

# UNIVERSIDAD NACIONAL DE COLOMBIA



## FACULTAD DE INGENIERÍA



OBJECT ORIENTED PROGRAMMING

ENGINEERING FACULTY

WORKSHOP 4

AUTHORS:

Eddy Johan Tocancipa Muñoz

Valeria Aranda Pacheco

Luis Santiago Vargas Rodríguez

Bogotá DC. 2025

## 1. UML AND CRC CARDS:

Class: House		
Responsibilities:	Collaborators:	Interface
It displays the project title and main menu.	Rooms	LAYER #1:  This class will be the main and first interface that will appear in the program. From here there'll be buttons that take the user to the specific room.
Load the general house layout (bedroom, living room, kitchen).	Component Panel	
Detect clicks on rooms and zoom into the selected one.	Connection Simulator	
Load the corresponding side panel according to the selected room.		

Class: Kitchen		
Responsibilities:	Collaborators:	Interface
Graphically represent a kitchen.	Kitchen Electronics	LAYER #2,1:  This class will display the layout of a kitchen, where the user could drag components.
Serves as a structure for the classes attached to the kitchen behaviour.	Bug	
Contains the furniture of a kitchen.	Temperatures	

Class: Bedroom		
Responsibilities:	Collaborators:	Interface
Graphically represent a Bedroom.	Bedroom electronics	LAYER #2,2:  This class will display the layout of a Bedroom, where the user could drag components.
Serves as a structure for the classes attached to the kitchen behaviour.		
Contains the furniture of a Bedroom..		

Class: Living Room		
Responsibilities:	Collaborators:	Interface
Graphically represent a Living Room.	Living Room Electronics	LAYER #2,3:  This class will display the layout of a Living Room, where the user could drag components.
Serves as a structure for the classes attached to the living room behaviour.		
Contains the furniture of a Livingroom.		

**Class: Kitchen Electronics LAYER #2,1**

<b>Responsibilities:</b>	<b>Collaborators:</b>	<b>Interface</b>
Represents sensors	Bulb	
Store its electrical properties (voltage, polarized)	Heat Sensor	This class serves as an inheritance. It will not display on the program.
Respond to events (turn on/off, connection error, alarm).	Movement Sensor	It could serve as a "Column" where the icons of each component appear.
	Resistance	

**Class: Living Room Electronics LAYER #2,3**

<b>Responsibilities:</b>	<b>Collaborators:</b>	<b>Interface</b>
Represents Electromechanics	Resistance	
Store its electrical properties (voltage, power)	TV	This class serves as an inheritance. It will not display on the program.
Respond to events (turn on/off, connection error, display, audio).	Radio	It could serve as a "Column" where the icons of each component appear.

Class: Bedroom Electronics LAYER #2,2		
Responsibilities:	Collaborators:	Interface
Represent Lights	Resistance	This class serves as an inheritance. It will not display on the program.
Store its electrical properties (voltage, Luminosity)	Bulb	
Respond to events (turn on/off, connection error, light).	Desk Lamp	It could serve as a "Column" where the icons of each component appear.
	Lamp	

Class: Cables LAYER #2,(1,2,3)		
Responsibilities:	Collaborators:	Interface
Represent connections between components.	Kitchen, Bedroom, Livingroom Electronics	This class will appear on every room layout. It will be shown as a cable that connects each component to a resistance, then to a plug.
Verify electrical continuity.	Plug	
It Detects connection errors (short circuits, missing ground, etc.).	Test	
Conducts current	Control Panel	

Class: Result Panel LAYER #2, (1,2,3)		
Responsibilities:	Collaborators:	Interface
Display messages about the simulation status.	Connection Simulator	This class could appear as text. where from each component it will say if it's on or off.
Indicate whether the components work properly or have malfunctions.	Rooms	ways of implementing it are still discussed
Show specific alerts (Sensor not calibrated correctly, Short circuit in living room).		

Class: Resistance LAYER #2, (1,2,3)		
Responsibilities	Collaborators	Interface
Allows to regulate the current	Electronic Elements	
Prevents electronics to burn	Plug	
Gives ohms	Cables	

Class: Bulb LAYER #2, (1,2)		
Responsibilities	Collaborators	Interface
Generates light	Resistance	
Gives differents intensities of light	Cables	
Consume Volts		

**Class: Lamp LAYER #2,2**

Responsibilities	Collaborators	Interface
Can give different colors of lights	Resistance	It will have its own icon and appear in the bedroom. It could be connected to a cable.
Gives different intensities of lights	Cables	
Consume volts		

**Class: Desk Lamp LAYER #2,2**

Responsibilities	Collaborators	Interface
Gives different intensities of lights	Resistance	It will have its own icon and appear in the bedroom. It could be connected to a cable.
Consume Volts	Cables	

**Class: Heat Sensor LAYER #2,1**

Responsibilities	Collaborators	Interface
Detects high temperatures	Resistance	It will have its own icon and appear in the kitchen. It could be connected to a cable.
Can give an alarm when temperatures are high	Cables	
Parameters vary by volts received	Bulb	

**Class: Motion Sensor LAYER #2,1**

Responsibilities	Collaborators	Interface
Detects Movement	Resistance	It will have its own icon and appear in the kitchen. It could be connected to a cable.
Can give an alarm when it detects movements.	Cables	
Parameters vary by volts received	Bulb	

**Class: TV LAYER #2,3**

<b>Responsibilities</b>	<b>Collaborators</b>	<b>Interface</b>
Gives display	Stereo	
Gives audio	Resistance	
Consume watts	Cables	It will have its own icon and appear in the living room. It could be connected to a cable.

**Class: Radio LAYER #2,3**

<b>Responsibilities</b>	<b>Collaborators</b>	<b>Interface</b>
Gives audio	Stereo	
Plays music	Resistance	
Plays news	Cables	
Plays podcast		
Consume watts		

**Class: Plug LAYER #2,(1,2,3)**

<b>Responsibilities</b>	<b>Collaborators</b>	<b>Interface</b>
Gives volts	Electronic Components	
Give a terminal connection to the cables	Cables	It will have its own icon but this one will not be moved by the user. It will be attached to a specific location in every room.
Gives earth		The cables will connect with it.



### 3. File-based Persistence:

#### 1. Component Class

The Component class represents an object placed by the user within the room (Bulb, TV, Lamp, etc.). This class stores the information necessary to reconstruct the circuit when loaded from a file.

Code:

```
class Component:  
    def __init__(self, id, type, x, y, state):  
        self.id = id  
        self.type = type  
        self.x = x  
        self.y = y  
        self.state = state
```

Explanation:

- id: Unique identifier of the component.
- type: Type of component (Bulb, Radio, HeatSensor, etc.).
- x, y: Coordinates where the user placed the component.
- state: State of the component ('on', 'off', alert, etc.).

#### 2. Saving and Loading Methods within the Room Class

##### 2.1 to\_dict() Method

This method converts the room information into a JSON-compatible dictionary. The dictionary contains:

- Room name.
- All components and their attributes.
- The connections between components.
- Whether the circuit is plugged into the outlet.

##### 2.2 The save() Method

This method receives a filename and writes all the dictionary information to a JSON file using json.dump().

Code:

```
def save(self, filename="savefile.json"):  
    data = self.to_dict()  
    with open(filename, "w") as f:  
        json.dump(data, f, indent=4)
```

Explanation:

- Converts the data into a dictionary.
- Saves it in JSON format with indentation for readability.

### 2.3 The load() Method

This method loads the data from a JSON file and rebuilds all the components within the room.

Code:

```
def load(self, filename="savefile.json"):
    if not os.path.exists(filename):
        print("No save file exists.")
        return
    with open(filename, "r") as f:
        data = json.load(f)
        # rebuild components...
        # rebuild connections...
        # refresh interface if it exists
```

Explanation:

- Checks if the file exists.

- Loads the data using json.load().
- Recreates each component based on the loaded data.
- Restores connections.
- Calls redraw\_room() to update the interface.

### 3. Interface Integration (SAVE and LOAD Buttons)

The buttons within each room call the save() and load() methods of the current Room instance.

Code:

```
save_button.config(command=lambda: current_room.save())
load_button.config(command=lambda: current_room.load())
```

Explanation:

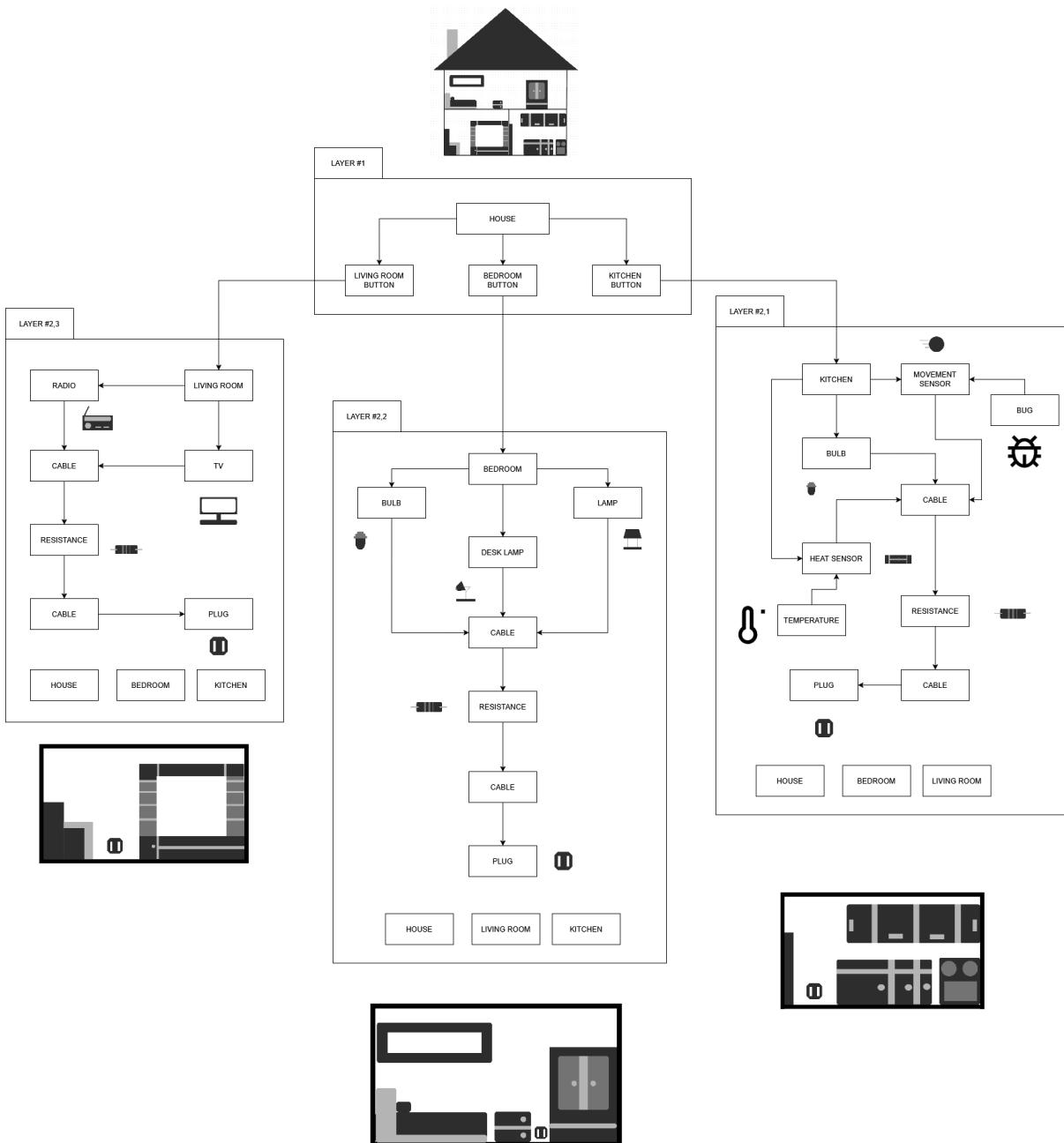
- When the user presses “SAVE”, the current state is saved.

When the user presses “LOAD”, the program rebuilds the circuit.

### 4. Conclusion

This system fulfills the functionality requested in the workshop:

Save a circuit, close the program, and upon reopening it, load exactly the positions, types, and states of each component, along with all the connections.



Link:

<https://drive.google.com/file/d/1ohkkRC-V8XbcULfD8vNI9kIGSiutKSeZ/view?usp=sharing>

In the last workshops the teacher couldn't access the full image because of problems with the applications that were used to create the diagrams.

We make sure that this link provides access to the diagram in a free way, without the need of accounts or downloads.

This way the teacher can see the diagram properly.

## **Interface User Manual:**

Hello! Welcome to ElecHouse.

Once you start our program you will see a house, this house is divided by 3 rooms, each one of them has unique electronic components that you could interact with.

Note that each room has a button that will take you to their respective interface.

Here's a short review of each room:

**1.** Bedroom: You will interact mostly with luminous components such as a bulb, a desk lamp or a lamp.

**2.** Livingroom: You will interact mostly with electrodomestics such as a TV and a radio.

**3.** Kitchen: You will interact mostly with sensors, only 2, the heat sensor and movement sensor. Those will be useful to interact with some events in the kitchen.

**NOTE:** Every room has a plug and a resistance component.

Now that you have chosen a room you will note that some icons appeared on the new layout. Those are your components!.

Here's a walkthrough in how we intended for you to use this simulator:

-Drag one of them and put it in a location you see adequate.

-Connect the component to the resistance.

-Connect the resistance to the plug, this to close the circuit.

-See if the component works!.

### **NOTES:**

1. If you want to connect multiple components in the same circuit, all of these should be connected to the same resistance.

2. Some sort of alert will show up to tell you if a component broke or is working.

Additionally, you will have at your disposal 3 Buttons at the top of the interface.

Here are their uses:

There 3 buttons will return to each room in the House!

Thanks for using our simulator and have fun!.

### **END NOTES:**

These notes are directed to the teacher, not the user.

At the time of creating the interface we are still testing the methods of adding values

so the user can establish voltage or resistance and how icons change according to these values. In the version delivered in this workshop, those things are not implemented yet or instructed in the manual.

```
import sys
import os
from PyQt5.QtWidgets import QApplication, QMainWindow, QLabel,
QPushButton, QWidget, QVBoxLayout, QHBoxLayout
from PyQt5.QtGui import QIcon, QFont, QPixmap, QPainter
from PyQt5.QtCore import Qt, QTimer

class MainWindow(QMainWindow):
    def __init__(self):
        super().__init__()

        self.setWindowTitle("Elechouse")
        self.setGeometry(500, 20, 1000, 1000)
        self.setWindowIcon(QIcon("microcontrolador.png"))

        # To avoid resizing before the window is fully initialized
        self.ventana_iniciada = False

        # ----- BACKGROUND IMAGE MAP -----
        self.imagenes = {
            "menu": "Casa Completa.jpeg",
            "sala": "Sala.jpeg",
            "cocina": "Cocina.jpeg",
            "room": "Habitación.jpeg",
        }
        self.imagen_actual = "menu"

        # ----- DEVICES PER ROOM -----
        # In living room and bedroom => lamp
        # In kitchen => radio
        self.dispositivos = {
            "sala": "lámpara",
            "room": "lámpara",
            "cocina": "radio",
        }

        # Image for each device (uses the ones you sent me)
        self.imagenes_dispositivos = {
```

```
"sala": "Lámpara Sala.png",
"room": "Lámpara Habitación.png",
"cocina": "Radio.png",
}

# Device state (connected / disconnected)
self.estado_dispositivos = {
    "sala": False,
    "room": False,
    "cocina": False,
}

# ----- CENTRAL WIDGET + LAYOUTS -----
contenedor = QWidget(self)
self.setCentralWidget(contenedor)

layout_principal = QVBoxLayout(contenedor)
layout_botones = QHBoxLayout()
layout_botones_acciones = QHBoxLayout()

# ----- TITLE -----
self.title_label = QLabel("ELECHOUSE", self)
self.title_label.setFont(QFont("Arial", 32))
self.title_label.setStyleSheet(
    "color:#000000;" +
    "background-color:white;" +
    "font-weight:bold;" +
    "font-style:italic;"
)
self.title_label.setAlignment(Qt.AlignCenter)

# ----- IMAGE LABEL -----
self.background = QLabel(self)
self.background.setAlignment(Qt.AlignCenter)
self.background.setMinimumHeight(400)

# ----- ROOM BUTTONS -----
self.button_sala = QPushButton("Sala", self)
self.button_cocina = QPushButton("Cocina", self)
self.button_room = QPushButton("Habitación", self)

for b in (self.button_sala, self.button_cocina,
          self.button_room):
```

```
b.setStyleSheet("font-size:22px; padding:10px;")

# Button to connect a device (lamp or radio)
self.button_dispositivo = QPushButton("Conectar dispositivo",
self)
    self.button_dispositivo.setStyleSheet("font-size:20px;
padding:8px;")
    self.button_dispositivo.setEnabled(False) # not used in the
main menu

# Connect all to the SAME generic function
self.button_sala.clicked.connect(lambda:
self.alternar_imagen("sala"))
    self.button_cocina.clicked.connect(lambda:
self.alternar_imagen("cocina"))
    self.button_room.clicked.connect(lambda:
self.alternar_imagen("room"))

# Connect device action

self.button_dispositivo.clicked.connect(self.conectar_dispositivo)

# ----- BUILD LAYOUT -----
layout_principal.addWidget(self.title_label)
layout_principal.addLayout(layout_botones)
layout_principal.addLayout(layout_botones_acciones)
layout_principal.addWidget(self.background, stretch=1)

layout_botones.addWidget(self.button_sala)
layout_botones.addWidget(self.button_cocina)
layout_botones.addWidget(self.button_room)

layout_botones_acciones.addWidget(self.button_dispositivo)

# Load initial image
QTimer.singleShot(0, self.actualizar_imagen)

# -----
# Generic toggle: menu <-> corresponding image
# -----
def alternar_imagen(self, clave):
    if self.imagen_actual == clave:
        self.imagen_actual = "menu"
```

```

        else:
            self.imagen_actual = clave

            # Enable or disable the button depending on the screen
            if self.imagen_actual == "menu":
                self.button_dispositivo.setEnabled(False)
                self.button_dispositivo.setText("Conectar dispositivo")
                print("You are in the main menu. There are no devices here.")
            else:
                self.button_dispositivo.setEnabled(True)

                if self.imagen_actual in self.dispositivos:
                    dispositivo = self.dispositivos[self.imagen_actual]
                    estado = self.estado_dispositivos[self.imagen_actual]
                    if estado:
                        self.button_dispositivo.setText(f"Desconectar {dispositivo}")
                    else:
                        self.button_dispositivo.setText(
                            f"Conectar {dispositivo.capitalize()}")
                )

                print(
                    f"You are in the {self.imagen_actual}. "
                    f"You can connect the {dispositivo}."
                )
            else:
                self.button_dispositivo.setEnabled(False)

        self.actualizar_imagen()

# -----
# Decide which image to show (with or without a device)
# -----
def actualizar_imagen(self):
    clave = self.imagen_actual
    ruta_base = self.imagenes[clave]

    # If the room has a device and it is connected, draw it
    if clave in self.dispositivos and self.estado_dispositivos[clave]:
        ruta_disp = self.imagenes_dispositivos.get(clave, None)

```

```
        if ruta_disp is not None:
            self.cargar_imagen_con_dispositivo(ruta_base,
ruta_disp)
            return

        # If there is no device or it does not apply, only the
background is loaded
        self.cargar_imagen(ruta_base)

# -----
# Load "normal" background image
# -----
def cargar_imagen(self, ruta):
    print(f"Entering: {ruta}")

    if not os.path.exists(ruta):
        self.background.setPixmap(QPixmap())
        self.background.setText(f"File not found:\n{ruta}")
        return

    pixmap = QPixmap(ruta)

    if pixmap.isNull():
        self.background.setPixmap(QPixmap())
        self.background.setText(f"Could not load:\n{ruta}")
    else:
        ancho = self.background.width()
        alto = self.background.height()

        pixmap_escalado = pixmap.scaled(
            ancho,
            alto,
            Qt.KeepAspectRatio,
            Qt.SmoothTransformation
        )

        self.background.setPixmap(pixmap_escalado)

    self.ventana_iniciada = True

# -----
# Load background + draw device on top (lamp or radio)
# -----
```

```
def cargar_imagen_con_dispositivo(self, ruta_fondo, ruta_disp):
    print(f"Entering: {ruta_fondo} with device {ruta_disp}")

    if not os.path.exists(ruta_fondo):
        self.background.setPixmap(QPixmap())
        self.background.setText(f"File not found:\n{ruta_fondo}")
        return

    if not os.path.exists(ruta_disp):
        print(f"Warning: device image not found: {ruta_disp}")
        self.cargar_imagen(ruta_fondo)
        return

    fondo = QPixmap(ruta_fondo)
    disp = QPixmap(ruta_disp)

    if fondo.isNull():
        self.background.setPixmap(QPixmap())
        self.background.setText(f"Could not load:\n{ruta_fondo}")
        return

    ancho = self.background.width()
    alto = self.background.height()

    fondo_escalado = fondo.scaled(
        ancho,
        alto,
        Qt.KeepAspectRatio,
        Qt.SmoothTransformation
    )

    # Scale device to a reasonable size (approx. 1/3 of height)
    alto_disp = alto // 3
    disp_escalado = disp.scaledToHeight(
        alto_disp,
        Qt.SmoothTransformation
    )

    # Position device centered horizontally and towards the bottom
    x = (fondo_escalado.width() - disp_escalado.width()) // 2
    y = fondo_escalado.height() - disp_escalado.height() - 20

    compuesto = QPixmap(fondo_escalado)
```

```

painter = QPainter(compuesto)
painter.drawPixmap(x, y, disp_escalado)
painter.end()

self.background.setPixmap(compuesto)
self.ventana_iniciada = True

# -----
# Action: connect/disconnect device in current room
# -----

def conectar_dispositivo(self):
    if self.imagen_actual not in self.dispositivos:
        print("No room selected. Cannot connect device.")
        return

    dispositivo = self.dispositivos[self.imagen_actual]
    estado_actual = self.estado_dispositivos[self.imagen_actual]

    if not estado_actual:
        # Connect
        self.estado_dispositivos[self.imagen_actual] = True
        self.button_dispositivo.setText(f"Desconectar
{dispositivo}")

        print(f"{dispositivo.capitalize()} in {self.imagen_actual}
connected to the outlet.")

        print(f"Components in {self.imagen_actual}: {dispositivo}
connected to the outlet.")

    else:
        # Disconnect
        self.estado_dispositivos[self.imagen_actual] = False
        self.button_dispositivo.setText(f"Conectar
{dispositivo.capitalize() }")

        print(f"{dispositivo.capitalize()} in {self.imagen_actual}
disconnected from the outlet.")

        print(f"Components in {self.imagen_actual}: {dispositivo}
disconnected.")

    # Update image (to show or hide the device)
    self.actualizar_imagen()

# -----
# Rescale when window size changes
# -----

```

```

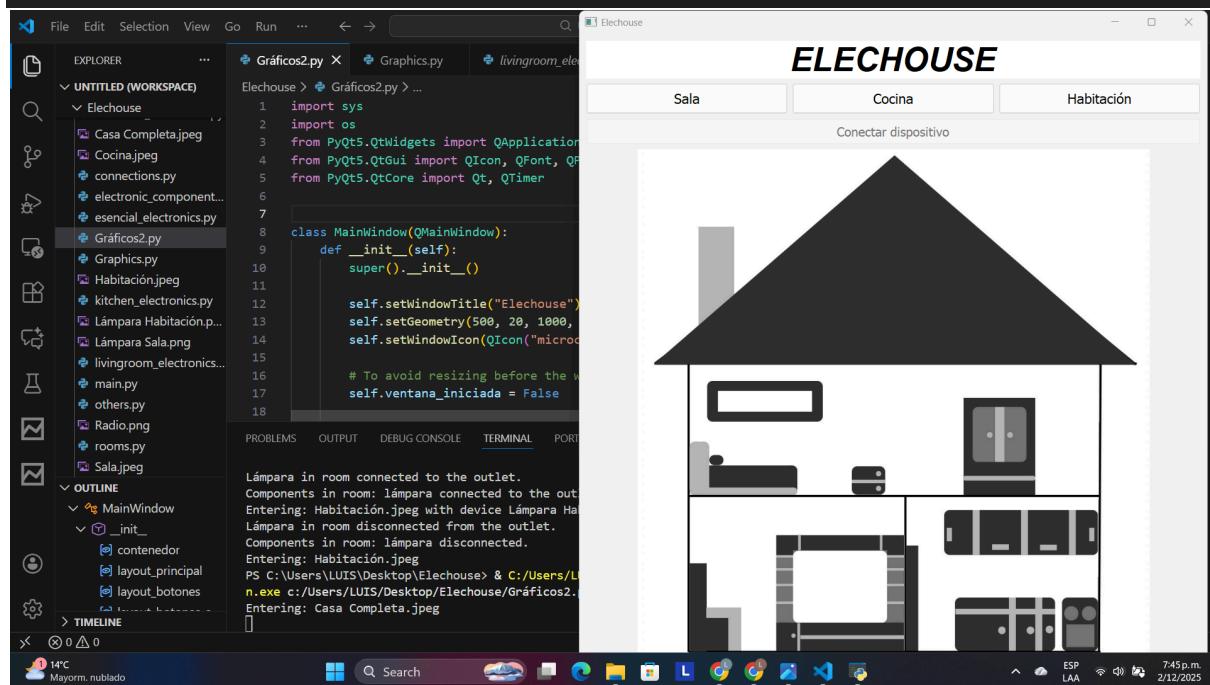
def resizeEvent(self, event):
    if self.ventana_iniciada:
        self.actualizar_imagen()

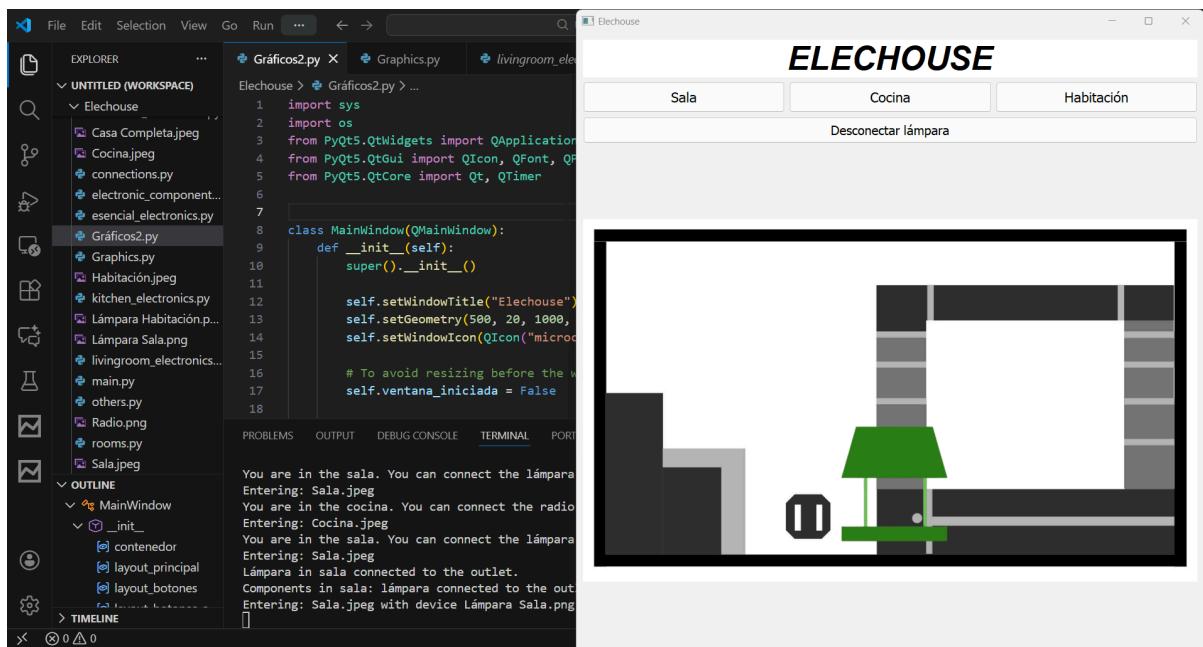
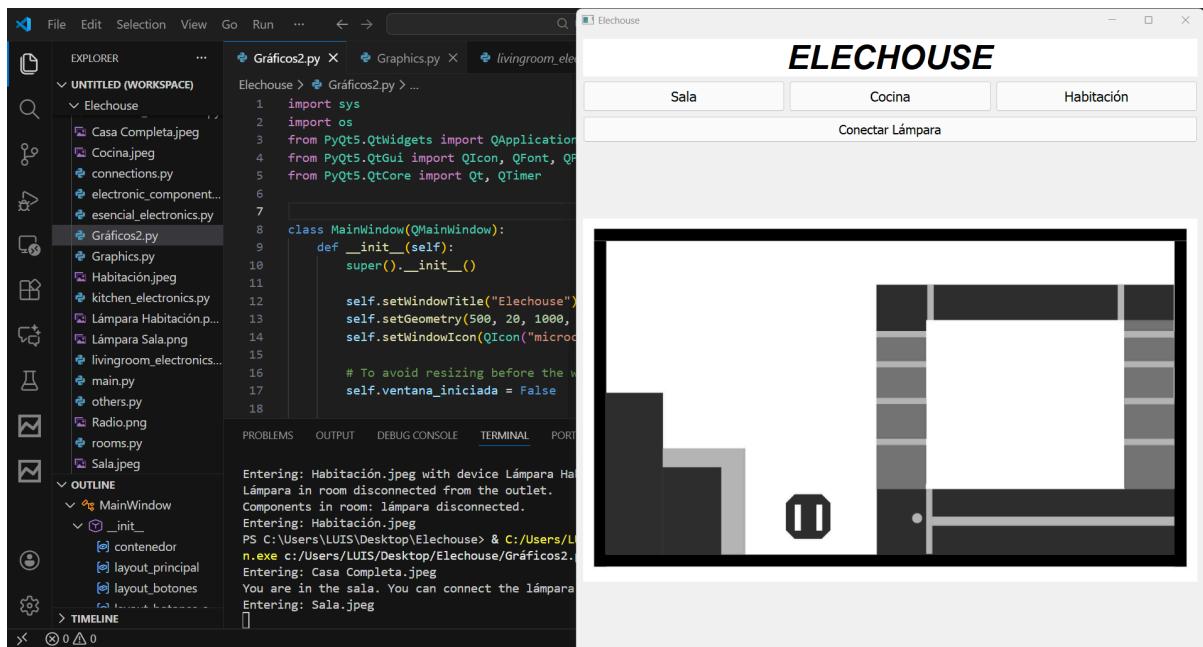
    super().resizeEvent(event)

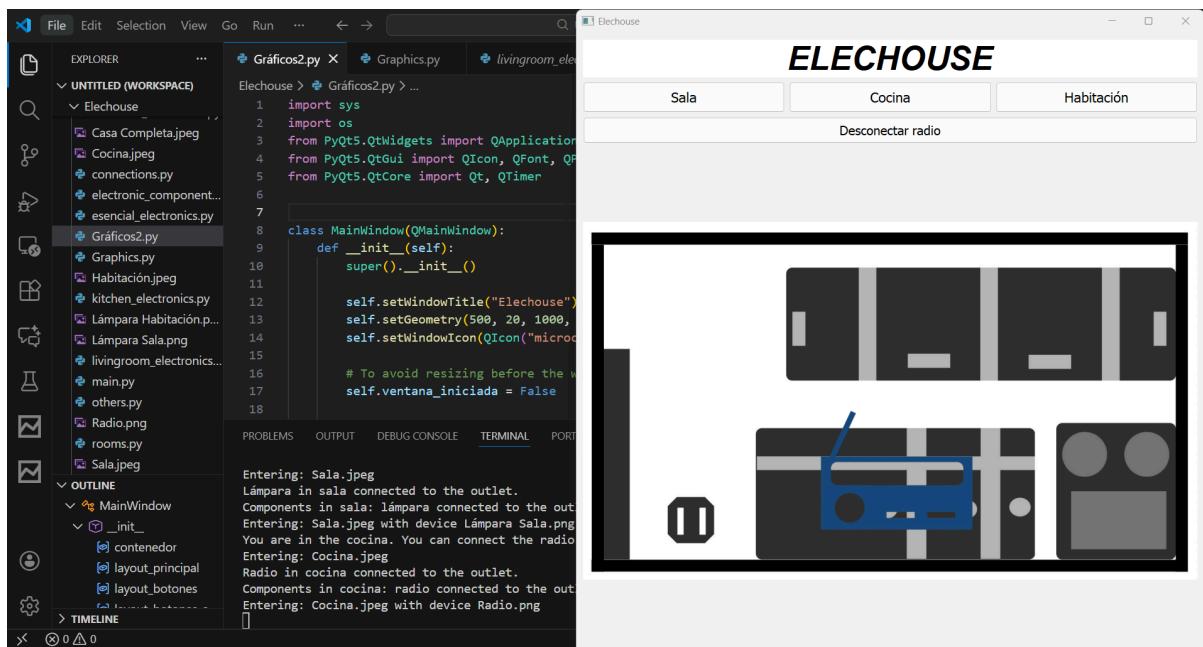
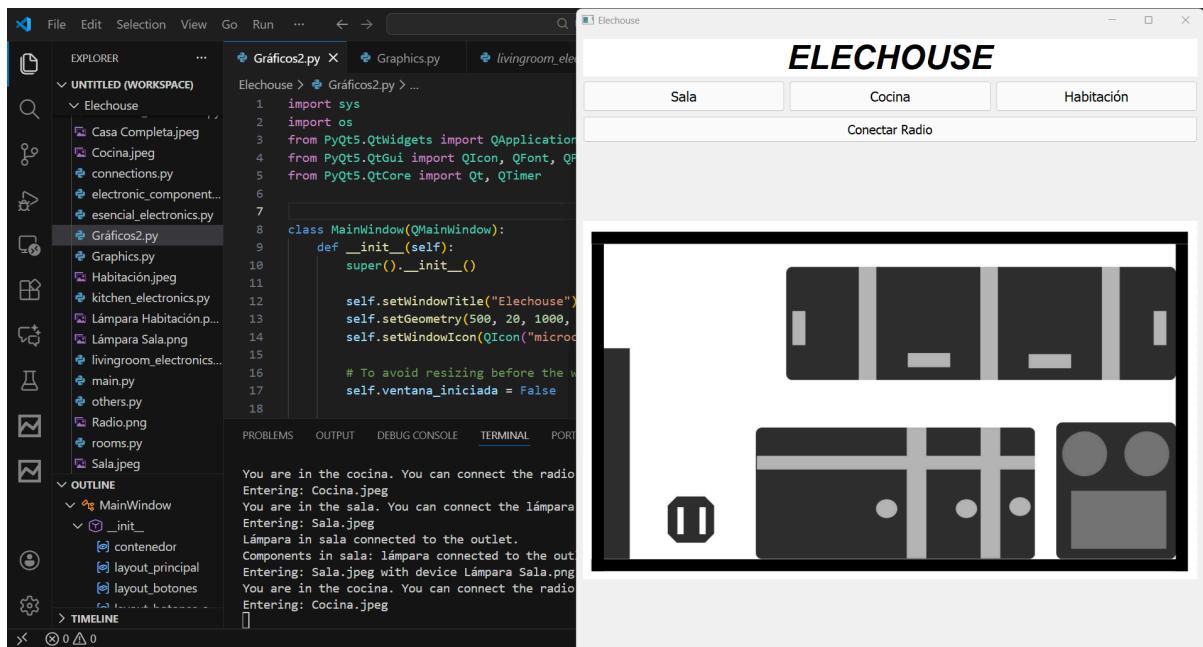
def main():
    app = QApplication(sys.argv)
    window = MainWindow()
    window.show()
    sys.exit(app.exec_())

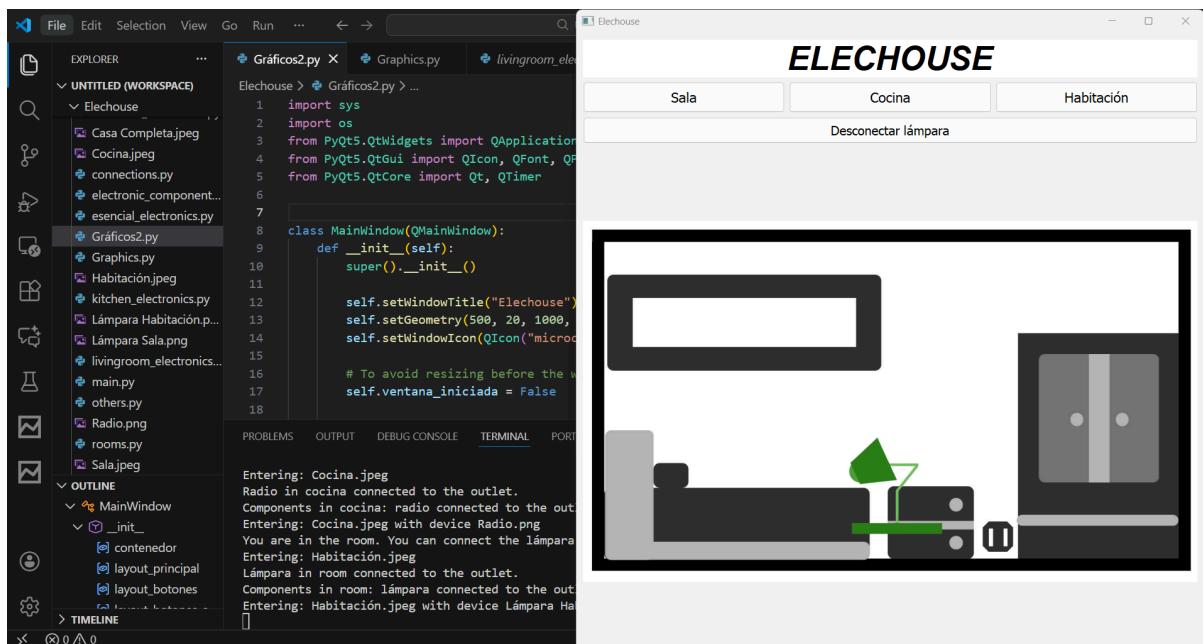
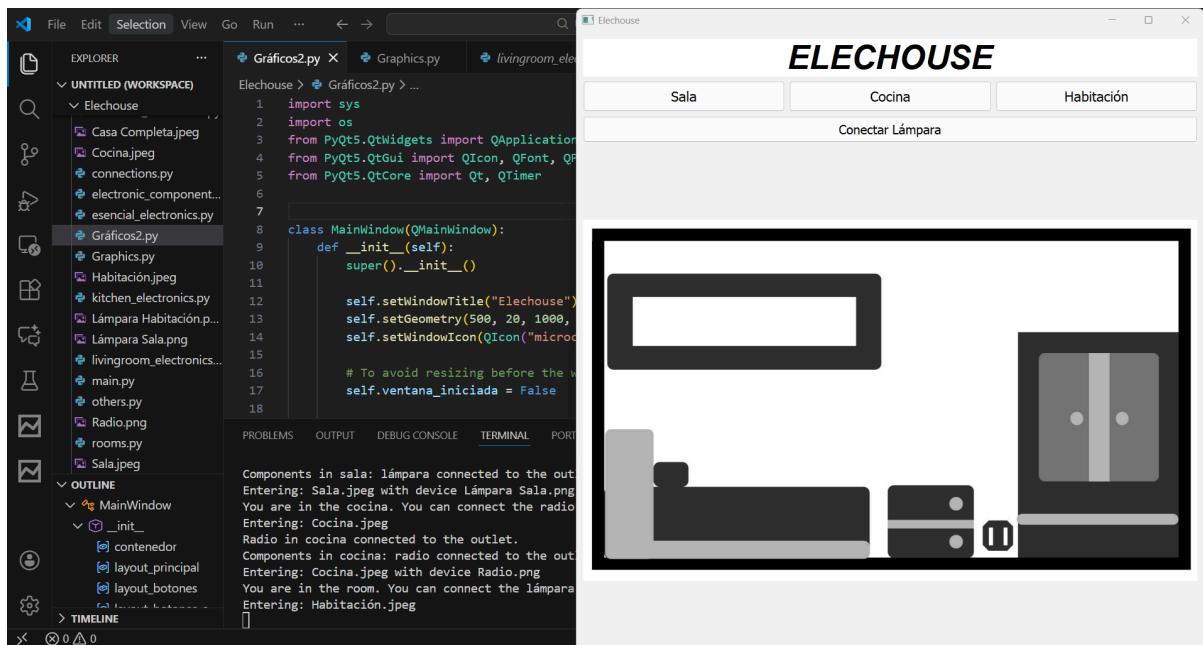
if __name__ == "__main__":
    main()

```









The program demonstrates its functionality through the console and graphically. Although not fully completed, the logic of the code is understandable, and it illustrates the messages that appear in the console, that is, it describes the actions it performs through the console.