



# **Advanced GIPO Sniffing with AWS Communication**

**By Myles Daniels 2104397**

CMP408: IOT and Cloud Secure Development

2024/25

*Note that Information contained in this document is for educational purposes*

# Contents

---

1	Introduction .....	1
1.1	Background.....	1
1.2	Aim.....	1
2	Procedure.....	2
2.1	Overview of Procedure.....	2
3	Conclusion.....	3
3.1	Conclusion .....	3
3.2	Future Work.....	3
	References .....	4
	Appendices.....	5
	Appendix A picture of the circuit setup (Raspberry pi is out of shot) .....	5
	Appendix B Circuit Diagram .....	5
	Appendix C: 1 <sup>st</sup> Version of the GPIO Sniffer.....	6
	Appendix D: dynamo DB database setup .....	7
	Appendix E: final GPIO sniffer script.....	7

# 1 INTRODUCTION

## 1.1 BACKGROUND

---

Raspberry Pis have been a useful tool for an introduction to the learning environment however they can be used in a variety of other ways over the years there simple construction, guides and use in educational environments have given rise to their popularity over the last 10 years especially for their role in creating simple rudimentary circuits to teach students about programming and simple electrical circuitry and whilst most Pis are relatively low security by default they come with basic password protection and are mainly marketed towards computer beginners it can be a useful case study into some of the dangers that may befall users of such devices if they are left unsecured. [8][9]

Therefore, by the end of this project/investigation it can be shown that these devices can be controlled and used to extract data from them which can be simulated as a malicious extraction. The reason that this project in particular was decided on was that it will throw the doors wide open on the possibility that it can entail including but not limited to manipulation of sensors and servos, speakers and read incoming information especially if such information is related to passwords and other sensitive data and the use in smart homes.

## 1.2 AIM

---

The aims of this project are very simple and reflect both the simplicity of the circuit, the devices being used and the overall goal of the project these are in order:

- To successfully take control of the LEDS connected to the raspberry PI and successfully manipulate them in any form.
- To successfully initiate a cloud computing connection between a cloud source and the Rpi
- To be able to transmit data to this cloud site and be able to convert it into a database.

## 2 PROCEDURE

### 2.1 OVERVIEW OF PROCEDURE

---

The first step was testing the Pi to make sure that it was capable of performing the basic level of this tasks by way of the Led circuit whilst making sure that all components were working correctly however there was an issue in that the pi in its current form could only handle one LED at a time anymore would exceed the power limit for the device even though it was powered by mains electricity through an extension lead with USB slots, this obstacle was overcome successfully and the main circuit was tested and connected it up to GPIO pin 20, the issue was solved by increasing resistor capacity (see appendix a and b). With this issue passed the next challenge was to connect the pi up to my computer via ssh or Secure Shell protocol which is mainly used by network administrators to remote access devices across a network and manage them remotely by using a client-server model for effective communication.<sup>[3][4]</sup> In particular this project is using a program called PUTTY which is a free open source ssh emulator<sup>[5]</sup>, through this program A direct SSH connection was set up between my Windows Device (laptop) and the Raspberry Pi itself through which A console existed where commands could be entered and executed on the rPi including python scripts which is vital for this project to demonstrate that the rpi could be manipulated a script was developed that would search all the open ports on the rpi in order to find the one that was connected to the circuit, when activated it would then move into the second stage off the plan in which it would allow the script controller to input a flashing sequence for the pi with time between flashes and a number of iterations this first form of the script was created to prove the raspberry could be exploited if left with simple security settings such as default passwords which it was reset back to.<sup>(see appendix C)</sup>

The next task is the cloud element of the project for this we will be using the AWS Learner Lab platform, which is an educational platform that allows students to learn about cloud computing<sup>[6]</sup>, the next step was to create a database within this platform to create a simple database that has the GPIO pin used and a timestamp showing that it received a signal from the rpi itself, satisfied with this the next issue was needing to set up the connection from the pi to the database, through the PUTTY connection was able to install the required files necessary to complete the connection which were AWS CLI and boto3, then manually configured the connection between the rpi and the AWS DynamoDB database<sup>[7]</sup> using the necessary keys and session token provided was able after difficulties to establish a stable connection to the database and modified the script to send out alerts when an open GPIO port was found through testing this script and checking the corresponding table to see a total of 51 entries with several other tests further confirming a direct and successful transmission of data between the dynamo database and the rpi therefore completing the project and showing that data can be extracted from a compromised pi to a third party cloud platform.<sup>(see appendix's D and E)</sup>

## 3 CONCLUSION

### 3.1 CONCLUSION

---

With this mini project complete it was found to be quite invigorating and was able to demonstrate the initial proposal through a simple rpi and cloud setup that did not affect my budget in any major significant manor which is good for myself personally as cost was a major factor in my choice of project as a project that would not infringe upon that budget mainly due to the fear that it would breach the financial limit and cause a major setback. Additionally this project was a foundation of hopeful future research into how RPIs can be breached and exploited, an example that was found whilst undertaking research for this project is a NASA breach using a RPI to download 500mb of classified data [8].

During the course of this project several technical issues of the usual verity were encountered including but not limited to password failures, AWS configuration errors and issues related to the setup of PUTTY on both windows and RPI mainly due to location issues (i.e lack of access to a monitor) however these were overcome thanks to a number of techniques including the program PUTTY which was essential to the project's completion, overall this mini project has been a resounding success and hopefully paves the way for future research projects into how RPI's can be affected by malicious actors and hopefully serve as a warning to those who continue to use these devices on their default settings hoping for more stringent security measures in the future.

### 3.2 FUTURE WORK

---

With more time and resources especially with an increased AWS budget there would be improvements on the circuit to possibly add additional devices for testing onto the breadboard including but not limited to LED, showing messages that the malicious actors would be able to put on the unsecured device also I would add in speakers in order to exploit them like the led screens whilst with the testers experience with this is in a Wemos setting<sup>[10]</sup>, such a scenario setup would not be a huge challenge. also the python script would be expanded to add in a password cracker if there is a second line of defence that this scenario is not accounting for (this scenario works on the basis of default settings) and finally the DynamoDB database would be expanded to account for more details about the pins accessed, what it is connected to and device types among other things.

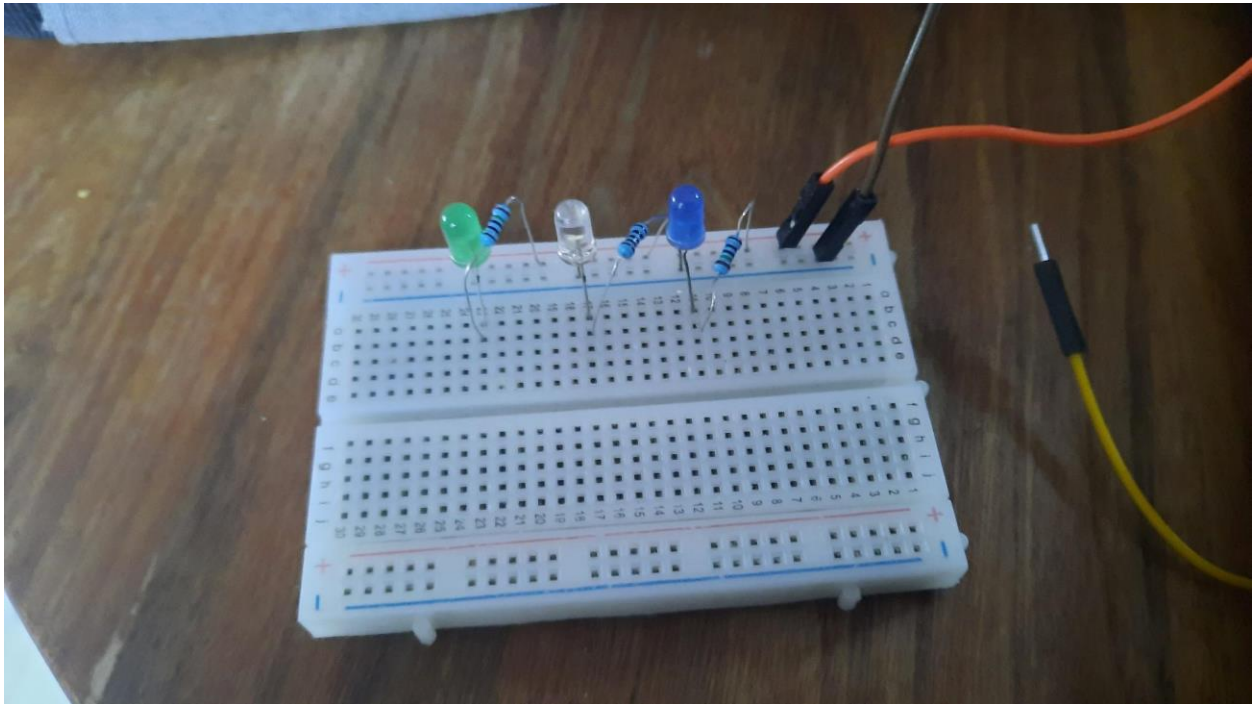
# REFERENCES

For URLs, Blogs:

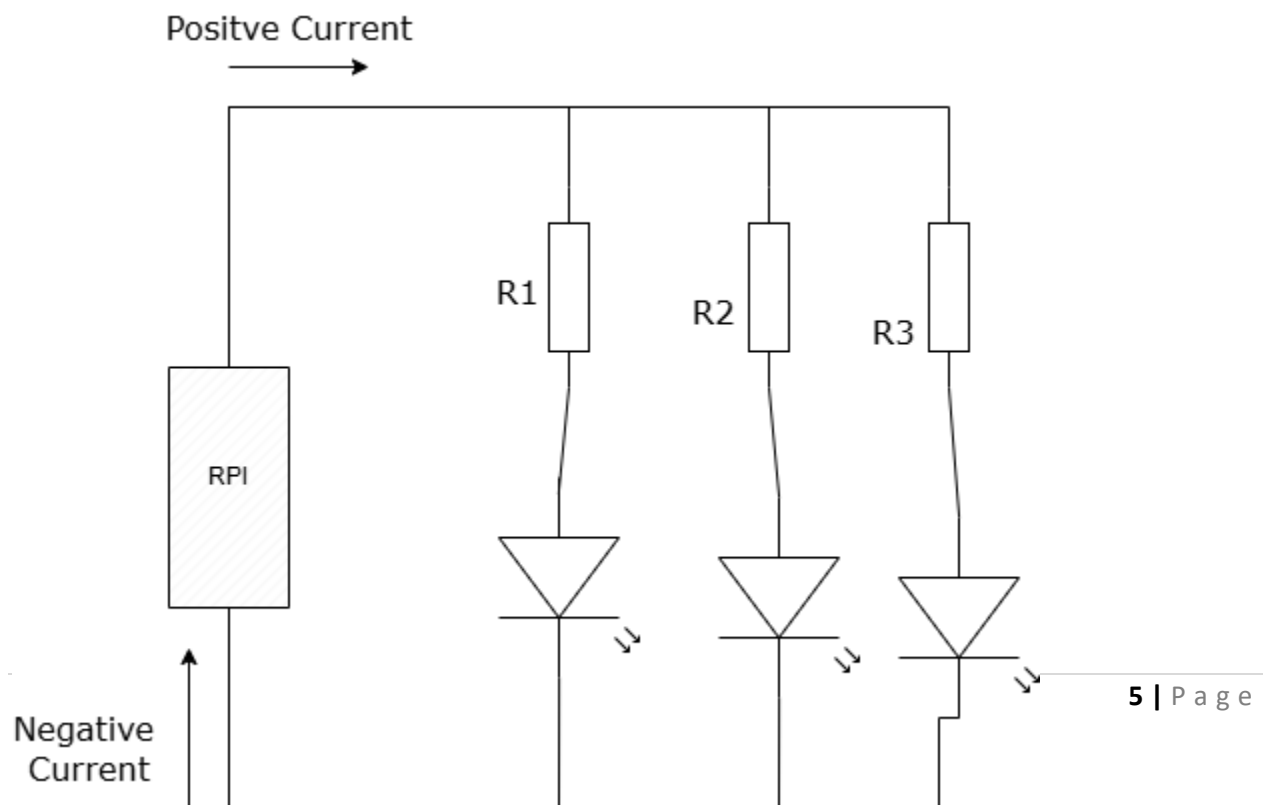
1. **Diva Portal.** (Abbas, Mazhar.) *Wireless Network Security Using Raspberry Pi*. Available at: <https://www.diva-portal.org/smash/record.jsf?pid=diva2%3A1909496&dswid=5286> (Accessed: 11 December 2024).
2. **Cruz de la Cruz, J.E., Romero Goyzueta, C.A., and Delgado Cahuana, C.** (2020) *Wireless Network Security Using Raspberry Pi*. Universidad Nacional del Altiplano, Puno, Perú, 03–05 September. Available at: <https://ieeexplore.ieee.org/abstract/document/9220240> (Accessed: 11 December 2024).
3. **Gillis, A.S., Loshin, P., and Cobb, M.** (n.d.) *What is SSH (Secure Shell) and How Does It Work?* Available at: <https://www.techtarget.com/searchsecurity/definition/Secure-Shell> (Accessed: 11 December 2024).
4. **TechTarget Contributor.** (n.d.) *What is the Client-Server Model?* Available at: <https://www.techtarget.com/searchnetworking/definition/client-server> (Accessed: 11 December 2024).
5. **SSH Communications Security.** (n.d.) *PuTTY: Free SSH and Telnet Client*. Available at: <https://www.ssh.com/academy/ssh/putty> (Accessed: 11 December 2024).
6. **Amazon Web Services (AWS).** (n.d.) *AWS Academy: Empowering Higher Education Institutions*. Available at: <https://aws.amazon.com/training/awsacademy/> (Accessed: 11 December 2024).
7. **Amazon Web Services (AWS).** (n.d.) *Amazon DynamoDB – Managed NoSQL Database Service*. Available at: <https://aws.amazon.com/pm/dynamodb/> (Accessed: 11 December 2024).
8. **BBC News.** (2019) *Malicious Hacker Won Access to Data in Security Breach*. Available at: <https://www.bbc.co.uk/news/technology-48743043> (Accessed: 11 December 2024).
9. **Toulas, B.** (2024) *\$700 Cybercrime Software Turns Raspberry Pi into an Evasive Fraud Tool*. BleepingComputer, 26 March. Available at: <https://www.bleepingcomputer.com/news/security/700-cybercrime-software-turns-raspberry-pi-into-an-evasive-fraud-tool/> (Accessed: 11 December 2024).
10. **Mybotic.** (n.d.) *How to Use Arduino WeMos D1 WiFi UNO ESP8266 IoT IDE Compatible Board by Using Blynk*. Available at: <https://www.instructables.com/Arduino-WeMos-D1-WiFi-UNO-ESP-8266-IoT-IDE-Compati/> (Accessed: 11 December 2024).

## APPENDICES

### APPENDIX A PICTURE OF THE CIRCUIT SETUP (RASPBERRY PI IS OUT OF SHOT)



### APPENDIX B CIRCUIT DIAGRAM



## APPENDIX C: 1<sup>ST</sup> VERSION OF THE GPIO SNIFFER

---

```
1 import RPi.GPIO as GPIO
2 import time
3
4 # Function to check if a GPIO pin is free
5 def is_pin_free(pin):
6     try:
7         GPIO.setup(pin, GPIO.OUT)
8         GPIO.output(pin, GPIO.LOW)
9         GPIO.output(pin, GPIO.HIGH)
10        GPIO.output(pin, GPIO.LOW)
11        return True
12    except Exception as e:
13        print(f"Error testing GPIO pin {pin}: {e}")
14        return False
15    finally:
16        GPIO.cleanup(pin)
17
18 # Function to flash an LED
19 def flash_led(pin, flash_time=1, cycles=10):
20     GPIO.setmode(GPIO.BCM) # Use Broadcom GPIO numbering
21     GPIO.setup(pin, GPIO.OUT)
22
23     print(f"Flashing LED connected to GPIO pin {pin}.")
24     try:
25         for i in range(cycles):
26             GPIO.output(pin, GPIO.HIGH) # LED ON
27             time.sleep(flash_time) # Wait for 'flash_time' seconds
28             GPIO.output(pin, GPIO.LOW) # LED OFF
29             time.sleep(flash_time) # Wait for 'flash_time' seconds
30             print(f"Flash {i + 1}/{cycles}")
31     except KeyboardInterrupt:
32         print("Interrupted by user. Cleaning up...")
33     finally:
34         GPIO.output(pin, GPIO.LOW) # Ensure the LED is OFF
35
36     GPIO.cleanup() # Reset GPIO settings
37     print("Cleanup complete.")
38
39 # Main script to find open GPIO pins and offer flashing
40 if __name__ == "__main__":
41     GPIO.setmode(GPIO.BCM) # Use Broadcom numbering
42     available_pins = []
43
44     try:
45         print("Scanning GPIO pins...")
46         for pin in range(2, 28): # Test GPIO pins 2 through 27
47             print(f"Testing GPIO pin {pin}...")
48             if is_pin_free(pin):
49                 print(f"GPIO pin {pin} is free!")
50                 available_pins.append(pin)
51             else:
52                 print(f"GPIO pin {pin} is not free.")
53
54     if not available_pins:
55         print("No free GPIO pins found!")
56     else:
57         print(f"Available GPIO pins: {available_pins}")
58
59         # Ask the user if they want to flash an LED on a free pin
60         flash_pin = int(input("Enter a free GPIO pin number to flash an LED, or 0 to exit: "))
61         if flash_pin in available_pins:
62             flash_time = float(input("Enter flash time in seconds (e.g., 0.5 for half a second): "))
63             cycles = int(input("Enter the number of flash cycles: "))
64             flash_led(flash_pin, flash_time, cycles)
65         elif flash_pin == 0:
66             print("Exiting without flashing.")
67         else:
68             print("Invalid pin selected. Exiting.")
```

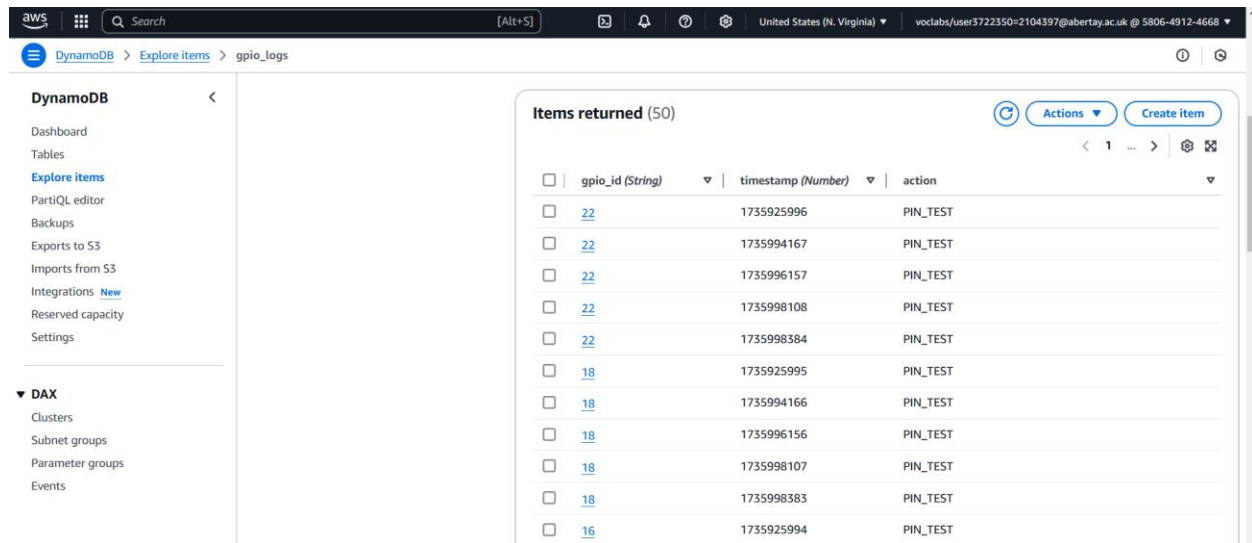


```

67         print("Invalid pin selected. Exiting.")
68     finally:
69         GPIO.cleanup()
70         print("GPIO cleanup complete.")
71
72

```

## APPENDIX D: DYNAMO DB DATABASE SETUP



The screenshot shows the AWS DynamoDB console interface. On the left is a navigation menu with options like Dashboard, Tables, Explore items, PartIQ editor, Backups, Exports to S3, Imports from S3, Integrations, Reserved capacity, and Settings. The main area displays the 'gpio\_logs' table. A table of items is shown with columns for 'gpio\_id (String)', 'timestamp (Number)', and 'action'. The items are listed in a table with checkboxes for each row.

gpio_id (String)	timestamp (Number)	action
22	1735925996	PIN_TEST
22	1735994167	PIN_TEST
22	1735996157	PIN_TEST
22	1735998108	PIN_TEST
22	1735998384	PIN_TEST
18	1735925995	PIN_TEST
18	1735994166	PIN_TEST
18	1735996156	PIN_TEST
18	1735998107	PIN_TEST
18	1735998383	PIN_TEST
16	1735925994	PIN_TEST

## APPENDIX E: FINAL GPIO SNIFFER SCRIPT

```

C:\Users > MDROD > OneDrive > Documents > MylesDaniels-2104397 > gpio_sniffer_v2.py > log_to_dynamodb
1  import RPi.GPIO as GPIO
2  import time
3  import boto3
4  from botocore.exceptions import NoCredentialsError, ClientError
5
6  # AWS Configuration
7  REGION_NAME = "us-east-1" # Update with your AWS region
8  TABLE_NAME = "gpio_logs" # Ensure this matches your DynamoDB table name
9
10 # Initialize DynamoDB Client
11 dynamodb = boto3.client('dynamodb', region_name=REGION_NAME)
12
13 # Function to log data to DynamoDB
14 def log_to_dynamodb(gpio_id, action):
15     timestamp = str(int(time.time())) # Current UNIX timestamp
16
17     try:
18         response = dynamodb.put_item(
19             TableName=TABLE_NAME,
20             Item={
21                 'gpio_id': {'S': str(gpio_id)}, # Primary Key
22                 'timestamp': {'N': timestamp}, # Numeric timestamp
23                 'action': {'S': action} # "ON" or "OFF"
24             }
25         )
26         print(f"Logged to DynamoDB: GPIO {gpio_id} - {action}")
27     except NoCredentialsError:
28         print("AWS credentials not found. Ensure 'aws configure' is set up.")
29     except ClientError as e:
30         print(f"AWS Error: {e.response['Error']['Message']}")
31
32 # Function to check if a GPIO pin is free
33 def is_pin_free(pin):
34     try:
35         GPIO.setup(pin, GPIO.OUT)

```

```

35 def is_pin_free(pin):
36     GPIO.output(pin, GPIO.LOW)
37     GPIO.output(pin, GPIO.HIGH)
38     GPIO.output(pin, GPIO.LOW)
39     log_to_dynamodb(pin, "PIN_TEST") # Log to AWS
40     return True
41 except Exception as e:
42     print(f" Error testing GPIO pin {pin}: {e}")
43     log_to_dynamodb(pin, "PIN_TEST_FAILED")
44     return False
45 finally:
46     GPIO.cleanup(pin)
47
48 # Function to flash an LED
49 def flash_led(pin, flash_time=1, cycles=10):
50     GPIO.setmode(GPIO.BCM) # Use Broadcom GPIO numbering
51     GPIO.setup(pin, GPIO.OUT)
52
53     print(f" Flashing LED on GPIO {pin}")
54     try:
55         for i in range(cycles):
56             GPIO.output(pin, GPIO.HIGH) # LED ON
57             log_to_dynamodb(pin, "ON") # Log ON event
58             time.sleep(flash_time)
59
60             GPIO.output(pin, GPIO.LOW) # LED OFF
61             log_to_dynamodb(pin, "OFF") # Log OFF event
62             time.sleep(flash_time)
63
64             print(f"Flash {i + 1}/{cycles}")
65     except KeyboardInterrupt:
66         print(" Interrupted by user.")
67     finally:
68         GPIO.output(pin, GPIO.LOW) # Ensure LED is OFF
69         GPIO.cleanup()
70     print(" Cleanup complete.")

```

```

68     GPIO.output(pin, GPIO.LOW) # Ensure LED is OFF
69     GPIO.cleanup()
70     print(" Cleanup complete.")
71
72 # Main script to find open GPIO pins and offer flashing
73 if __name__ == "__main__":
74     GPIO.setmode(GPIO.BCM) # Use Broadcom numbering
75     available_pins = []
76
77     try:
78         print(" Scanning GPIO pins...")
79         for pin in range(2, 28): # Test GPIO pins 2 through 27
80             print(f" Testing GPIO pin {pin}...")
81             if is_pin_free(pin):
82                 print(f" GPIO pin {pin} is free!")
83                 available_pins.append(pin)
84             else:
85                 print(f" GPIO pin {pin} is NOT free.")
86
87     if not available_pins:
88         print(" No free GPIO pins found!")
89     else:
90         print(f"Available GPIO pins: {available_pins}")
91
92     # Ask user to flash LED on a free pin
93     flash_pin = int(input(" Enter a free GPIO pin to flash an LED (or 0 to exit): "))
94     if flash_pin in available_pins:
95         flash_time = float(input(" Enter flash time (e.g., 0.5 seconds): "))
96         cycles = int(input(" Enter number of flash cycles: "))
97         flash_led(flash_pin, flash_time, cycles)
98     elif flash_pin == 0:
99         print(" Exiting without flashing.")
100     else:
101         print("Invalid pin selected. Exiting.")

```

```

101         print("Invalid pin selected. Exiting.")
102
103     finally:
104         GPIO.cleanup()
105         print(" GPIO cleanup complete.")
106
107

```