, decodifica el código, hace las consultas MySQL, para guardar los datos que nos servirán para hacer las estadísticas del sistema y hacer los ajustes de inventario correspondientes.

Fig. 31: Diagrama de bloques con la relación software-hardware
4.5.1 Diagrama de flujo

El flujo del programa se puede representar de la siguiente forma: una vez la cámara detecta el código, lo pasa al procesador que lo decodifica y acciona el pistón correspondiente si ese es el caso. Se pueden dar 3 casos, que son:

El código de zona es 1, el empaque será sacado de la banda por un pistón hacia la derecha.

El código de zona es 2, el empaque será empujado sacándolo de la banda hacia la izquierda.

Si es 3, se dejará el empaque libre que salga por el extremo de la banda.

Fig. 32: Diagrama de flujo del programa Python
4.5.2 Captura y decodificación

Fig. 33: Esquemático del proceso de toma de imagen y decodificación QR
4.5.3 Accionamiento del eyector

El pistón eléctrico es el accionamiento de esta máquina encargado de derivar los paquetes a sus respectivos destinos al sacarlos de la banda transportadora

El piston tiene un rango de accion de 20mm, por lo que debe desarrollar un sistema para amplificar su recorrido, para que sea capaz de mover el empaque a todo lo ancho de la banda, 120mm aproximadamente. Se visualiza como una palanca.

Para determinar el largo necesario de la palanca, se debe multiplicar el rango del piston para que la palanca unida a un pivote amplifique su accion a 120mm, para es decir 6 veces la longitud de accionamiento del pistón por si solo.
Por ser A y B ángulos opuestos por el vértice y de catetos iguales entre sí, entonces sabemos que la hipotenusa del triángulo menor es:

\[ x = \sqrt{2 \times x^2} \]

Esto es \( x = 2,83\text{cm} \), además se sabe que son correspondientes y sus medidas son escalas el uno del otro, y a simple vista se ve que el triángulo pequeño representa la sexta parte del mayor, por lo tanto sabemos que la hipotenusa del mayor (largo de la palanca), debe ser 6 veces este valor, es decir: 16,87cm.

Si desde el pibote hacia arriba mide 2,83 y hacia abajo vale 16,87, el largo mínimo de la palanca debe ser: 19,7cm.

El pistón accionado por motor eléctrico debe ejecutar la acción de extensión y retracción, por lo que se debe hacer girar su motor en un sentido y luego en otro.

Se sabe que el giro de un motor dc depende de la conexión positivo negativo, como se muestra en la imagen.
Cuando la caja llega hasta el dispositivo de eyecación el pistón se retrae accionando la palanca que empuja el paquete fuera de la banda, para luego extenderse y volver al punto de reposo, para esto es necesario cambiar dirección de giro del motor del pistón. Para invertir el giro durante este proceso se requiere de un puente H.

Fig. 37: Captura de código QR por parte de la cámara del sistema.

Durante la ejecución del programa es posible ver por un instante la captura realizada por la cámara.

Normalmente durante la captura se le debe dar un tiempo de captura para que la cámara ajuste los parámetros de luz y foco automáticamente, en este sistema no se le da ese tiempo por lo que hay que mantener la zona de la toma con buenas condiciones de luz para una captura en el menor tiempo posible con la mejor calidad.
4.5.4 Red y base de datos Mysql

Se utilizó la siguiente red para efectuar las pruebas:

![Red de prueba](image)

Fig. 38: Red de prueba.

IP ROUTER: 192.168.1.1
IP RASPBERRY: 192.168.1.102
IP PC: 192.168.1.101

Se implementó un servidor MySQL corriendo en puerto 3306 con las siguientes tablas:

<table>
<thead>
<tr>
<th>#</th>
<th>Nombre</th>
<th>Tipo</th>
<th>Atributos</th>
<th>Nulo</th>
<th>Predeterminado</th>
<th>Comentarios</th>
<th>Extra</th>
<th>Acción</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>numero</td>
<td>int(8)</td>
<td>UNSIGNED</td>
<td>Sí</td>
<td>Ninguna</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>prod</td>
<td>int(4)</td>
<td>UNSIGNED</td>
<td>Sí</td>
<td>Ninguna</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>cant</td>
<td>varchar(10)</td>
<td>latin1_swedish_ci</td>
<td>Sí</td>
<td>Ninguna</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>destino</td>
<td>int(3)</td>
<td>Sí</td>
<td>Ninguna</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>fech</td>
<td>varchar(20)</td>
<td>latin1_swedish_ci</td>
<td>Sí</td>
<td>Ninguna</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>estado</td>
<td>int(3)</td>
<td>No</td>
<td>Ninguna</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Fig. 39: Tabla de notas de venta.

Esta es la base de datos más relevante, dado a que se está simulando la actividad de una empresa de encomiendas, los clientes mayoritariamente son ocasionales por lo que no se considera un registro de clientes excepto aquellos que solicitan factura, pero aquello corresponde a otro sistema, al contable.

El campo “numero” se refiere al número de la orden de transporte y es de tipo entero de largo 8.
El siguiente campo es el campo “prod”, de tipo entero de largo 4, que es una descripción del paquete.

En el campo “cant”, de tipo varchar de largo 10, se registra la cantidad de paquetes que conforman la encomienda.

Obviamente al ser una encomienda se espera que sea transportada a alguna dirección, para eso es el campo “destino”.

La fecha de emisión de la orden de transporte es el campo “fech”, de largo 20 y tipo varchar.

En el “estado” se indica el estado de la encomienda, inicialmente es 0 que indica recibido, 1 es despachado y 2 en destino. Al momento de ser detectado el código QR correspondiente a este paquete por la selectora, le pone la marca “1” al registro, indicando que fue despachado, este campo es de tipo entero de 3 cifras.

En esta tabla se registran las comunas de Santiago. Su campo id es un número índice de la comuna, de tipo entero de 3 cifras.

El campo “nombre” se refiere al nombre de la comuna, es un varchar de 25. “Zona” es el numero asignado a la zona en que se encuentra la comuna en cuestión, de tipo entero de largo 2.
4.5.5 Relaciones entre bases de datos.

La tabla “nota (de ventas)”, perteneciente a la base de datos “despachos”, dentro de sus campos, debe tener un campo que lo relacione con el cliente, y otro campo que lo vincula con el producto vendido, para ver los camos que componen esta tabla, ver fig. 39.

4.5.6 Estadísticas Generadas de MySQL.

Existen tres estados de las encomiendas despachadas:
0 Por despachar
1 Despachado y
2 En destino

La siguiente tabla entrega las estadísticas de estos estados:
### TABLA FRECUENCIAS COMUNAS

<table>
<thead>
<tr>
<th>Comuna</th>
<th>Frecuencia</th>
<th>Frec. Rel</th>
<th>Frec. Acum</th>
<th>Frec. Rel Acr</th>
<th>Estado</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>1</td>
<td>0,10</td>
<td>1</td>
<td>0,10</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>0,10</td>
<td>2</td>
<td>0,20</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>0,10</td>
<td>3</td>
<td>0,30</td>
<td>1</td>
</tr>
<tr>
<td>16</td>
<td>1</td>
<td>0,10</td>
<td>4</td>
<td>0,40</td>
<td>0</td>
</tr>
<tr>
<td>18</td>
<td>1</td>
<td>0,10</td>
<td>5</td>
<td>0,50</td>
<td>0</td>
</tr>
<tr>
<td>21</td>
<td>1</td>
<td>0,10</td>
<td>6</td>
<td>0,60</td>
<td>1</td>
</tr>
<tr>
<td>27</td>
<td>2</td>
<td>0,20</td>
<td>8</td>
<td>0,80</td>
<td>0</td>
</tr>
<tr>
<td>29</td>
<td>1</td>
<td>0,10</td>
<td>9</td>
<td>0,90</td>
<td>0</td>
</tr>
<tr>
<td>30</td>
<td>1</td>
<td>0,10</td>
<td>10</td>
<td>1,00</td>
<td>0</td>
</tr>
</tbody>
</table>

Fig. 42: Frecuencias por campo "estado".

### TABLA DE FRECUENCIAS ESTADO

<table>
<thead>
<tr>
<th>Estado</th>
<th>Frecuencia</th>
<th>Frec. Acum</th>
<th>Frec. Rel</th>
<th>Frec. Rel Acr</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>7</td>
<td>0,7</td>
<td>7</td>
<td>0,7</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>0,2</td>
<td>9</td>
<td>0,9</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0,1</td>
<td>10</td>
<td>1</td>
</tr>
</tbody>
</table>

Fig. 43: Tabla de frecuencias campo "Comunas".
4.5.7 Pruebas y validaciones

Se realizaron 90 pruebas, alimentando la banda con cajas con códigos QR, 30 por cada zona de entrega, en forma aleatoria.

Cada caja fue numerada del 1 al 9, y se tomaron los datos de error y tipo de error que se presentaron, por cada caja. Los errores detectados fueron 2:

- Problemas al eyectar la caja de la banda, ya sea que la caja se encontró con un obstáculo al momento de ser eyectado o la palanca eyectora se trabó con la caja

- No reconocimiento del código QR, la caja llega al final de la banda, sin ser del código correspondiente a esa acción, es decir no hubo intento de accionamiento.

A continuación se presenta la tabla de pruebas y los errores detectados:

<table>
<thead>
<tr>
<th>Caja / Prueba</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Ok</td>
<td>Ok</td>
<td>Ok</td>
<td>Ok</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>Ok</td>
<td>Ok</td>
<td>Ok</td>
</tr>
<tr>
<td>2</td>
<td>Ok</td>
<td>Ok</td>
<td>Ok</td>
<td>Ok</td>
<td>Ok</td>
<td>Ok</td>
<td>Ok</td>
<td>Ok</td>
<td>Ok</td>
<td>Ok</td>
</tr>
<tr>
<td>3</td>
<td>Ok</td>
<td>Ok</td>
<td>Ok</td>
<td>Ok</td>
<td>E</td>
<td>Ok</td>
<td>R</td>
<td>Ok</td>
<td>Ok</td>
<td>Ok</td>
</tr>
<tr>
<td>4</td>
<td>Ok</td>
<td>Ok</td>
<td>Ok</td>
<td>Ok</td>
<td>Ok</td>
<td>Ok</td>
<td>Ok</td>
<td>Ok</td>
<td>Ok</td>
<td>Ok</td>
</tr>
<tr>
<td>5</td>
<td>Ok</td>
<td>Ok</td>
<td>Ok</td>
<td>Ok</td>
<td>Ok</td>
<td>Ok</td>
<td>Ok</td>
<td>Ok</td>
<td>Ok</td>
<td>Ok</td>
</tr>
<tr>
<td>6</td>
<td>Ok</td>
<td>Ok</td>
<td>Ok</td>
<td>Ok</td>
<td>R</td>
<td>Ok</td>
<td>Ok</td>
<td>Ok</td>
<td>Ok</td>
<td>Ok</td>
</tr>
<tr>
<td>7</td>
<td>Ok</td>
<td>Ok</td>
<td>Ok</td>
<td>Ok</td>
<td>Ok</td>
<td>Ok</td>
<td>Ok</td>
<td>Ok</td>
<td>Ok</td>
<td>Ok</td>
</tr>
<tr>
<td>8</td>
<td>Ok</td>
<td>Ok</td>
<td>Ok</td>
<td>E</td>
<td>R</td>
<td>Ok</td>
<td>Ok</td>
<td>Ok</td>
<td>Ok</td>
<td>Ok</td>
</tr>
<tr>
<td>9</td>
<td>Ok</td>
<td>Ok</td>
<td>Ok</td>
<td>E</td>
<td>Ok</td>
<td>Ok</td>
<td>Ok</td>
<td>Ok</td>
<td>Ok</td>
<td>Ok</td>
</tr>
</tbody>
</table>

E = Error de eyecación  R = Error de reconocimiento

Fig. 44: Tabla de resultados de las pruebas.
La tabla de frecuencias, nos muestra que las pruebas arrojaron un 90% de efectividad, bueno para ser un prototipo.

El error de eyección representa el 3% dentro del universo de las pruebas, y el error de reconocimiento un 6%.

La tabla anterior nos muestra que dentro del universo de errores, el error de reconocimiento representa el 67% de los errores detectados y el error de eyección el 33%.
5. Discusiones

La maqueta de la banda transportadora, resultó exactamente como fue imaginada, así mismo los soportes de los pistones.

A la hora de comenzar a realizar la obra electrónica comenzaron a detectarse algunos problemas que debieron resolverse sobre la marcha. Al comenzar a integrarse el hardware y software, algunos componentes de hardware no funcionaron como se había pensado o como en teoría debían hacerlo.

El próximo problema a enfrentar, fue la velocidad de procesamiento, se tiende a pensar que por ser computacional es todo muy rápido, y se podía procesar las imágenes en forma fluida como un video, pero la velocidad de procesamiento de la imagen impuso un nuevo reto.

Cronometrando el tiempo de ejecución con un programa simple, resultó que la captura de la imagen le toma al procesador 0,89s ejecutarla, y la decodificación 0,14s, ambos procesos deben ser ejecutados dentro del mismo ciclo, y le toman al programa 1,03 segundos, en ese tiempo, tenemos que la banda avanza 25,4cms.

Como no se tiene forma de saber en qué posición de la banda se encuentra la imagen, lo más lógico es tomar imágenes consecutivas procesarlas y si hay un código en ellas proceder con lo que corresponde, de lo contrario continuar con el proceso. Entonces, se tiene que el programa se divide entre tomar la captura cuya ejecución tarda 0,89s y el resto del programa que se ejecuta en 0,14016s.

Como con el PWM no era posible controlar correctamente la banda, debido a que un ancho de pulso muy bajo dejaba sin fuerza al motor haciéndolo detenerse, se
hacía imposible el trabajo con las imágenes. Para bajar la velocidad sin quitarle potencia al motor, se optó por un PWM con un ciclo de trabajo de 80% y sincronizar el ciclo de la toma (0.89s) versus el resto del programa (0.14016s), deteniendo el motor durante la captura y poniéndolo en marcha el resto del ciclo del programa. Esto significaba un pulso de 0,14016 segundos de marcha lo que dio una velocidad de banda de 27,552mm/s, casi la mitad de la longitud de los códigos usados en las pruebas.

Resuelto de momento el problema de la banda, la siguiente tarea fue determinar cómo realizar las tomas de los códigos en un punto específico. La toma debe ser hecha mientras el código se encuentra dentro del rango de captura de la cámara.

Con la velocidad y potencia del motor obtenida conectado al puente h y la sincronización de pulsos dentro del ciclo de programa, a pesar de su andar irregular (debido al carácter artesanal de fabricación de la banda), era posible realizar la lectura de códigos, pero al no estar sincronizado el avance de la banda y la toma de imágenes, la tarea se hacía compleja dado que en ocasiones la toma no coincidía con la existencia de código dentro del rango de la captura y otras veces coincidían dos códigos en la misma toma o solo la fracción de uno.

La velocidad de la banda debía ser la suficiente para que no quedase ningún código sin fotografiar, si en algún momento alguno se fotografiaba dos veces, el problema se resuelve condicionando en el programa el reconocimiento de solo una de las tomas por cada código, con lo cual se obtuvo un programa funcional con el que se podía hacer muestreo, aunque aun así en ocasiones se obtenía una toma doble o ninguna toma, por lo tanto, el programa y la maquina podían ser mejorados.

Como fue mencionado anteriormente, la caída de tensión en el circuito del puente
h, restaba mucha fuerza al motor, entonces, por qué no hacer un PWM con un relé. Si bien es cierto el relé no puede generar frecuencias muy altas con un relé, no existe restricción que impida realizar un PWM con una frecuencia de 20 o menos Hertz, esta limitación en la velocidad de conmutación se debe a que hay que excitar las bobinas lo que produce un retardo en la respuesta del relé.

Resultó que un PWM con un ciclo de 10hz a través de relé fue óptimo. Con esto se resolvió el problema del control de la banda, bajando su velocidad a 25,07 mm/s, desde los 27,552mm/s conseguidos con el puente h y además con el aumento de la potencia, se regularizó bastante el andar de la banda.

Mejorado el funcionamiento del programa y de la máquina, cabía seguir cuestionándose, como mejorar aun el sistema.

El tomar una fotografía por ciclo aunque no hubiese código dentro de su campo de visión es muy ineficiente, por lo que el próximo paso fue eliminar esta “ineficiencia”.

Cuando se toma una fotografía se enfoca al sujeto que se desea fotografiar, una vez en el campo visual de la cámara, se presiona el obturador, y se tiene la fotografía con encuadre del sujeto deseado. Entonces, por qué no ponerle un obturador a la máquina.

Así surgió la idea de un sensor que nos indicara cuando nuestro sujeto está encuadrado dentro del campo visual de la cámara para fotografiarlo. Las posibilidades barajadas fueron, sensor capacitivo o infrarrojo.

El sensor capacitivo, da el rango requerido de aproximadamente 50mm de campo de acción, y el infrarrojo 450mm. Por lo tanto la decisión fue tomada en favor del
infrarrojo dado su mayor campo de acción en caso de necesitar algún ajuste especial a la máquina.

Con un obturador se obtuvo mayor eficiencia del programa, debido al ahorro de tiempo de procesamiento, se pudo incrementar la velocidad del sistema en su totalidad.

Por otro lado surgió el problema de hacer el seguimiento del paquete por la banda una vez detectado el código, para poder sincronizar su avance y poder realizar su eyección de manera correcta. Para ello se creó un sistema de posición de paquetes en el tiempo sobre la banda.

Al detectar un código se marca su posición y según la cantidad de ciclos de ejecución del programa (a modo de medir el tiempo), se va moviendo la marca de su posición, y cuando esta coincide con la posición del eyector correspondiente, este actúa.
6. Conclusiones

Al introducir nuestro tema, se habló de la necesidad de acortar los tiempos de inventario, obviamente toda automatización tiene como parte de sus objetivos mejorar los tiempos de realización de las actividades, y al encontrar durante el desarrollo del proyecto con que la toma de imágenes del sistema y posterior decodificación tardaba más de un segundo (1,03s), parecía todo desmoronarse y terminar concluyendo que este proyecto era inviable por la incapacidad de aumentar su velocidad.

En efecto, si los resultados se analizan desde la perspectiva de tareas con objetos pequeños a clasificar, resulta mucho más rápido y eficiente la realización por mano humana, pero, si analizamos los resultados desde la perspectiva de manipular paquetes grandes, en ocasiones difíciles de manejar ya sea por su tamaño o peso, en que la rapidez se transforma en un obstáculo o en un inconveniente, debido entre otras cosas a las dificultades técnicas que implica el mover materiales grandes y pesados por la cantidad de fuerza que implica mantener sus velocidad, que los materiales de regular tamaño o masa moviéndose a velocidades considerables representan un peligro para las personas que trabajan en torno a ellos, como también el cansancio que pueden producir al tratarlos manualmente, resulta que una velocidad de casi un empaque por segundo resulta bastante conveniente, ya sea para la clasificación de encomiendas hacia diferentes zonas de carga o por qué no, la implementación de este sistema en terminales de buses, trenes y hasta en las bandas transportadoras de equipajes en aeropuertos. “El Aeropuerto de Santiago dispone de un sistema bastante avanzado pero aún parte de sus operaciones se realizan de manualmente, lo que produce una menor calidad de servicio y problemas en la gestión del aeropuerto” (Cavada et al, 2012).
No obstante lo anteriormente expuesto, cambiando algunos componentes de esta máquina es posible realizar otras labores de clasificado con objetos pequeños a altas velocidades, como podría ser cartas, cajas de medicamentos, etc. La limitante en este experimento fue el dispositivo lector elegido, la cámara presenta ese inconveniente, que al no ser un dispositivo especializado en códigos QR, precisa de mayor tiempo para el procesamiento de imágenes. Actualmente los lectores de códigos (de barra o QR), poseen una velocidad de lectura de alrededor de 2000 códigos por segundo.

Definitivamente el proyecto propuesto es viable, resultando un prototipo aun mejorable en muchos aspectos, pero funcional, escalable y de bajo costo. Posible de implementar en muchas áreas de la economía, particularmente en ingreso y egreso de productos de gran volumen a stock, esto es respaldado por los resultados de las pruebas que arrojaron un 10% de error, cuando la diferencia entre el empaque de mayor y menor largo utilizados en las pruebas es de un 20% (88mm el más corto y 110mm el más largo).
7. Bibliografía


8. Anexos
Estadísticos.html

<html>
<head>
<title>Control pedidos</title>
<link href="../estilo.css" type="text/css" rel="stylesheet">
</head>
<body>
<table width="100%" border="0">
<tr>
<td colspan="5" align="center"><h4> TABLA DE FRECUENCIAS ESTADO</h4></td>
</tr>
<tr>
<td width="5%" bgcolor="#FF9900">Estado</td>
<td width="5%" bgcolor="#FF9900">Frecuencia</td>
<td width="5%" bgcolor="#FF9900">Frec.Acum.</td>
<td width="5%" bgcolor="#FF9900">Frec.Rel.</td>
<td width="5%" bgcolor="#FF9900">Frec.Rel. Acum.</td>
</tr>
</body>
</html>

<html>
<head>
<title>Control pedidos</title>
<link href="../estilo.css" type="text/css" rel="stylesheet">
</head>
<body>
<table width="100%" border="0">
<tr>
<td colspan="5" align="center"><h4> TABLA FRECUENCIAS COMUNAS</h4></td>
</tr>
<tr>
<td width="5%" bgcolor="#FF9900">Comuna</td>
<td width="5%" bgcolor="#FF9900">Frecuencia</td>
<td width="5%" bgcolor="#FF9900">Frec. Rel.</td>
<td width="5%" bgcolor="#FF9900">Frec. Acum.</td>
<td width="5%" bgcolor="#FF9900">Frec. Rel. Ac.</td>
<td width="5%" bgcolor="#FF9900">Estado</td>
</tr>
</body>
</html>
Estadisticos.php

<?php
include ('conexion2.php'); ?>
<html>
<head>
<title>Control pedidos</title>
<link href="../estilo.css" type="text/css" rel="stylesheet">
<meta http-equiv="refresh" content="20"/>
</head>
<body>
<table width="100%" border="0">
<?php
$sql="SELECT * FROM `notas` order by estado";
$res=mysqli_query($mysqli,$sql);
if (!$res){ echo "no hay ",mysqli_error($mysqli);}
else {
$c=(mysqli_num_rows($res));
$f=0;
$fa=0;
//echo ($c);
$d_ant="";
// echo $c;
while($filas=mysqli_fetch_array($res)){
    $num=$filas['numero'];
    $prod=$filas['prod'];
    $cant=$filas['cant'];
    $dest=$filas['destino'];
    $fecha=$filas['fecha'];
    $estado=$filas['estado'];
    if ($d_ant == "") {$d_ant=$estado;}
    if ($d_ant != $estado) { $fa=$fa+$f;
        <tr>
            <td width="5%" bgcolor="#FFFADD" ALIGN="RIGHT">$d_ant</td>
            <td width="5%" bgcolor="#FFFADD" ALIGN="RIGHT">$f</td>
            <td width="5%" bgcolor="#FFFADD" ALIGN="RIGHT">$f/$c</td>
            <td width="5%" bgcolor="#FFFADD" ALIGN="RIGHT">$fa</td>
            <td width="5%" bgcolor="#FFFADD" ALIGN="RIGHT">$fa / $c</td>
        </tr>
        $f=1;
        $d_ant=$estado;
    } else {
        $f = $f+1;
    }
}$c ?>
</table>
</body>
</html>
<?php }
</table>
</body>
</html>

<p align="center"> <a href="index1.html">Volver</a></p>

</body>
</html>

65
import io
import os
import glob
import time
import picamera
from PIL import Image
import zbar
import RPi.GPIO as gpio
import MySQLdb
nombre=""'
zona=""
duty=70

####[VARIABLES GLOBALES
	tiempo=5 #TIEMPO DE CAPTURA EN SEGUNDOS
	pl=Image.new('RGB', (60, 30))
####[FUNCIONES
def lado_a():
    p.ChangeDutyCycle(100)
    gpio.output(6, 0) ## Hace el GPIO 6 de salida
    gpio.output(5, 1) ## Hace el GPIO 5 de salida con 5VCC (HIGH)
    time.sleep(.20)
    gpio.output(6, 1) ## Hace el GPIO 6 de salida con 5VCC (HIGH)
    gpio.output(5, 0) ## Hace el GPIO 5 de salida con 0VCC (LOW)
    time.sleep(.20)
    gpio.output(6, 0) ## Hace el GPIO 6 de salida con 0VCC (LOW)
    p.ChangeDutyCycle(duty)
    time.sleep(.40)
    #a[9]="0"

def lado_b():
    p.ChangeDutyCycle(100)
    gpio.output(19, 0) ## Hace el GPIO 19 de salida con 0VCC (LOW)
    gpio.output(13, 1) ## Hace el GPIO 17 de salida con 5VCC (HIGH)
    time.sleep(.2)
    gpio.output(19, 1) ## Hace el GPIO 17 de salida con 5VCC (HIGH)
    gpio.output(13, 0) ## Hace el GPIO 17 de salida con 5VCC (LOW)
    time.sleep(.20)
    gpio.output(19, 0) ## Hace el GPIO 17 de salida con 0VCC (LOW)
    p.ChangeDutyCycle(duty)
    time.sleep(.40)

def InicializaGPIO():
    gpio.setwarnings(False)
    gpio.setmode(gpio.BCM)
    gpio.setup(5, gpio.OUT)
    gpio.output(5, gpio.OUT)
    gpio.setup(6, gpio.OUT)
    gpio.output(6, gpio.OUT)
    
    def lado_a():
        p.ChangeDutyCycle(100)
        gpio.output(6, 0) ## Hace el GPIO 6 de salida
        gpio.output(5, 1) ## Hace el GPIO 5 de salida con 5VCC (HIGH)
        time.sleep(.20)
        gpio.output(6, 1) ## Hace el GPIO 6 de salida con 5VCC (HIGH)
        gpio.output(5, 0) ## Hace el GPIO 5 de salida con 0VCC (LOW)
        time.sleep(.20)
        gpio.output(6, 0) ## Hace el GPIO 6 de salida con 0VCC (LOW)
        p.ChangeDutyCycle(duty)
        time.sleep(.40)
        #a[9]="0"

    def lado_b():
        p.ChangeDutyCycle(100)
        gpio.output(19, 0) ## Hace el GPIO 19 de salida con 0VCC (LOW)
        gpio.output(13, 1) ## Hace el GPIO 17 de salida con 5VCC (HIGH)
        time.sleep(.2)
        gpio.output(19, 1) ## Hace el GPIO 17 de salida con 5VCC (HIGH)
        gpio.output(13, 0) ## Hace el GPIO 17 de salida con 5VCC (LOW)
        time.sleep(.20)
        gpio.output(19, 0) ## Hace el GPIO 17 de salida con 0VCC (LOW)
        p.ChangeDutyCycle(duty)
        time.sleep(.40)

    def InicializaGPIO():
        gpio.setwarnings(False)
        gpio.setmode(gpio.BCM)
        gpio.setup(5, gpio.OUT)
        gpio.output(5, gpio.OUT)
        gpio.setup(6, gpio.OUT)
        gpio.output(6, gpio.OUT)

76

66
import pymysql

gpio.setup(13, gpio.OUT)
gpio.output(13, gpio.OUT)
gpio.setup(19, gpio.OUT)
gpio.output(19, gpio.OUT)
gpio.setup(24, gpio.OUT)
gpio.setup(23, gpio.OUT)  # iluminacion
gpio.output(23, gpio.OUT)
gpio.setup(4, gpio.IN)

def Captura():
    #timestart=time.time()
    gpio.output(23, 0)
    with picamera.PiCamera() as picam:
        picam.resolution = (300, 237)
        picam.color_effects = (128,128)
        picam.start_preview()
        time.sleep(0)
        picam.capture('/home/pi/Pictures/foto.jpg')
        picam.stop_preview()
        picam.close()
        gpio.output(23, 1)
    #print('--- %s seconds cap--- % (time.time() - timestart))

def Decodifica():
    #timestart=time.time()
    scanner = zbar.ImageScanner()
    scanner.parse_config('enable')
    pil = Image.open('/home/pi/Pictures/foto.jpg')
    pil=pil.convert('L')
    width, height = pil.size
    raw = pil.tobytes()
    image = zbar.Image(width, height, 'Y800', raw)
    scanner.scan(image)
    data = ''
    global nombre, zona
    for symbol in image:
        data += symbol.data
    nombre=data[:-2]
    zona=data[-1]
    del(image)
    #print('--- %s seconds ---" % (time.time() - timestart))
    #print(nombre, zona)

def Conectar():
    p.ChangeDutyCycle(100)
    db =
    pymysql.connect(host="localhost",user="usuario",passwd="password",db="despacho"
    )
    cur = db.cursor()
    #query=('"""SELECT * FROM `comunas` WHERE `Nombre`= ?"""", {nombre})
    global nombre
    cur.execute('"""SELECT * FROM `notas` where zon = %d"""" %int(zona))
    #mysql_query($conexion, "SELECT * FROM `comunas` WHERE
    `Nombre`=".$nombre
    rows = cur.fetchone()
cant=int(rows[2])
cant=cant-1
if 0>cant:
    cant=0

cur.execute("UPDATE notas SET estado = 1, cant=%s WHERE fech=%s" % (cant, rows[5]))
#WHERE numero=%d and prod=%d and destino=%d and fech=%s and zon=%d"
#(cant, rows[0], rows[1], rows[3], rows[4], rows[5])
print(rows[0], rows[1], rows[2], rows[3], rows[4], rows[5])
#db.commit()
p.ChangeDutyCycle(duty)

#MAIN
InicializaGPIO()
#for filename in glob.glob(os.path.join(path, '*.png')):
    #timestart=time.time()
    #1 2 3
    #12345678901234567890123456789012345678901234567890
    a="00000000000000000000000000000"
    b="00000000000000000000000000000"
p = gpio.PWM(24, 10)  # Pin=24 frecuencia=10Hz
p.start(duty)
ob=0
c=0
try:
    while 1:
        a2="0"
        b2="0"

        #timestart=time.time()
        if gpio.input(4)==0:
            global nombre
            p.ChangeDutyCycle(100)
            Captura()
            p.ChangeDutyCycle(duty)
            Decodifica()
            print(nombre, zona)
            if zona !="":
                Conectar()
            ob=1
            if c==0:
                c=1
            else:
                ob=0
        if zona=="1":
            if a[0]=="0":
                a2="1"
        elif zona=="2":
            if b[0]=="0":
                b2="1"
if (zona!="") or (c==5487):
    a=a2+a
    b=b2+b
    a=a[:18]
    b=b[:13]

c=c+1
if c>5487:
    c=1

nombre=""
zona=""
if a[15]=="1":
    a=a[:15]+"00000000"
    lado_a()
if b[9]=="1":
    b=b[:9]+"000000"
    lado_b()
if nombre!="":
    print("nada")
nombre=""
zona=""
print (a)
print (b)
# print("--- %s seconds ---" % (time.time() - timestart))
# break
except KeyboardInterrupt:
    pass
p.stop()
gpio.output(23, 1)
gpio.cleanup()
Fotos Prototipos
Ejemplo de codificación QR.

Para mostrar el procedimiento, se codificará: “U. ANDRES BELLO”.

Se selecciona el modo de la tabla de modos de codificación, en este caso 0010, modo alfanumérico

Determinamos el largo de nuestra cadena de caracteres: 15 (1111 en binario) caracteres.

Una vez cuantificada la información que queremos codificar, seleccionamos la versión de QR apropiada

Se agrega el indicador de largo que representa la cantidad de caracteres de la información en binario y el largo de la cadena de bits que lo representa, este indicador depende de la versión QR a utilizar, a continuación la tabla de largo de cadena de bits de acuerdo a la versión QR. Ver tabla 3.

Ahora se separa la cadena de caracteres seleccionada en pares y se le asigna su número de carácter según la tabla de caracteres alfanuméricos presentada a continuación (Ver tabla 4)

El primer número del par se multiplica por 45 y se le suma el valor de su par. El resultado se convierte en binario de 11 bits. Según se muestra abajo:

<table>
<thead>
<tr>
<th>Letra</th>
<th>45x Valor</th>
<th>Valor</th>
<th>Binario</th>
</tr>
</thead>
<tbody>
<tr>
<td>U.</td>
<td>30 x 45 +</td>
<td>42</td>
<td>1392</td>
</tr>
<tr>
<td>A.</td>
<td>36 x 45 +</td>
<td>10</td>
<td>1630</td>
</tr>
<tr>
<td>ND</td>
<td>23 x 45 +</td>
<td>13</td>
<td>1048</td>
</tr>
<tr>
<td>RE</td>
<td>27 x 45 +</td>
<td>14</td>
<td>1229</td>
</tr>
</tbody>
</table>
El caracter que queda aislado mantiene su valor y se transforma a binario de 6 bits.

Entonces, seleccionamos el modo alfanumérico de la versión 1 y NCE (nivel de corrección de errores) Q, el cual puede contener 16 caracteres. Se puede seleccionar cualquier versión en que el modo alfanumérico y NCE tenga una capacidad igual o mayor a 15 caracteres.

<table>
<thead>
<tr>
<th>Versión</th>
<th>Nivel de corrección de errores</th>
<th>Modo Numérico</th>
<th>Modo Alfanumérico</th>
<th>Modo Byte</th>
<th>Modo Kanji</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>L</td>
<td>41</td>
<td>25</td>
<td>17</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>M</td>
<td>34</td>
<td>20</td>
<td>14</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>Q</td>
<td>27</td>
<td>15</td>
<td>11</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>H</td>
<td>17</td>
<td>10</td>
<td>7</td>
<td>4</td>
</tr>
</tbody>
</table>

Figura 1.1
Fig. 47: Versión seleccionada.

Con la cadena codificada, se concatenan:

Indicador de modo,

<table>
<thead>
<tr>
<th>Modo</th>
<th>Indicador</th>
<th>Contenido</th>
</tr>
</thead>
<tbody>
<tr>
<td>Numérico</td>
<td>0001</td>
<td>Dígitos del 0 – 9</td>
</tr>
<tr>
<td>Alfanumérico</td>
<td>0010</td>
<td>Dígitos y las letras mayúsculas, y los símbolos %, *, +, - y /</td>
</tr>
</tbody>
</table>
Indicador de cantidad de caracteres en formato de 9 bits (000001111),

y un terminador de cadena “0000”, la cadena codificada queda de la siguiente manera:

0010 000001111 10101110000 11001011110 10000011000 10011001101 10100010000 001111110 1111000110 011000 0000

La cadena concatenada no debe tener espacios, aquí los espacios se muestran para delimitar cada componente de la concatenación.

Ahora la cadena tiene un largo de 100 caracteres, pero el número de caracteres de la cadena debe ser múltiplo de 8 por lo que se debe en este caso agregar ceros al final para llevarlo al largo del próximo múltiplo de 8 que es 104.

0010 000001111 10101110000 11001011110 10000011000 10011001101 10100010000 001111110 011110000 0000 0000.

El estándar de códigos QR exige que la cadena de bits llene completamente la capacidad de éste, es posible que se deba agregar bytes al final para completar.

Según la tabla de corrección de errores, para el ejemplo tomado (Versión 1-Q), su capacidad es de 13 palabras de código (13x8=104 bits), por lo tanto no se debe agregar nada.

Si la cadena no es suficientemente larga, se debe agregar las siguientes palabras hasta completar la capacidad del código: 11101100 00010001.

Tenemos las 13 palabras de 8 bits, resultantes de la división del registro
correspondiente a nuestra información codificada en bits.

00100000 = 32
01111101 = 125
01110000 = 112
11001011 = 203
11010000 = 208
01100010 = 98
01100110 = 102
11010001 = 209
00000011 = 3
11111010 = 250
11110001 = 241
10011000 = 152
0000 0000 = 0

Nuestro “Polinomio Mensaje” sería:

\[32x^{12} + 125x^{11} + 112x^{10} + 203x^9 + 208x^8 + 98x^7 + 102x^6 + 209x^5 + 3x^4 + 250x^3 + 241x^2 + 152x + 0\]

Y el “Polinomio Generador”:

\[x^{13} + \alpha^{24}x^{12} + \alpha^{37}x^{11} + \alpha^{181}x^{10} + \alpha^{87}x^9 + \alpha^{80}x^8 + \alpha^{106}x^7 + \alpha^{104}x^6 + \alpha^{130}x^5 + \alpha^{218}x^4 + \alpha^{206}x^3 + \alpha^{140}x^2 + \alpha^{78}x + \alpha^8\]

Para evitar que el “Polinomio Mensaje” (desde ahora PM), se acorte demasiado pronto y porque en este caso es menor que el “Polinomio Generador” (de aquí en adelante PG), lo amplificamos, dando: por \(x^{13}\) dando:

\[32x^{25} + 125x^{24} + 112x^{23} + 203x^{22} + 208x^{21} + 98x^{20} + 102x^{19} + 209x^{18} + 3x^{17} + 250x^{16} + 241x^{15} + 152x^{14} + 0x^{13}\]

Como el PG tiene que ser del mismo grado que PM, se le amplifica por \(\alpha^{5}x^{12}\)
\[ \alpha^5 x^{25} + \alpha^7 x^{24} + \alpha^{15} x^{23} + \alpha^{17} x^{22} + \alpha^{100} x^{21} + \alpha^{85} x^{20} + \alpha^{106} x^{19} + \alpha^{104} x^{18} + \alpha^{218} x^{15} + \alpha^{206} x^{14} + \alpha^{140} x^{13} + \alpha^{78} \]

Ahora se transforma los \( \alpha \) a notación entera:

\[ 32x^{25} + 240x^{24} + 213x^{22} + 49x^{21} + 26x^{20} + 163x^{19} + 23x^{18} + 206x^{17} + 189x^{16} + 169x^{15} + 9x^{14} + 178x^{13} + 77x^{12} + 18x^{11} \]

Al efectuar la división:

\[ 32x^{12} + 125x^{11} + 112x^{10} + 203x^9 + 208x^8 + 98x^7 + 102x^6 + 209x^5 + 3x^4 + 250x^3 + 241x^2 + 152x^1 + 0x^0 \]

\[ 32x^{12} + 240x^{24} + 213x^{22} + 49x^{21} + 26x^{20} + 163x^{19} + 23x^{18} + 206x^{17} + 189x^{16} + 169x^{15} + 9x^{14} + 178x^{13} + 77x^{12} + 18x^{11} \]

El resultado es la operación XOR de los factores de \( x \).

<table>
<thead>
<tr>
<th>X ( \times 18 )</th>
<th>20</th>
<th>21</th>
<th>22</th>
<th>23</th>
<th>24</th>
<th>25</th>
<th>26</th>
<th>27</th>
<th>28</th>
<th>29</th>
<th>30</th>
<th>31</th>
<th>32</th>
<th>33</th>
<th>34</th>
<th>35</th>
<th>36</th>
<th>37</th>
</tr>
</thead>
<tbody>
<tr>
<td>32</td>
<td>125112</td>
<td>201</td>
<td>20916</td>
<td>102</td>
<td>20916</td>
<td>250</td>
<td>15210</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>240203</td>
<td>496</td>
<td>16323</td>
<td>206189</td>
<td>1099</td>
<td>17877</td>
<td>187</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>141</td>
<td>165202</td>
<td>103124</td>
<td>311509</td>
<td>83248827</td>
<td>77187</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>141</td>
<td>176113</td>
<td>195181</td>
<td>2418170</td>
<td>15910123082</td>
<td>211180</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>233</td>
<td>13913</td>
<td>116100</td>
<td>17024820415720431104180</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>233</td>
<td>9242</td>
<td>65133350</td>
<td>233217113581</td>
<td>200488</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>212</td>
<td>9875</td>
<td>141180</td>
<td>17212624778</td>
<td>16013488</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>212</td>
<td>12425</td>
<td>15521</td>
<td>20421</td>
<td>417771791613149147</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>91</td>
<td>8222</td>
<td>161231</td>
<td>0</td>
<td>197</td>
<td>18625317625117147</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>91</td>
<td>63250</td>
<td>135130</td>
<td>171130</td>
<td>7</td>
<td>200138260119247215</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>109</td>
<td>22638</td>
<td>55171</td>
<td>71129</td>
<td>53</td>
<td>1852</td>
<td>100215</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>109</td>
<td>10696</td>
<td>8024386</td>
<td>24322239292171945178</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>74</td>
<td>7015</td>
<td>8817</td>
<td>7835</td>
<td>17719421915250176</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>74</td>
<td>206874</td>
<td>2421581971588421212911252191324</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>130</td>
<td>00170</td>
<td>143129195229277205141251224</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>136</td>
<td>2191</td>
<td>1432248722416624898012549</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>131</td>
<td>1710</td>
<td>1072345393</td>
<td>253</td>
<td>23057</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>122</td>
<td>98168</td>
<td>22591</td>
<td>12091</td>
<td>67217239877 ( \times 144 )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>200</td>
<td>268151</td>
<td>177125</td>
<td>72643610</td>
<td>906</td>
<td>13794</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>200</td>
<td>4135</td>
<td>54248175246757322124713124</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>129</td>
<td>179135</td>
<td>13959</td>
<td>182111209128356</td>
<td>172170</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>129</td>
<td>180583912171214319517615161173174</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>221</td>
<td>18517266</td>
<td>17722</td>
<td>250671479137</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>221</td>
<td>20154</td>
<td>51140</td>
<td>156140160404012012422512</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>116</td>
<td>142313</td>
<td>611381182271249115117912</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>116</td>
<td>107391</td>
<td>11010521710324663216119545254</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22929068</td>
<td>287175132671992477214088254</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Para seleccionar la máscara de acuerdo al protocolo QR, el código obtenido hasta aquí, es evaluado aplicándole las 8 mascaras ya mencionadas anteriormente (numeradas del 0 al 7).

Una vez aplicadas las máscaras se aplican 4 criterios de evaluación, cada uno asigna un puntaje al código enmascarado. La máscara que sume el puntaje más bajo de los cuatro criterios, será la máscara a utilizar.

La evaluación se aplica también a las zonas reservadas y de función.

Para mostrar el procedimiento se utilizará la máscara 3 que es la que finalmente se seleccionó para el código de ejemplo (ver Fig. 16).

El primer criterio es buscar filas de 5 celdas del mismo color consecutivas. Por cada patrón encontrado se suman 3 puntos, y por cada celda del mismo color extra se suma 1 punto.

Se repite el mismo procedimiento con las columnas.

![Fig. 50: Resultado aplicación criterio n°1 horizontal. Total Horizontal 111](image)
El próximo criterio de evaluación, es buscar bloques del mismo color conformados por dos columnas y dos líneas (2x2), estos bloques pueden estar superpuestos.
Total evaluación 2x2 = 180

El tercer criterio consiste en buscar el patrón “1011010000” y “00001011101”, vertical y horizontalmente, cada vez que se encuentra se suman 40 puntos.

Fig. 53: Resultado aplicación criterio n°3, bloques 2x2.

Total de este criterio: 80 puntos

Para el cuarto criterio se debe contar las celdas negras y calcular su porcentaje respecto del total de celdas del código.

Total celdas negras 206.

Total celdas código 21x21 = 441

Entonces 206/441x100 = 46,7

Luego se encuentra el próximo múltiplo entero de 5 hacia arriba y hacia abajo del porcentaje. Estos son 45 y 50.

Se hace la resta absoluta cada uno de estos múltiplos de 5 contra 50 y el
El resultado se multiplica por 10

\[|45 - 50| = 5 \times 10 = 50\]
\[|50 - 50| = 0 \times 10 = 0\]

Se toma el menor de estos números, en este caso cero, y será la cantidad que se sumará al puntaje para este código enmascarado.

Se suma el total de puntaje de los cuatro criterios:

\[111 + 109 + 180 + 80 + 0 = 480.\]

Se evalúan las 8 mascaras bajo estos cuatro criterios y se selecciona el código enmascarado con menor puntaje.

Se recalca que no es obligatorio realizar este complejo proceso para determinar la máscara a utilizar, se puede seleccionar arbitrariamente. Si, es obligatorio aplicar una máscara e incluirla en el código para su posterior decodificación.

En este punto cabe mencionar que la aplicación de la máscara obedece a una cuestión estética del código. Las máscaras están diseñadas para que la distribución de las celdas negras y blancas sea lo más homogénea posible.

**Codificación de la máscara.** (011101000000110)

El número de mascara se compone del binario para el nivel de corrección de errores y el binario para la máscara utilizada para la codificación.

Para el ejemplo que se ha trabajado hasta aquí:

Nivel de corrección de errores “Q” = 11
Numero de mascara = 011, unidos 11011.

El polinomio generador es: 10100110111
Primero convertimos en un número de 15 bits ambos binarios, colocando ceros a la derecha de estos números para conseguir el largo deseado. De existir se eliminan los ceros a la izquierda de ellos.

Hecho esto, se realiza la operación XOR tantas veces como sea necesario para convertirlo en un número de 10 bits.

Entonces tenemos:

110110000000000
101001101110000
011111101110000

Cada vez que se realice esta operación, se debe eliminar los ceros la izquierda e igualar el largo del polinomio generador con el resultado poniendo ceros a la derecha de este.

11111101110000
10100110111000
01011011001000
1010011011100
0001000010100

Una vez que se tiene el binario de 10 bits (1000010100), se concatena a los 5 bits de Nivel de corrección y mascara (11011), dando:

110111000010100

Este número se debe formatear, para ello se busca en la tabla de cadenas de formato.

Existe una cadena de formato por cada nivel de corrección de errores y mascara,
como son 4 niveles de corrección y 8 mascaras en total existen 32 cadenas de formato. Para el ejemplo: 011101000000110.

Nuevamente se aplica XOR.
11011000010100
011101000000110
101010000010010

Se numera cada bit, y de acuerdo a ese número se distribuyen dentro de código.

<table>
<thead>
<tr>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

*Fig. 54: Tabla de distribución de la máscara dentro del código.*

Se inserta en el código en las posiciones indicadas de acuerdo a la tabla 1.7 y la figura 1.20, quedando:

*Fig. 55: Código resultante.*