|  |
| --- |
| An orange and white logo  Description automatically generated |



|  |  |  |  |
| --- | --- | --- | --- |
| Student Name: | Doi Pan Sin | Student No. | 202413854 |
| Instructor Name: | Engr. Dexter James Cuaresma | Raw Score: |  |
| Date: | 4/25/2025 | Percent: |  |
| Schedule: | 4/25/2025 |  |  |

**Smart Home Automation using Raspberry Pi & Tkinter Designer**

**Objective:**

To research, design, and implement a smart home automation system with sensor-actuator integration, using **Raspberry Pi**, **Figma** for GUI design, and **Tkinter Designer** for GUI conversion in Python.

**Part 1 - Introduction**

With the rapid advancement of technology, our daily lives have become more convenient and automated. Tasks that once required manual effort can now be performed effortlessly with modern solutions. For example, instead of physically going to a switch to turn lights on or off, we can now control them instantly using a smartphone.

In this project, I will design a GUI application using Figma, convert the design into Python code, and deploy it on a Raspberry Pi to control four smart appliances: lights, air conditioning, a smart door lock, and window shades. For practical demonstration, four LEDs will be used to represent these appliances.

This documentation includes: A system block diagram, GUI design using Figma, Hardware integration and Testing procedures and results.

The goal of this project is to create a user-friendly, efficient, and scalable smart home control system.

**Power Supply**

House Appliances

Raspberry Pi

Sensors

4 channel Relay

Home Control Application

Figure 1: System Block Diagram of Smart Home Automation

**Reviews**

1. Low Cost Implementation of Smart Home Automation

Researchers Ravi Kishore Kodali and Kopulwar Shishir Mahesh developed an affordable smart home system using the ESP8266 NodeMCU, known for its low power consumption. They demonstrated a straightforward setup where the ESP8266 controls a light bulb via a 4-channel relay. For remote control, they utilized MIT App Inventor 2 to create an Android application featuring both button controls and voice commands, enhancing accessibility for individuals with mobility challenges. The project is notable for its simplicity and ease of replication.

1. Smart Home Automation Using Blynk & ESP32

This IoT project, shared by *techstudycell* on *Autodesk Instructables*, provides a hands-on, step-by-step guide with a supporting YouTube video. It demonstrates how to control home appliances using an ESP32 WiFi board connected to an 8-channel relay. The maker uses the Blynk app to manage the relays and explains how to set up button widgets clearly. The instructions are simple and practical, making it a useful guide for beginners.

**Part 2 - GUI Design with Figma**

I designed two GUI layouts using Figma. The first one “*Figure 2*” is for smartphones and includes five screens: a main menu and four screens for specific appliances, each with images and ON/OFF buttons. I made this design to practice Figma’s navigation system and ensure a clean interface. However, there’s a small issue—when a user clicks the ON/OFF button, Python doesn’t show feedback like "Button is being clicked."

To fix this, I created a second design “*Figure 3”,* that’s much simpler and more user-friendly. It’s a single-screen layout meant for a Raspberry Pi monitor, with all appliance buttons on one page.

After converting the Figma design to a Python GUI using Tkinter Designer, I made a few adjustments, like updating the image file paths and repositioning some pictures. These changes can be done in the gui.py file.



Figure 2: Smart Home Automation Design with Figma for Smart Phone

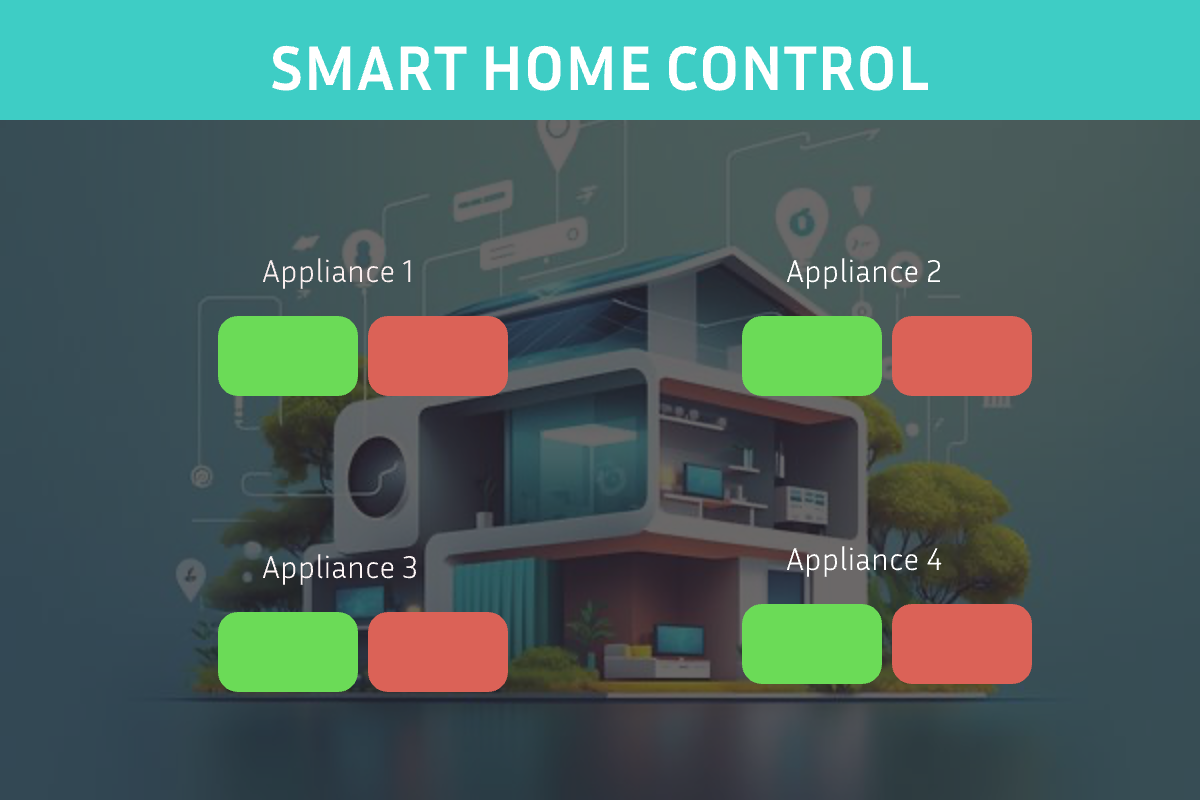


Figure 3: Smart Home Automation Design with Figma for Rpi’s Monitor

**Part 3 – Hardware Integration**

The wiring setup is straightforward. The Raspberry Pi's GPIO pins 17, 27, 22, and 23 are connected to the positive side of the LEDs, with resistors in between. The negative sides of the LEDs are all connected to the Raspberry Pi's ground pin.

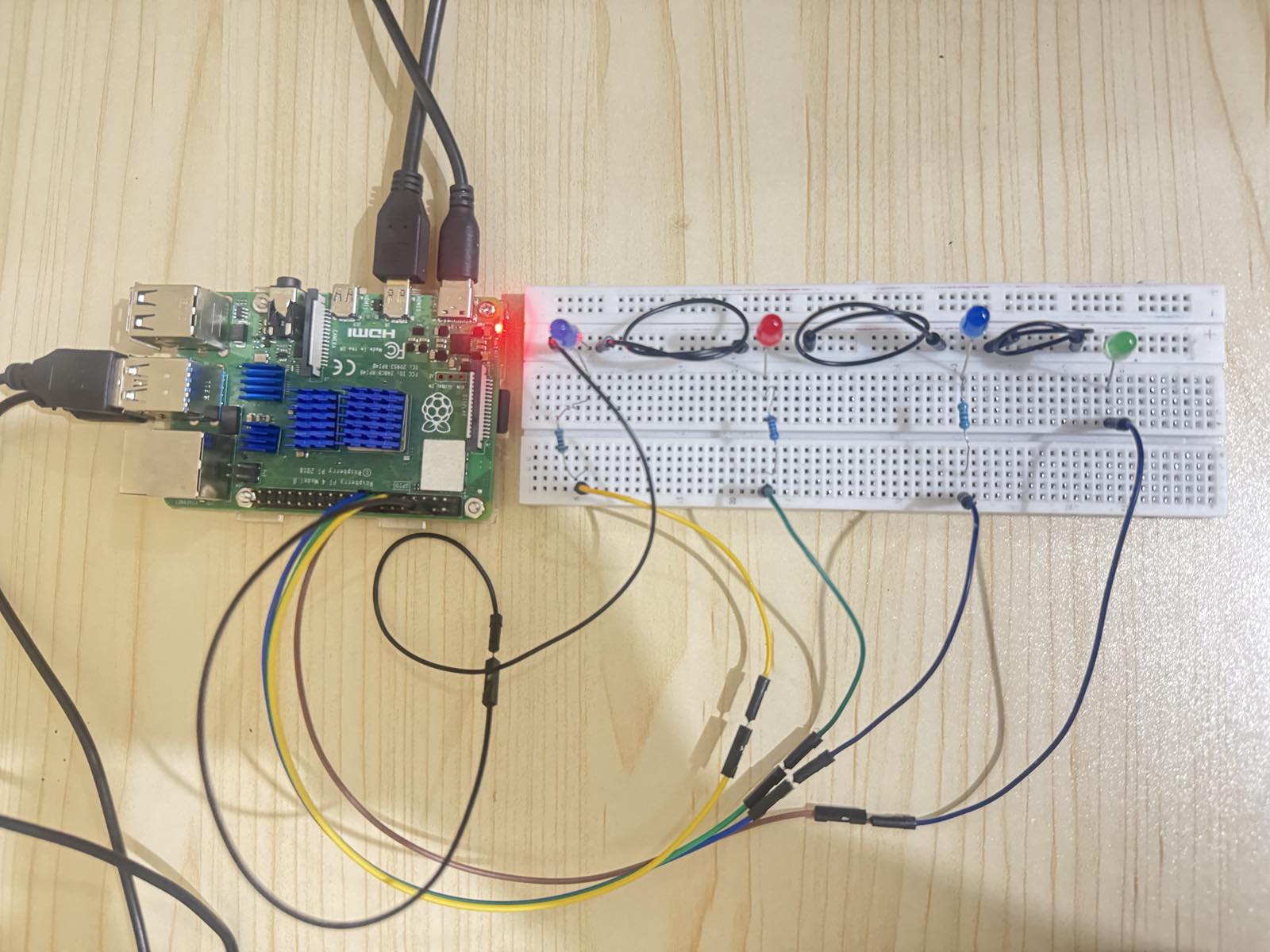


Figure 4: Wired Connection of LEDs and Raspberry Pi

Once everything was connected, all buttons and LEDs worked correctly when running the application. The second GUI design was programmed to display real-time feedback, such as "LED is ON/OFF" when a button is clicked. This way, users can confirm if the button is working, even if the LED has an issue.

**Program Modification**

We need to add a navigation feature for the first GUI layout design to work properly.   
Firstly, the required modules such as subprocess, os, and sys are imported as shown in “*Figure 5”*. Next, create a function named “open\_gui(script\_name)” to switch between screens. Then, the function is called from the each button with passing the target python file name as the following “Figure 6”. This lets users move between different app screens smoothly.

A screen shot of a computer code

Description automatically generated

Figure 5: A Python Function for File Navigation

A screen shot of a computer program

Description automatically generated

Figure 6: The Function is called from a Button

**Part 4 – Testing & Research Analysis**

When I tested the second GUI application, it worked successfully with responds. The green button is for the LED on and red button is for the LED off. The testing result with responds are shown in “*Table 1*”.

A computer screen with a blue light

Description automatically generated

Figure 7: Testing the program and circuit

|  |  |  |
| --- | --- | --- |
| Appliances | Button Clicked | Respond |
| 1 | Green | LED 1 is ON |
| 1 | Red | LED 1 is OFF |
| 2 | Green | LED 2 is ON |
| 2 | Red | LED 2 is OFF |
| 3 | Green | LED 3 is ON |
| 3 | Red | LED 3 is OFF |
| 4 | Green | LED 4 is ON |
| 4 | Red | LED 4 is OFF |

Table 1: Testing Result with Respond

Testing the GUI layout design 1 is successful. But it can not show real time respond, because there is a problem while calling the function for both GPIO output and print.

A computer screen with a black rectangular object on it

Description automatically generated

Figure 8: Testing GUI layout 1

A computer screen with a light bulb on it

Description automatically generated

Figure 9: Testing Appliance 1 or “Light”

A computer screen with a blue light

Description automatically generated

Figure 10: Testing Appliance 2 or “Air Condition”

A computer screen with a computer monitor and wires

Description automatically generated with medium confidence

Figure 11: Testing Appliance 3 or “TV”

A computer screen with a computer program

Description automatically generated with medium confidence

Figure 12: Testing Appliance 4 or “Door Lock”

**Key Learning for Converting Figma Designs to Python GUIs**

1. Name all components clearly (e.g., "Button\_1", "Image\_LED").
2. Avoid grouping components: It can cause errors in recognizing buttons/images.
3. Expect minor visual differences: Some manual adjusting may be needed in the Python code.

**Future Improvements**

As highlighted in Review Paper 1, I aim to enhance this smart home automation project by integrating voice control capabilities. This addition will allow users to operate appliances through voice commands, offering a more intuitive and hands-free experience.

To enhance the smart home automation system, integrating various sensors can significantly improve its intelligence and responsiveness. For instance, combining temperature sensors with motion detectors allows the system to automatically activate the air conditioning when high temperatures are detected in occupied rooms. Similarly, motion sensors can turn off lights in unoccupied areas, conserving energy. Incorporating security features, such as cameras with facial recognition, can further bolster the system. These cameras can identify unfamiliar individuals and, when the homeowners are away, send real-time alerts, enhancing home security.

By integrating these sensors and technologies, the smart home system becomes more efficient, responsive, and secure, offering a seamless and intelligent living experience.

**References**

1. Kodali, R. K., & Mahesh, K. S. (2017). Low cost implementation of smart home automation. *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, 461–466. https://doi.org/10.1109/icacci.2017.8125883
2. Techstudycell, & Instructables. (2021, February 2). *Smart Home Automation using Blynk & ESP32 IOT Projects: WIFI & Manual*. Instructables. https://www.instructables.com/Smart-Home-Automation-Using-Blynk-ESP32-IoT-Projec/