Trabajo Práctico Final

Introducción a la Robótica Empresarial

Alumno: **Santiago Reyes**
Legajo: **58148**
DNI: **40910696**
Fecha: **04/09/2023**

1. Ejercicio 1

Sonidos

Los tres sonidos se tomaron de pixabay y son:

1. Guitarra acustica[1]
2. Ruido de lluvia (recortado a 5 segundos)[2]
3. Frase diciendo: “Hi, you’re listening to the greatest radio station in the world!”[3]

Parte A - Oscilograma

1A.1) Tabla comparativa de sonidos

	# Muestras	Duración (s)	Muestreo	Mono/Estéreo
Guitarra	262656	5.96	44100	Estéreo
Lluvia	222336	5.04	44100	Estéreo
Habla	108288	4.51	24000	Estéreo

[1] https://pixabay.com/es/sound-effects/acoustic-guitar-logo-13084/
[2] https://pixabay.com/es/sound-effects/heavy-rain-nature-sounds-8186/
[3] https://pixabay.com/es/sound-effects/greatest-radio-station-104978/

1A.2) Oscilogramas de los sonidos

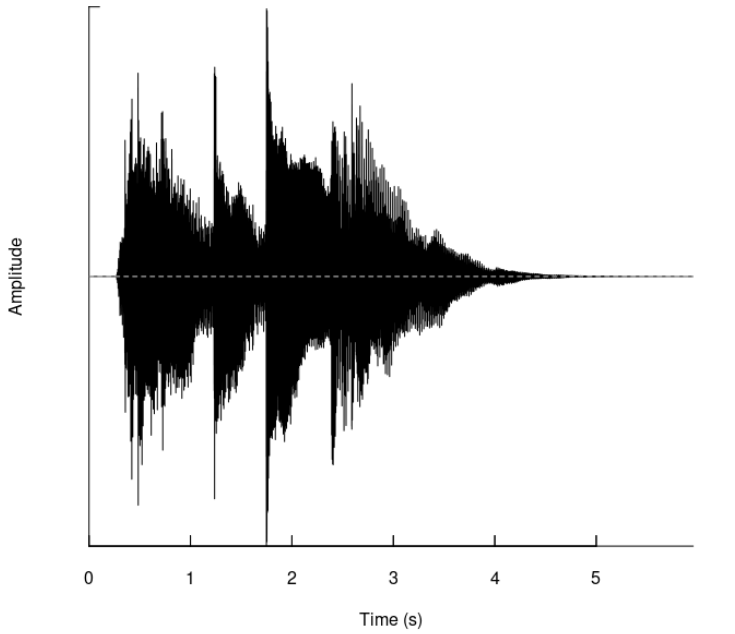


Figura 1: Oscilograma "guitar.mp3"

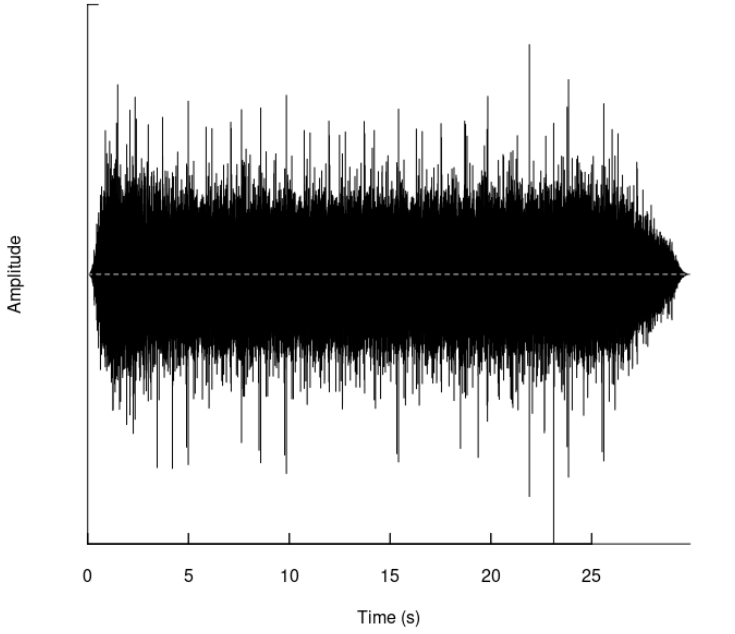


Figura 2: Oscilograma "rain_short.mp3"

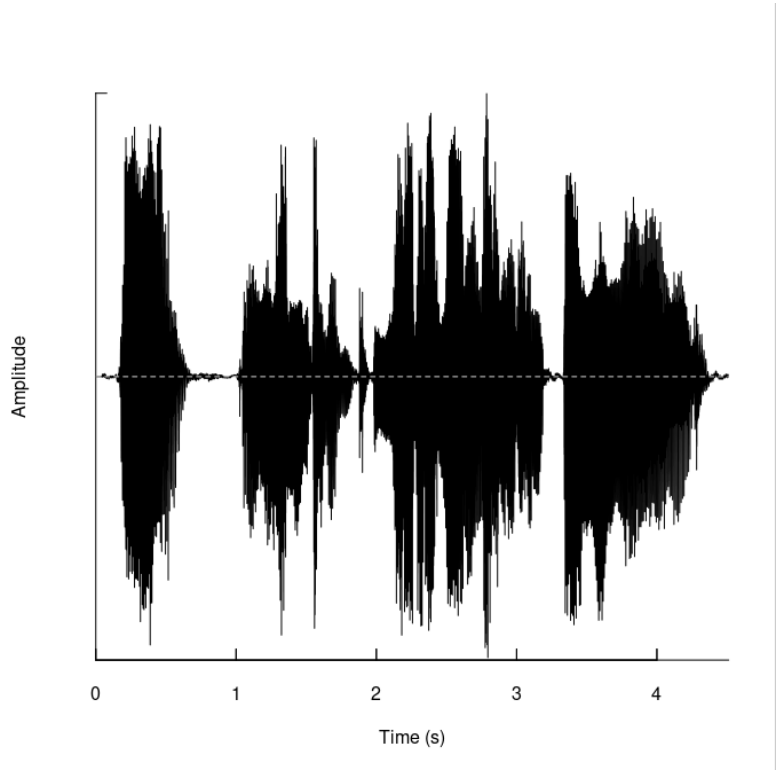


Figura 3: Oscilograma "speak.mp3"

1A.3) Oscilogramas zoomeados

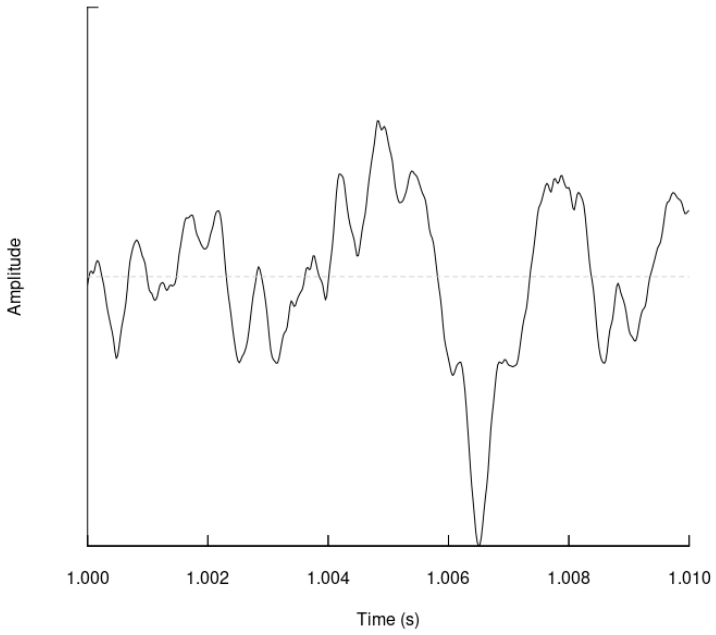


Figura 4: Oscilograma de "guitar.mp3" entre el segundo 1 y 1.01

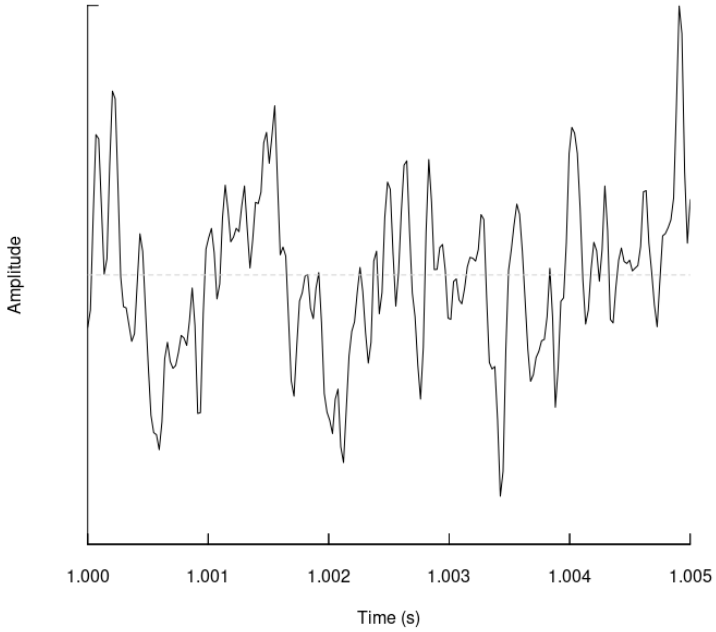


Figura 5: Oscilograma de "rain_short.mp3" entre el segundo 1 y 1.005

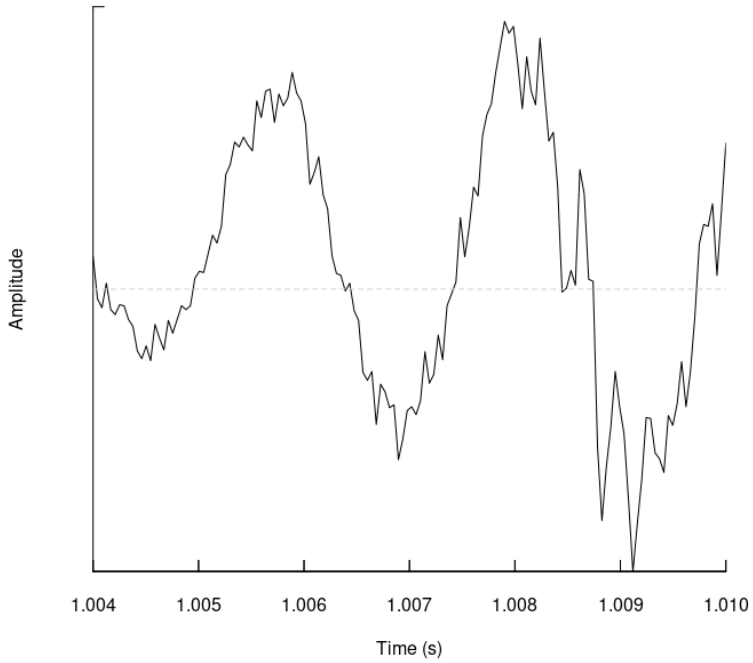


Figura 6: Oscilograma de "speak.mp3" entre el segundo 1.004 y 1.01

1A.4) Comparación de sonido normalizado

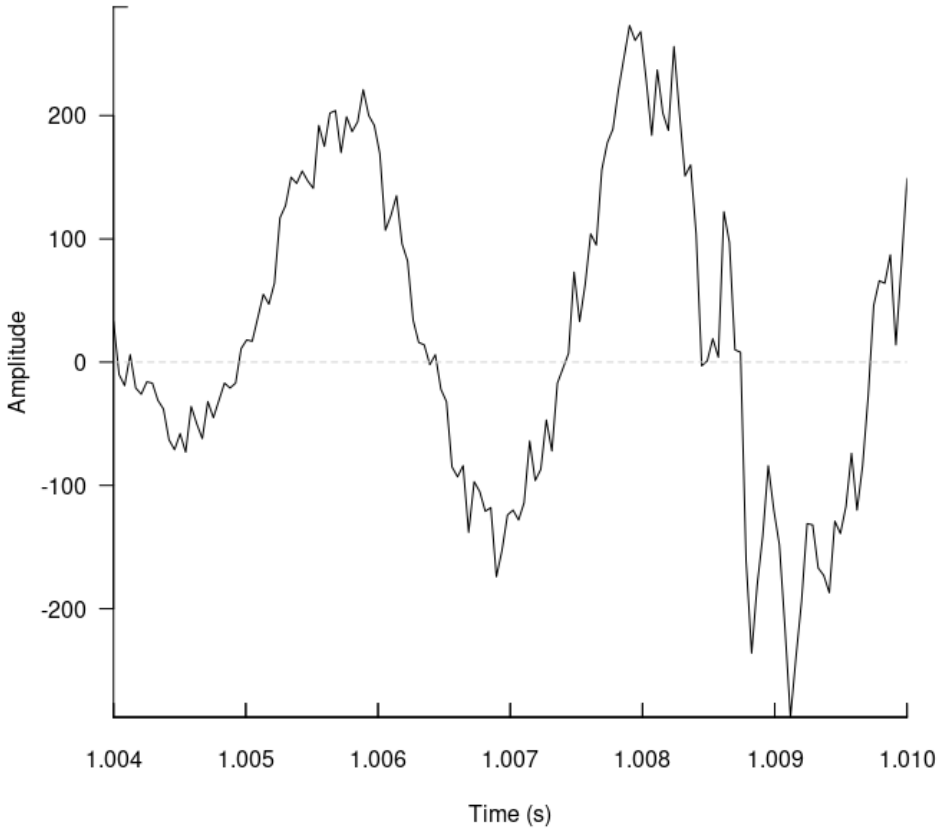


Figura 7: Oscilograma de "speak.mp3" entre el segundo 1.004 y 1.01 con label en la amplitud.

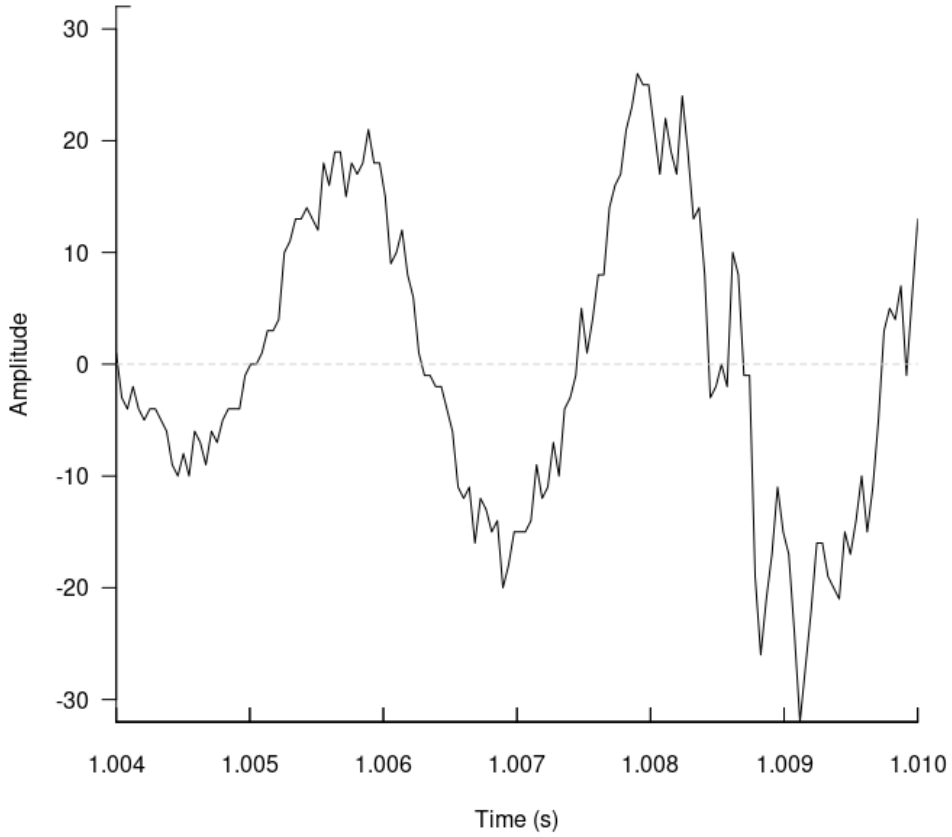


Figura 7: Oscilograma de "speak.mp3" entre el segundo 1.004 y 1.01 con label en la amplitud normalizado al nivel 0.1

Comparando la *Figura 6* y la *Figura 7* se puede apreciar que tras la normalización la onda se mantiene igual, sin embargo lo que cambió fue la amplitud (bajo el volumen).

Parte B - Periodograma-Espectrograma

1B.1) Periodogramas de los sonidos

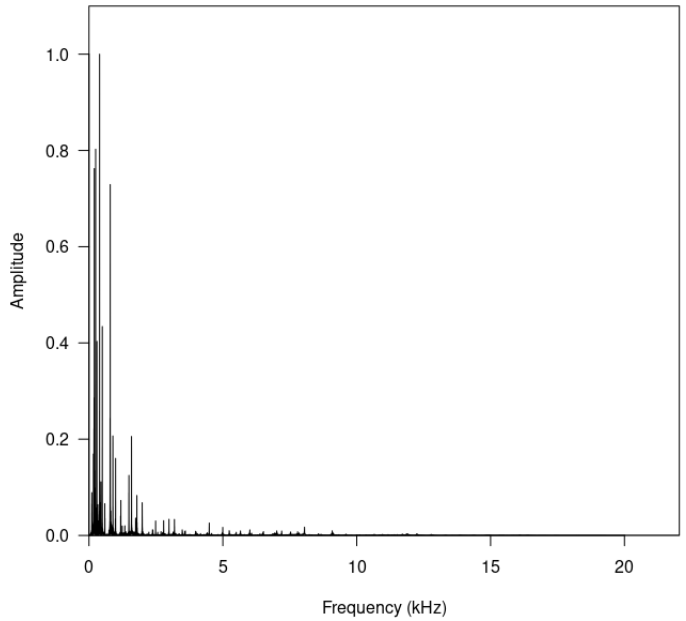


Figura 8: Periodograma "guitar.mp3"

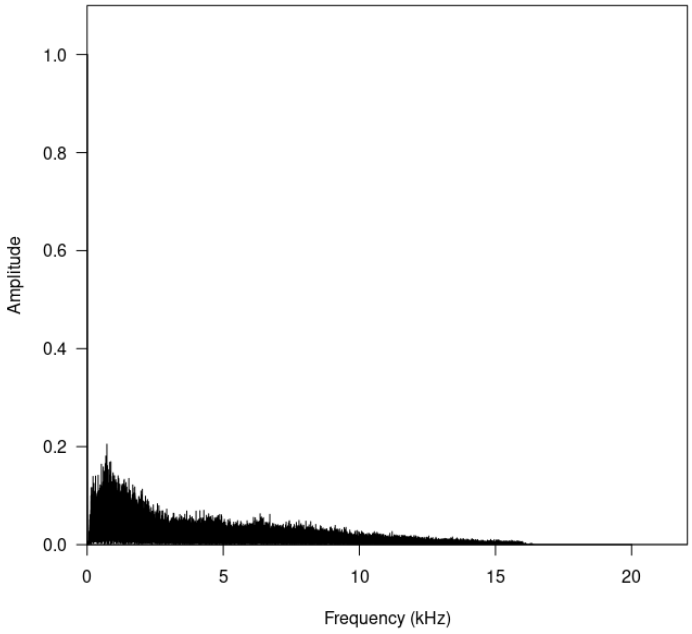


Figura 9: Periodograma "rain_short.mp3"

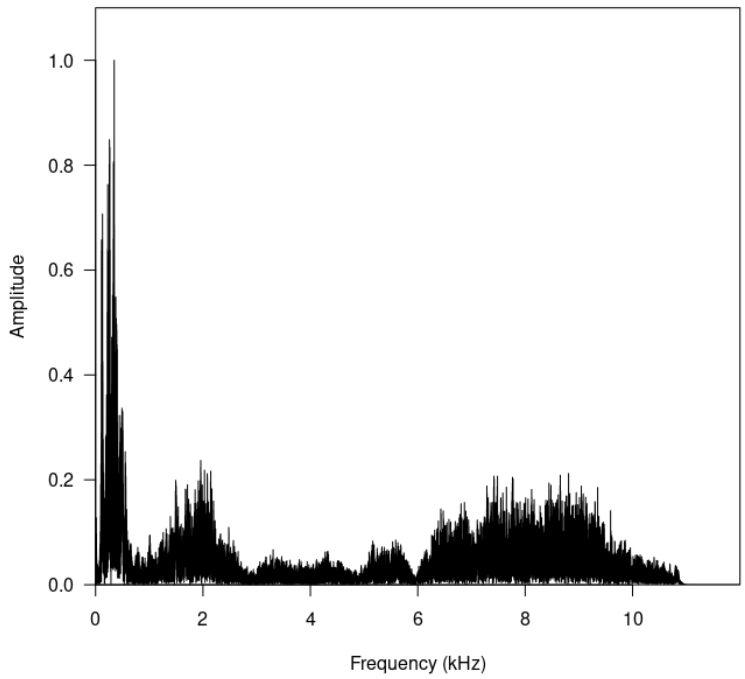


Figura 10: Periodograma "speak.mp3"

1B.2) Rangos de las frecuencias

Para el sonido de guitarra *"guitar.mp3"* las frecuencias están entre 0 y 13 kHz, descartando amplitudes muy bajas el rango apreciable está entre 0 y 5kHz.

Para el sonido de lluvia *"rain_short.mp3"* las frecuencias están entre 0 y 17 kHz, descartando amplitudes muy bajas el rango apreciable está entre 0 y 15kHz.

Para el sonido del habla humana *"speak.mp3"* las frecuencias están entre 0 y 13KHz.

1B.3) Espectrogramas de los sonidos

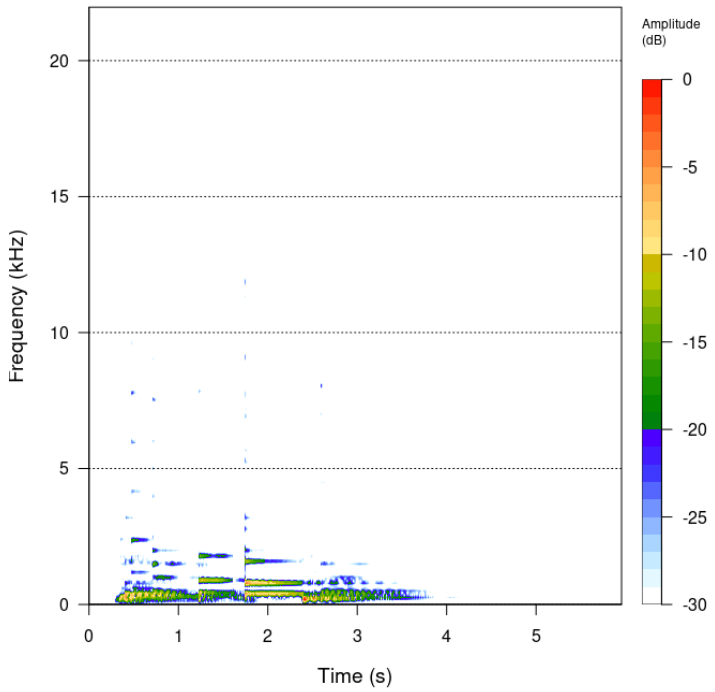


Figura 11: Espectrograma "guitar.mp3"

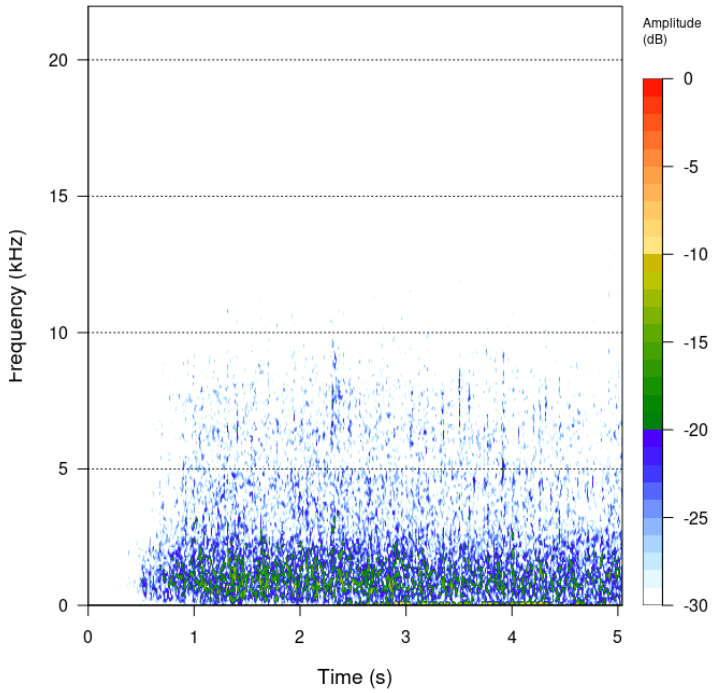


Figura 12: Espectrograma "rain_short.mp3"

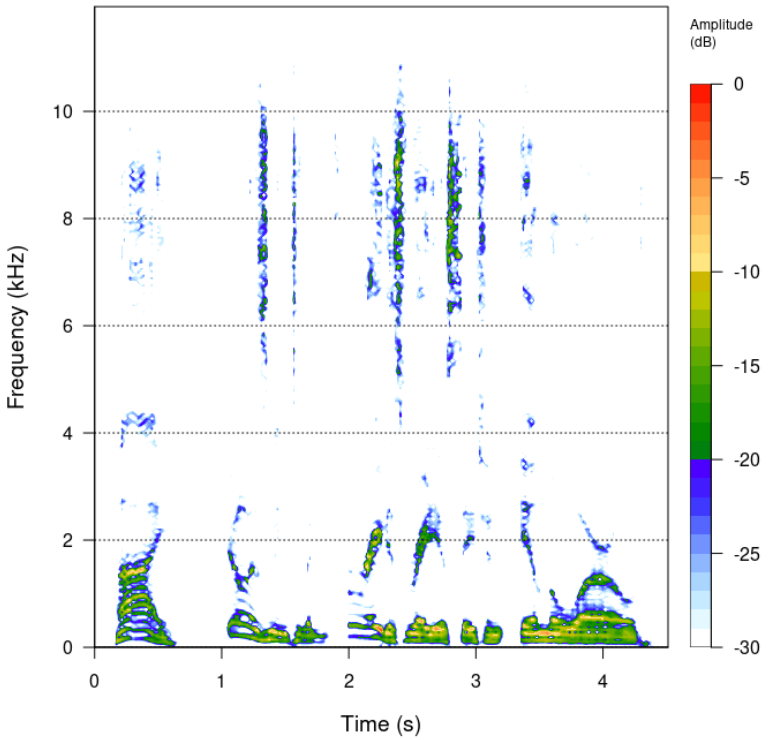


Figura 13: Espectrograma "speak.mp3"

1B.4) Filtros pasa-alto y pasa-bajo

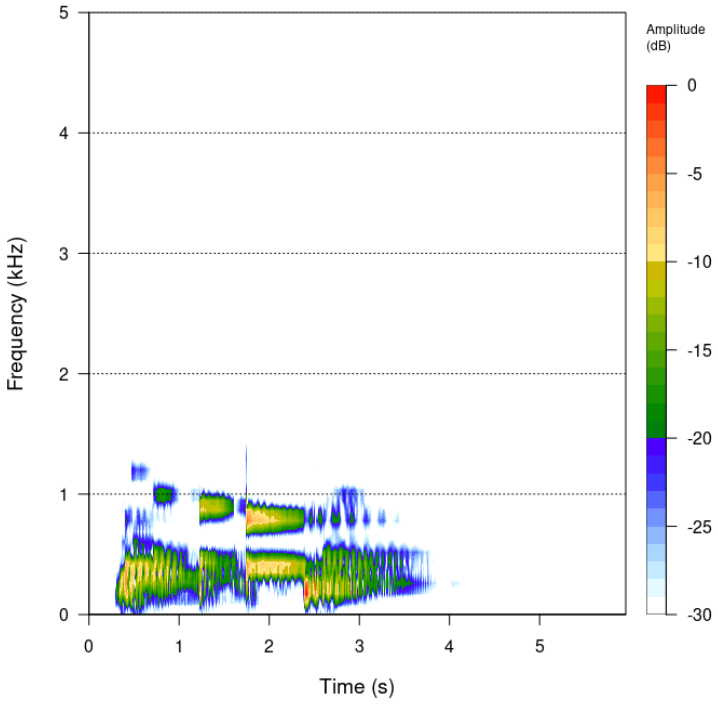


Figura 13: Espectrograma de "guitar.mp3" con un filtro pasa-bajo de 1500kHz

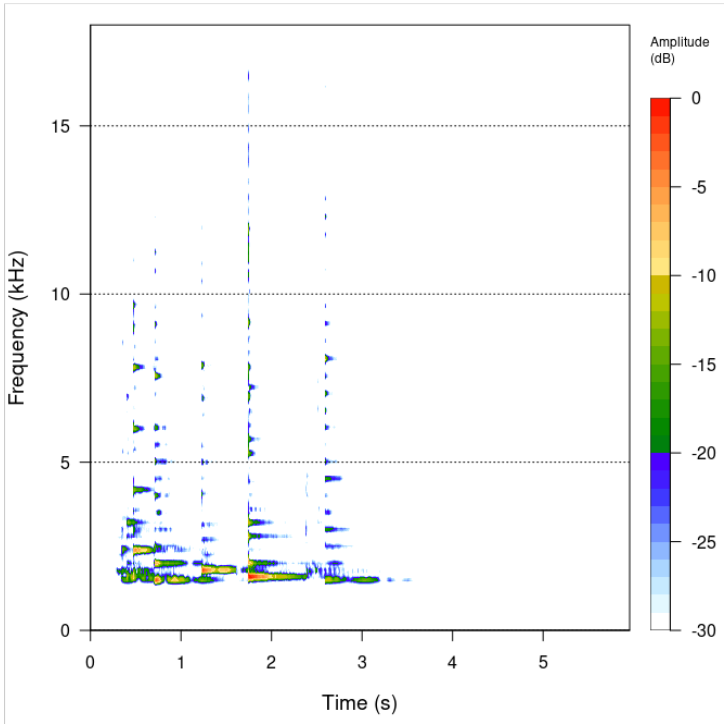


Figura 14: Espectrograma de "guitar.mp3" con un filtro pasa-alto de 1500kHz

Parte C - Análisis

1C.1) Frecuencias de los sonidos, comparando entre periodograma y espectrograma

Para el sonido de guitarra *"guitar.mp3"*, según el periodograma, las frecuencias están entre 0 y 13 kHz, descartando amplitudes muy bajas el rango apreciable está entre 0 y 5kHz. Sin embargo, al mirar el espectrograma se puede ver que las frecuencias apreciables están entre 0 y 3kHz, con el resto siendo ruido.

Para el sonido de lluvia *"rain_short.mp3"*, según el periodograma, las frecuencias están entre 0 y 17 kHz, descartando amplitudes muy bajas el rango apreciable está entre 0 y 15kHz.Sin embargo, al mirar el espectrograma se puede ver que las frecuencias apreciables están entre 0 y 10kHz, con el resto siendo ruido.

Para el sonido de habla humana *"speak.mp3"*, tanto en el periodograma como en el espectrograma, las frecuencias están entre 0 y 13KHz.

1C.2 y 3) Análisis de sonido del habla

Frase hablada: "Hi, you're listening to the greatest radio station in the world!"

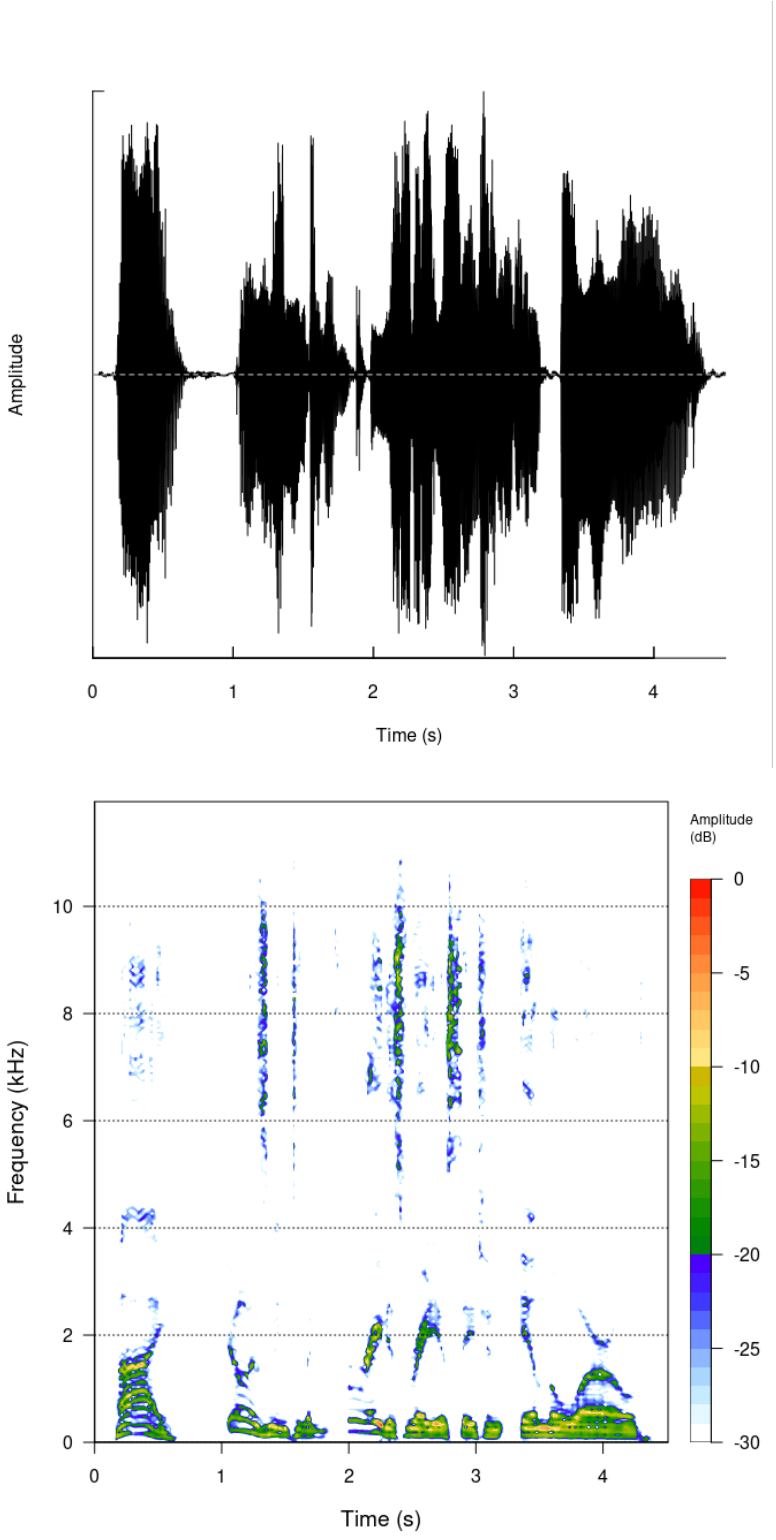


Figura 16: Oscilograma y espectrograma de "speak.mp3" apilados

En una primera etapa, mirando el oscilograma se pueden identificar claramente los silencios, entre ellos el más grande siendo el primero. En este caso queda claro que la primera palabra "Hi" se encuentra empieza en el segundo 0.16 y termina en el 0.6.

La siguiente palabra es "you're", que si bien técnicamente son dos palabras (*you* y *are)*, se las junta en inglés y suena como una sola. Mirando el espectrograma, se encuentra un sonido sibilante en el segundo 1.3 aproximadamente, que parecería corresponder a la ***s*** de *li**s**tening.* También hay un pequeño silencio justo antes, aproximadamente en el segundo 1.2 que probablemente corresponda a la ***l*** de ***l**istening.* Por lo tanto la siguiente palabra "you're" se encuentra entre el segundo 1 y el 1.2.

Por lo mencionado anteriormente, "listening" parecería arrancar en el segundo 1.3. Teniendo en cuenta que en el 1.6 se encuentra otro ruido agudo y fuerte, este probablemente refiera a la **t** de ***t**o.* Entonces, "listening" estaría entre el segundo 1.3 y el 1.6 y "to" en el 1.6. "the" va inmediatamente después de "the" en el segundo 1.65 al 1.75.

La siguiente palabra es una larga, "greatest", y esta parecería arrancar en el segundo 2 y terminar en el 2.4, donde se encuentra un bajón de volumen y ruido agudo, probablemente correspondiente a la **t** de *greates**t.***

Las próximas dos son relativamente fáciles de interpretar, ya que "radio" empieza con una **r,** que hace poco ruido y "station" con una **s**, que es fácil de interpretar en el espectrograma. Esto pone a las siguientes palabras en: 2.45s a 2.8s para "radio" y 2.8s a 3.2 para "station".

La próxima palabra "in" es muy distinguible ya que comienza con una vocal fuerte y termina con una consonante silenciosa. Esta se encuentra entre 3.35s y 3.55s.

Finalmente, las últimas dos palabras, "the world", se pueden encontrar prestando atención al ruido de alta frecuencia en el espectrograma que corresponde al **th** de ***th**e* y al bajón de amplitud (**w)** seguido por la subida de amplitud (**o**) debido a la vocal. Por lo tanto, las últimas palabras están en: 3.56s a 3.75s para "the" y 3.7s a 4.4s para "world".

1C.4) Comparación de espectrogramas

La primera diferencia muy marcada entre los espectrogramas corresponde a la *Figura 13*, en contraste con la *Figura 11* y la *Figura 12*. La *Figura 13*, corresponde al sonido del habla que es de un locutor. Es probable que este sonido haya tenido algún tratamiento, ya que se puede ver como tiene muy poco ruido. Esto se puede ver ya que no cuenta con frecuencias muy altas con mínima amplitud. Dentro de estos, el peor es el de la guitarra, que sí cuenta con muchas.

Otra diferencia interesante es en la forma de los espectrogramas, especialmente si se fuera a comparar la *Figura 11* (guitarra) con la *Figura 12* (lluvia). En el espectrograma de la guitarra se pueden ver ondulaciones que corresponden a la vibración de las cuerdas, mientras que en el de la lluvia el espectrograma se muestra como puntitos de amplitud uniforme.

2. Ejercicio 2

Parte A - Diseño Conceptual

2A.a) Objetivo

Para este ejercicio se decidió por implementar una máquina expendedora. La máquina cuenta con una grilla de productos, cada uno con su código correspondiente, un tablero para hacer selecciones, una pantalla para mostrar información y un botón que abre un formulario para simular la carga de dinero.

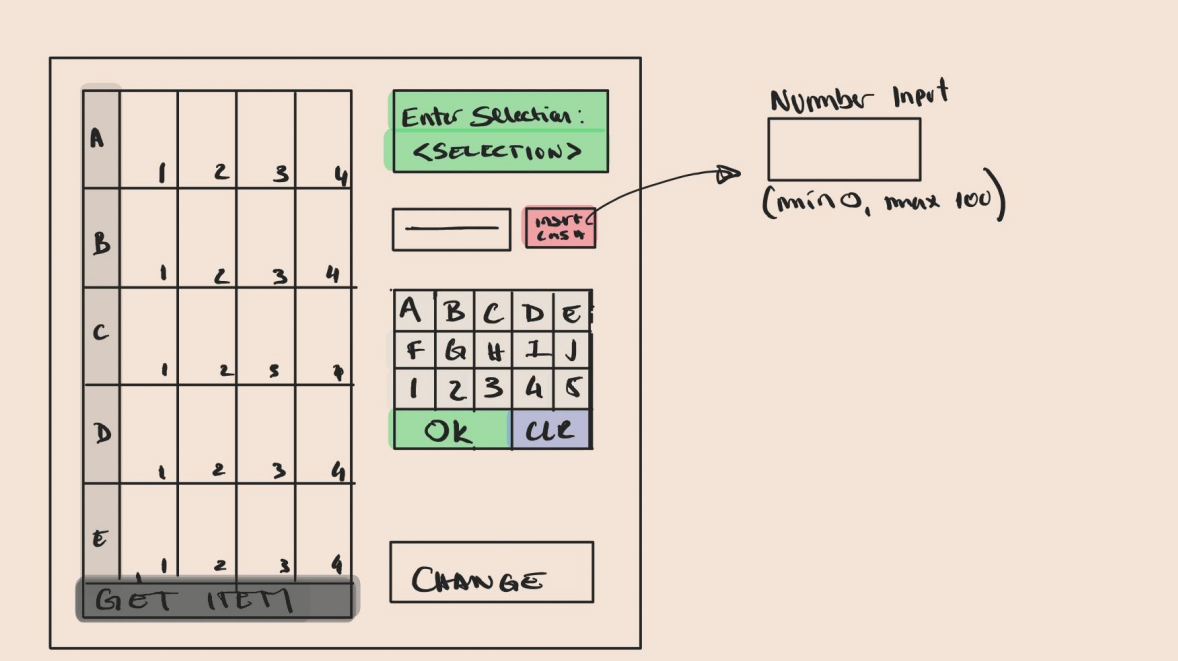


Diagrama de la simulación, previo a la implementación

Funciones de la simulación

1. Consultar precio: Igual que cualquier otra máquina expendedora, si se hace una selección antes de ingresar dinero, la misma muestra el precio del producto seleccionado.
2. Cargar dinero: Abre un formulario que permite ingresar un número entero entre 1 y 200 para cargar dinero. Luego muestra el balance actual y cambia la máquina a modo selección.
3. Hacer una selección: Ingresando un código a través del tablero se puede intentar hacer una selección. Si se cuenta con suficiente balance, la misma se considera exitosa, se muestra una pantalla con un mensaje para simular el producto expendido y se muestra el cambio que devolvería la máquina. Si no se cuenta con suficiente balance, la máquina

muestra un mensaje notificando al usuario que no cuenta con fondos suficientes y luego el usuario puede hacer otra selección o ingresar más dinero.

2A.b) Componentes

1. *Items*: Es un diccionario de los productos a la venta. La clave es el código de la máquina y cada uno cuenta con un nombre y un precio.
2. *Display:* Es la pantalla de la máquina expendedora.
3. *Balance*: Es el componente que maneja los fondos. Muestra el balance, el cambio (tras la compra exitosa) y permite agregar más dinero a través de un input.
4. *Keypad:* Es el tablero de la máquina. Cuenta con letras, botones, un botón de OK y uno de CLR para borrar la entrada de usuario.

2A.c) Actualización de variables

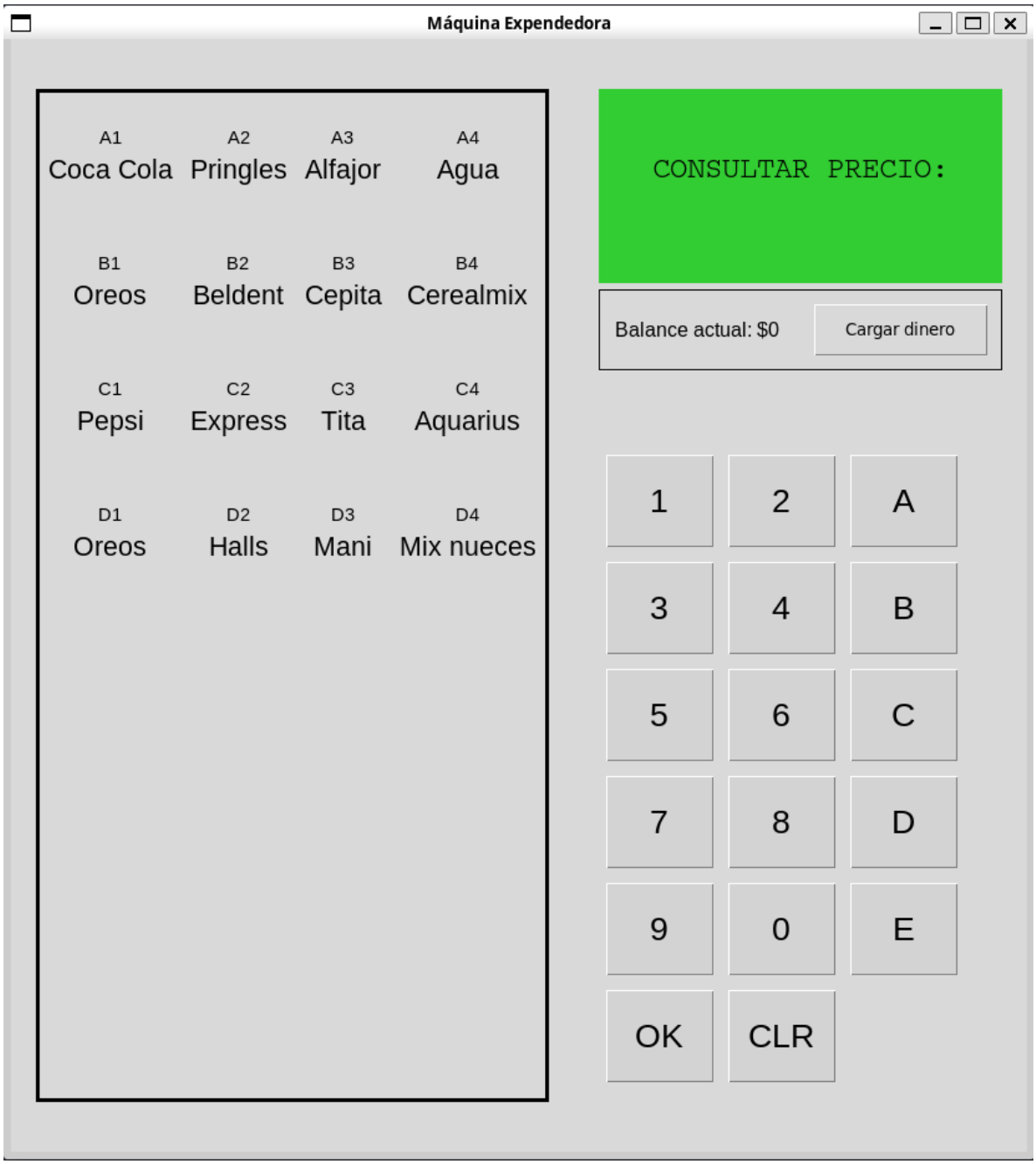
Para cambiar los productos se debe modificar la variable *items*. Se pueden cambiar los precios e inclusive se pueden agregar más productos. Actualmente van de la letra A a la D y de los números 1 al 4, pero se puede agregar una línea de números más (del 5 al 8) para cada letra y además agregar la letra E. Los elementos se acomodan automáticamente en la grilla ya que esta está generada de manera dinámica.

```
self.items = {
    "A1": {"name": "Coca Cola", "price": 15},
    "A2": {"name": "Pringles", "price": 10},
    "A3": {"name": "Alfajor", "price": 7.5},
    "A4": {"name": "Agua", "price": 12.5},
    "B1": {"name": "Oreos", "price": 17.5},
    "B2": {"name": "Beldent", "price": 5},
    "B3": {"name": "Cepita", "price": 18},
    "B4": {"name": "Cerealmix", "price": 19},
    "C1": {"name": "Pepsi", "price": 15},
    "C2": {"name": "Express", "price": 8},
    "C3": {"name": "Tita", "price": 5.5},
    "C4": {"name": "Aquarius", "price": 13.5},
    "D1": {"name": "Oreos", "price": 7.5},
    "D2": {"name": "Halls", "price": 3},
    "D3": {"name": "Mani", "price": 4},
    "D4": {"name": "Mix nueces", "price": 14},
}
```

Diccionario de productos

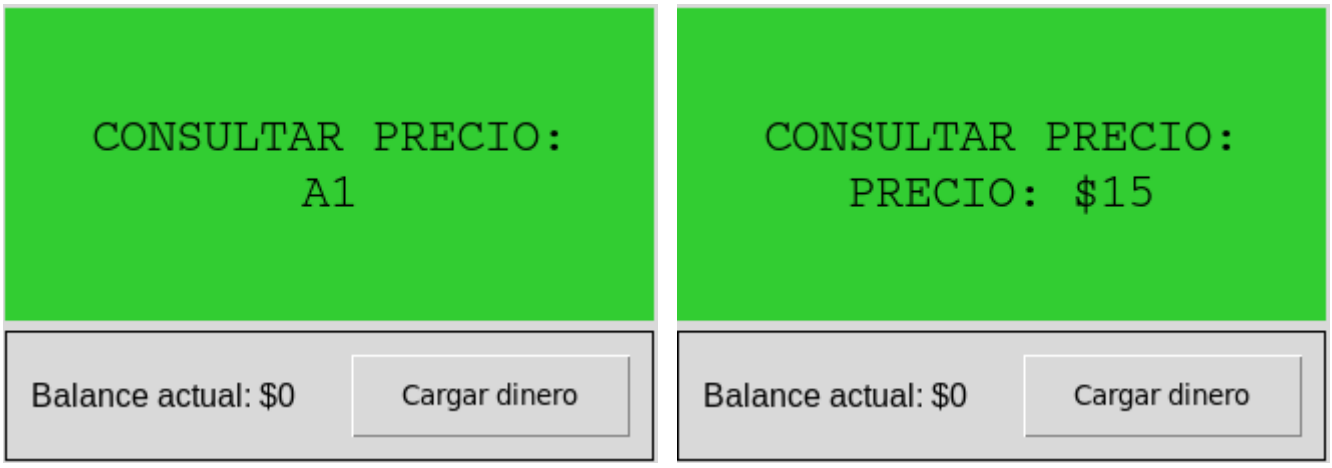
Parte B

2b.a) Interfaz implementada

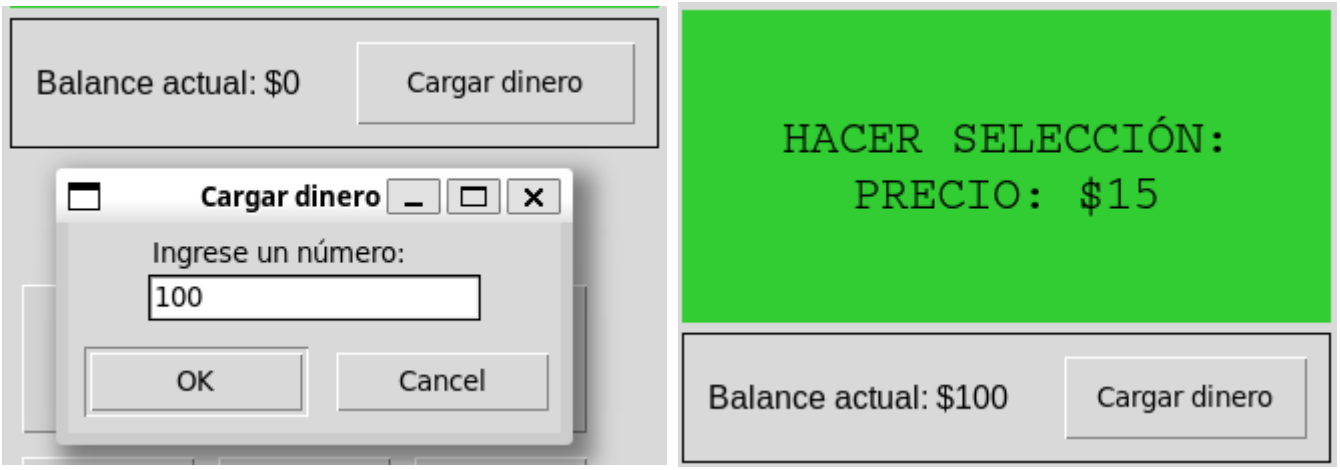


Interfaz en su estado 0

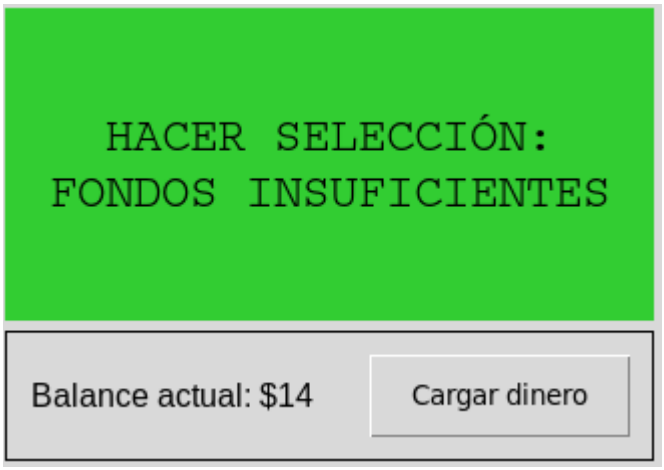
2B.b) Capturas en funcionamiento



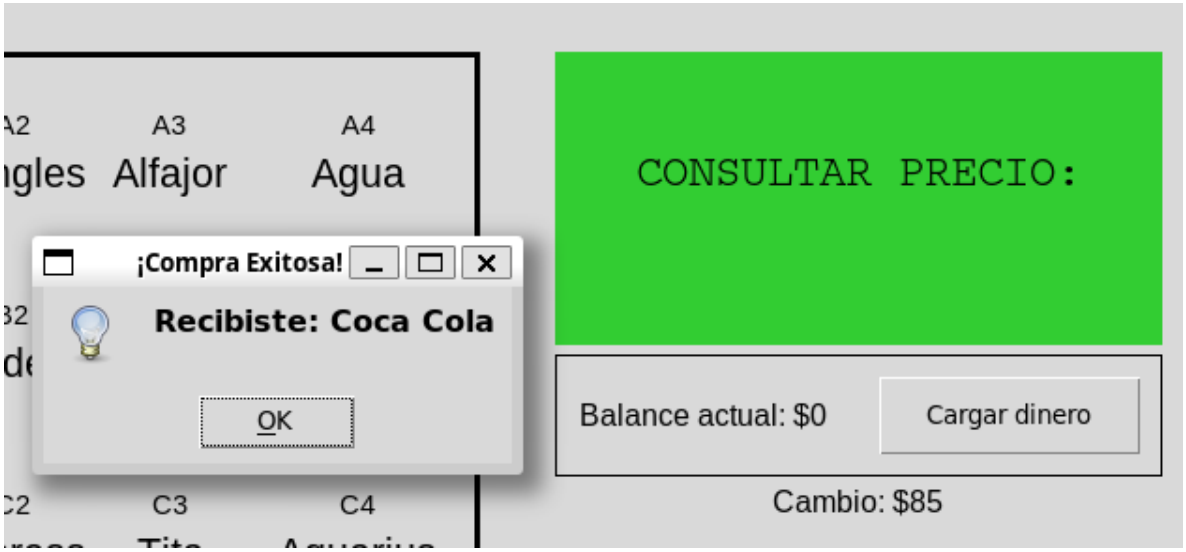
A la izquierda como cambia la pantalla al apretar botones en el tablero, a la derecha el estado luego de apretar "OK" para consultar el precio



A la izquierda el popup que aparece al apretar el botón de "Cargar dinero", a la derecha luego de darle "Ok" a la carga de dinero. Nótese que cambió el balance actual y que ahora la máquina se encuentra en modo de hacer selección en vez de consulta de precio.

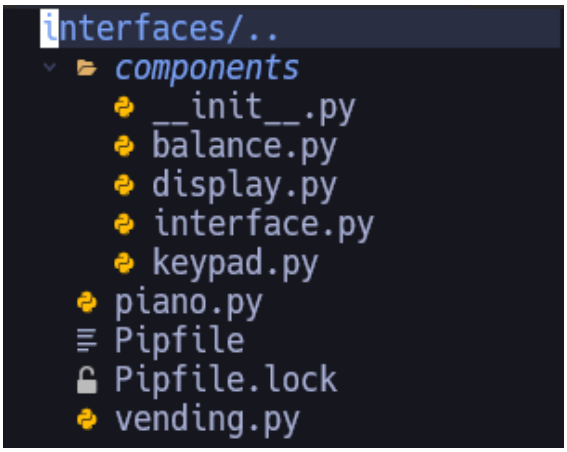


Seleccionando un elemento de $15 con solo $14



Resultado final de seleccionar el producto A1 después de haber cargado $100

2B.c) Estructura de código



Los archivos *balance.py, display.py* y *keypad.py* corresponden al componente de fondos, pantalla y tablero correspondientemente. El archivo *interface.py* cuenta con el manejo de estado de estos tres componentes a la vez y maneja la interacción entre los mismos.
Desde el archivo *vending.py* se arma la interfaz de productos, se conecta con la interfaz de *interface.py* y se arma la aplicación de tkinter.

Explicación de código

En esta sección se definen los productos y se arma la grilla de productos.

Se instancia la clase *Interface* mencionada previamente con una función *purchase* (mostrada a continuación) que hace la acción de comprar y al final se encajan los componentes en el marco.

```
class VendingMachineApp:
    def __init__(self, root):
        self.root = root
        self.root.title("Máquina Expendedora")
        self.items = {
            "A1": {"name": "Coca Cola", "price": 15},
            "A2": {"name": "Pringles", "price": 10},
            "A3": {"name": "Alfajor", "price": 7.5},
            "A4": {"name": "Agua", "price": 12.5},
            "B1": {"name": "Oreos", "price": 17.5},
            "B2": {"name": "Beldent", "price": 5},
            "B3": {"name": "Cepita", "price": 18},
            "B4": {"name": "Cerealmix", "price": 19},
            "C1": {"name": "Pepsi", "price": 15},
            "C2": {"name": "Express", "price": 8},
            "C3": {"name": "Tita", "price": 5.5},
            "C4": {"name": "Aquarius", "price": 13.5},
            "D1": {"name": "Oreos", "price": 7.5},
            "D2": {"name": "Halls", "price": 3},
            "D3": {"name": "Mani", "price": 4},
            "D4": {"name": "Mix nueces", "price": 14},
        }

        row_num, col_num = 0, 0
        items_grid = tk.Frame(root)
        for item_key in self.items.keys():
            itm = self.items[item_key]
            frame = tk.Frame(items_grid)
            itembox = tk.Label(frame, text=f"{itm['name']}", font=("Arial", 16))
            itembox.place(relx=0.5, rely=0.5, anchor=tk.CENTER)
            itemkey = tk.Label(frame, text=f"{item_key}", font=("Arial", 11))
            itembox.pack(expand=True, fill=tk.BOTH, side=tk.BOTTOM)
            itemkey.pack(expand=True, fill=tk.BOTH, side=tk.BOTTOM)

            frame.grid(row=row_num, column=col_num, padx=5, pady=25)
            col_num += 1
            if col_num > 3:
                col_num = 0
                row_num += 1

        self.interface = Interface(root, self.__purchase)
        self.interface.pack(expand=True, fill=tk.BOTH, side=tk.RIGHT, padx=20, pady=40)
        items_grid.config(highlightbackground="black", highlightthickness=3)
        items_grid.pack(expand=True, fill=tk.BOTH, side=tk.LEFT, padx=20, pady=40)
```

```
    def __purchase(self):
        input = self.interface.get_input()
        item = self.items.get(input, None)
        if item is None:
            self.interface.set_error("ITEM INVÁLIDO")
        else:
            self.interface.checkout(item)
```

Esta es la función que se le pasa a *Interface*. La función funciona a modo de *callback*, es decir, la ejecuta *Interface* cuando el usuario presiona el botón OK.

```
class Interface(tk.Frame):
    def __init__(self, parent, checkout_func):
        super().__init__(parent)

        self.keypad_state = "INPUTTING"
        self.topline = "CONSULTAR PRECIO:"
        self.bottomline = ""

        self.checking = True

        self.display = Display(self, self.topline)
        self.balance = Balance(self, 0, lambda: self.__set_checking(False))
        self.keypad = Keypad(self, self.__keypress, checkout_func, self.__clear)

        self.display.pack(expand=True, fill=tk.BOTH, side=tk.TOP, pady=(0, 5))
        self.balance.pack(expand=True, fill=tk.BOTH, side=tk.TOP, pady=(0, 25))
        self.keypad.pack(expand=True, fill=tk.BOTH, side=tk.BOTTOM)
```

En esta sección se instancia define la *Interface* del *interface.py* mencionada anteriormente. Se definen variables para manejar el estado compuesto de la misma, como *keypad_state, checking, topline y bottomline.* Estas últimas corresponden a las dos líneas de la pantalla de la máquina. Se instancia aquí el resto de los componentes, pasándose las funciones necesarias.

```
def __keypress(self, key):
    if self.keypad_state == "INPUTTING" and len(self.bottomline) < 2:
        self.bottomline += key
        self.display.update_display(self.topline, self.bottomline)
    if self.keypad_state == "ERROR":
        self.keypad_state = "INPUTTING"
        self.bottomline = key
        self.display.update_display(self.topline, self.bottomline)

def __clear(self):
    self.bottomline = ""
    self.display.update_display(self.topline, self.bottomline)
```

Estas son las funciones para manejar el presionado de teclas del tablero, que admite nada más que 2 caracteres, y la del botón CLR que borra el input del usuario.

```
def checkout(self, item):
    if self.checking:
        self.__set_botomline(f"PRECIO: ${item['price']}")
        self.keypad_state = "ERROR"
        return

    if self.balance.balance < item["price"]:
        self.__set_botomline("FONDOS INSUFICIENTES")
        self.keypad_state = "ERROR"
        return

    new_balance = self.balance.balance - item["price"]
    self.__set_botomline("")
    self.balance.update_balance(0)
    self.balance.update_change(new_balance)
    self.__set_checking(True)
    messagebox.showinfo("¡Compra Exitosa!", f"Recibiste: {item['name']}")
```

Esta es la función que se ejecuta al presionar el botón OK. Nótese que primero hace un chequeo de estado, para ver si debe mostrar el precio o hacer una compra y luego hace las modificaciones correspondientes.

```python
class Balance(tk.Frame):
    def __init__(self, parent, balance, on_success):
        super().__init__(parent)
        self.balance = balance

        self.frame = tk.Frame(self)
        self.display_label = tk.Label(
            self.frame, text=f"Balance actual: ${balance}", font=("Arial", 12)
        )
        self.change_label = tk.Label(self, text="", font=("Arial", 12))

        self.display_label.pack(pady=20, fill=tk.BOTH, side=tk.LEFT, padx=10)
        self.change_label.pack(pady=2, fill=tk.BOTH, side=tk.BOTTOM, padx=10)
        self.frame.pack(fill=tk.BOTH, side=tk.BOTTOM)

        def open_number_input_window():
            number = simpledialog.askinteger(
                "Cargar dinero", "Ingrese un número:", minvalue=1, maxvalue=200
            )
            if number is not None:
                self.balance += number
                self.display_label.config(text=f"Balance actual: ${self.balance}")
                on_success()

        open_button = tk.Button(
            self.frame, text="Cargar dinero", command=open_number_input_window
        )
        open_button.pack(
            expand=True, fill=tk.BOTH, side=tk.RIGHT, padx=(15, 10), pady=10
        )

        self.frame.config(highlightbackground="black", highlightthickness=1)

    def update_balance(self, new_balance):
        self.balance = new_balance
        self.display_label.config(text=f"Balance actual: ${self.balance}")

    def update_change(self, change):
        self.change_label.config(text=f"Cambio: ${change}")
```

El siguiente código corresponde al componente de balance. Se define la clase extendiendo de *tkinter.Frame* para generar un componente customizado. Dentro del mismo se define un *Frame* que es el que contiene el mensaje de balance actual y el botón que abre el formulario y luego este es embebido en la clase junto con el mensaje de cambio. El balance y el cambio se actualizan a través de las funciones *update_balance* y *update_change.*

Los componentes de *Display* y de *Keypad* están hechos de manera análoga a este. La totalidad del código se encuentra en el anexo.

3. Anexo

Código Ejercicio 1

```
library(tuneR)
library(seewave)

guitar <- readMP3("guitar.mp3")
rain <- readMP3("rain_short.mp3")
speak <- readMP3("speak.mp3")

# Ejercicio 1A.1
oscillo(guitar)
oscillo(rain)
oscillo(speak)

# Ejercicio 1A.2
oscillo(guitar, from = 1, to = 1.01)
oscillo(rain, from = 1, to = 1.005)
oscillo(speak, from = 1.004, to = 1.01)

# Ejercicio 1A.3
speak_b <- normalize(speak, unit = "16", level = 0.1)
oscillo(speak, from = 1.004, to = 1.01)
axis(side = 2, las = 2)
oscillo(speak_b, from = 1.004, to = 1.01)
axis(side = 2, las = 2)


# Ejercicio 2A.1
spec(guitar)
axis(side = 2, las = 2)

spec(rain)
axis(side = 2, las = 2)

spec(speak)
axis(side = 2, las = 2)

# Ejercicio 2A.3
spectro(guitar)
spectro(rain)
spectro(speak)

# Ejercicio 2A.4
guitar_pb <- ffilter(guitar, to = 1500, output = "Wave")
guitar_pb <- normalize(guitar_pb, unit = "16")
spectro(guitar_pb, flim = c(0, 5))

guitar_pa <- ffilter(guitar, from = 1500, output = "Wave")
guitar_pa <- normalize(guitar_pa, unit = "16")
spectro(guitar_pa, flim = c(0, 18))
```

Código Ejercicio 2

```
import tkinter as tk
from tkinter import simpledialog


class Balance(tk.Frame):
        def __init__(self, parent, balance, on_success):
        super().__init__(parent)
        self.balance = balance

        self.frame = tk.Frame(self)
        self.display_label = tk.Label(
        self.frame, text=f"Balance actual: ${balance}", font=("Arial", 12)
        )
        self.change_label = tk.Label(self, text="", font=("Arial", 12))

        self.display_label.pack(pady=20, fill=tk.BOTH, side=tk.LEFT, padx=10)
        self.change_label.pack(pady=2, fill=tk.BOTH, side=tk.BOTTOM, padx=10)
        self.frame.pack(fill=tk.BOTH, side=tk.BOTTOM)

        def open_number_input_window():
        number = simpledialog.askinteger(
        "Cargar dinero", "Ingrese un número:", minvalue=1, maxvalue=200
        )
        if number is not None:
        self.balance += number
        self.display_label.config(text=f"Balance actual: ${self.balance}")
        on_success()

        open_button = tk.Button(
        self.frame, text="Cargar dinero", command=open_number_input_window
        )
        open_button.pack(
        expand=True, fill=tk.BOTH, side=tk.RIGHT, padx=(15, 10), pady=10
        )

        self.frame.config(highlightbackground="black", highlightthickness=1)

        def update_balance(self, new_balance):
        self.balance = new_balance
        self.display_label.config(text=f"Balance actual: ${self.balance}")

        def update_change(self, change):
        self.change_label.config(text=f"Cambio: ${change}")
```

balance.py

```
import tkinter as tk


class Display(tk.Frame):
        def __init__(self, parent, topline="", bottomline=""):
        super().__init__(parent)

        self.topline = topline
        self.bottomline = bottomline

        self.configure(bg="lime green")
        self.display_label = tk.Label(
        self,
        text=f"{self.topline}\n{self.bottomline}",
        font=("Courier", 18),
        fg="black",
        bg="lime green",
        width=20,
        height=4,
        )
        self.display_label.pack(padx=20, pady=20)

        def update_display(self, topline, bottomline):
        self.topline = topline
        self.bottomline = bottomline
        self.display_label.config(text=f"{self.topline}\n{self.bottomline}")
```

display.py

```
import tkinter as tk


class Keypad(tk.Frame):
        def __init__(self, parent, keypress, ok, clear):
        super().__init__(parent)

        button_font = ("Arial", 20)
        button_bg = "lightgray"
        button_fg = "black"

        self.pack(pady=10)

        keypad_buttons = [
        "1",
        "2",
        "A",
        "3",
        "4",
        "B",
        "5",
        "6",
        "C",
        "7",
        "8",
        "D",
        "9",
        "0",
        "E",
        "OK",
        "Clear",
        ]

        row_num, col_num = 0, 0

        for key in keypad_buttons[:-2]:
        button = tk.Button(
        self,
        text=key,
        font=button_font,
        width=4,
        height=2,
        bg=button_bg,
        fg=button_fg,
        command=lambda key=key: keypress(key),
        )
        button.grid(row=row_num, column=col_num, padx=5, pady=5)
        col_num += 1
        if col_num > 2:
        col_num = 0
        row_num += 1
        ok_button = tk.Button(
        self,
```

```
text="OK",
font=button_font,
width=4,
height=2,
bg=button_bg,
fg=button_fg,
command=lambda _="OK": ok(),
)
ok_button.grid(row=row_num, column=col_num, padx=5, pady=5)
cancel_button = tk.Button(
self,
text="CLR",
font=button_font,
width=4,
height=2,
bg=button_bg,
fg=button_fg,
command=lambda _="CLR": clear(),
)
cancel_button.grid(row=row_num, column=col_num + 1, padx=5, pady=5)
```

keypad..py

```
import tkinter as tk
from tkinter import messagebox
from .display import Display
from .balance import Balance
from .keypad import Keypad


class Interface(tk.Frame):
        def __init__(self, parent, checkout_func):
        super().__init__(parent)

        self.keypad_state = "INPUTTING"
        self.topline = "CONSULTAR PRECIO:"
        self.bottomline = ""

        self.checking = True

        self.display = Display(self, self.topline)
        self.balance = Balance(self, 0, lambda: self.__set_checking(False))
        self.keypad = Keypad(self, self.__keypress, checkout_func, self.__clear)

        self.display.pack(expand=True, fill=tk.BOTH, side=tk.TOP, pady=(0, 5))
        self.balance.pack(expand=True, fill=tk.BOTH, side=tk.TOP, pady=(0, 25))
        self.keypad.pack(expand=True, fill=tk.BOTH, side=tk.BOTTOM)

        def __keypress(self, key):
        if self.keypad_state == "INPUTTING" and len(self.bottomline) < 2:
        self.bottomline += key
        self.display.update_display(self.topline, self.bottomline)
        if self.keypad_state == "ERROR":
        self.keypad_state = "INPUTTING"
        self.bottomline = key
        self.display.update_display(self.topline, self.bottomline)

        def __clear(self):
        self.bottomline = ""
        self.display.update_display(self.topline, self.bottomline)

        def __set_checking(self, val):
        if val:
        self.checking = val
        self.__set_topline("CONSULTAR PRECIO:")
        self.__clear()
        else:
        self.checking = val
        self.__set_topline("HACER SELECCIÓN:")

        def __set_topline(self, text):
        self.topline = text
        self.display.update_display(self.topline, self.bottomline)

        def __set_botomline(self, text):
        self.bottomline = text
```

```
self.display.update_display(self.topline, self.bottomline)

def get_balance(self):
return self.balance.balance

def get_input(self):
return self.bottomline

def set_error(self, error):
self.bottomline = error
self.keypad_state = "ERROR"
self.display.update_display(self.topline, self.bottomline)

def checkout(self, item):
if self.checking:
self.__set_botomline(f"PRECIO: ${item['price']}")
self.keypad_state = "ERROR"
return

if self.balance.balance < item["price"]:
self.__set_botomline("FONDOS INSUFICIENTES")
self.keypad_state = "ERROR"
return

new_balance = self.balance.balance - item["price"]
self.__set_botomline("")
self.balance.update_balance(0)
self.balance.update_change(new_balance)
self.__set_checking(True)
messagebox.showinfo("¡Compra Exitosa!", f"Recibiste: {item['name']}")
```

interface..py

```
import tkinter as tk
from components.interface import Interface


class VendingMachineApp:
        def __init__(self, root):
        self.root = root
        self.root.title("Máquina Expendedora")
        self.items = {
        "A1": {"name": "Coca Cola", "price": 15},
        "A2": {"name": "Pringles", "price": 10},
        "A3": {"name": "Alfajor", "price": 7.5},
        "A4": {"name": "Agua", "price": 12.5},
        "B1": {"name": "Oreos", "price": 17.5},
        "B2": {"name": "Beldent", "price": 5},
        "B3": {"name": "Cepita", "price": 18},
        "B4": {"name": "Cerealmix", "price": 19},
        "C1": {"name": "Pepsi", "price": 15},
        "C2": {"name": "Express", "price": 8},
        "C3": {"name": "Tita", "price": 5.5},
        "C4": {"name": "Aquarius", "price": 13.5},
        "D1": {"name": "Oreos", "price": 7.5},
        "D2": {"name": "Halls", "price": 3},
        "D3": {"name": "Mani", "price": 4},
        "D4": {"name": "Mix nueces", "price": 14},
        }

        row_num, col_num = 0, 0
        items_grid = tk.Frame(root)
        for item_key in self.items.keys():
        itm = self.items[item_key]
        frame = tk.Frame(items_grid)
        itembox = tk.Label(frame, text=f"{itm['name']}", font=("Arial", 16))
        itembox.place(relx=0.5, rely=0.5, anchor=tk.CENTER)
        itemkey = tk.Label(frame, text=f"{item_key}", font=("Arial", 11))
        itembox.pack(expand=True, fill=tk.BOTH, side=tk.BOTTOM)
        itemkey.pack(expand=True, fill=tk.BOTH, side=tk.BOTTOM)

        frame.grid(row=row_num, column=col_num, padx=5, pady=25)
        col_num += 1
        if col_num > 3:
        col_num = 0
        row_num += 1

        self.interface = Interface(root, self.__purchase)
        self.interface.pack(expand=True, fill=tk.BOTH, side=tk.RIGHT, padx=20, pady=40)
        items_grid.config(highlightbackground="black", highlightthickness=3)
        items_grid.pack(expand=True, fill=tk.BOTH, side=tk.LEFT, padx=20, pady=40)

        def __purchase(self):
        input = self.interface.get_input()
        item = self.items.get(input, None)
        if item is None:
```

```
            self.interface.set_error("ITEM INVÁLIDO")
            else:
            self.interface.checkout(item)


if __name__ == "__main__":
            root = tk.Tk()
            app = VendingMachineApp(root)
            root.mainloop()
```

vending.py