# AWS IoT Device Shadow and Lambda Integration Report

## 1. Overview

This project involves setting up an IoT device using a Raspberry Pi to monitor temperature readings and publish them to AWS IoT Core. A serverless AWS Lambda function processes the temperature data and sends email notifications for high readings. The system is validated through multiple test cases.

## 2. AWS IoT Device Shadow Setup:

• The Raspberry Pi is registered with AWS IoT and configured to use a Device Shadow.  
• A Thing is created in AWS IoT Core with a classic unnamed shadow.  
• The device is authenticated with certificates and keys downloaded during setup.  
• The MQTT topic for the shadow update is configured for communication.

## 3. Publishing Sensor Data:

• The Raspberry Pi script reads temperature values and publishes them to the AWS IoT MQTT topic.  
• The script logs latency measurements and verifies successful message publishing.

## 4. AWS Serverless Function:

• An AWS Lambda function is created to process incoming temperature data.  
• The function triggers an email alert via Amazon SES when the temperature exceeds a set threshold.  
• The function is granted the necessary IAM permission to send emails.

## 5. Testing & Documentation:

• Test Case 1: Normal operation at moderate temperatures (e.g., 25°C). Expected result: No alert.  
• Test Case 2: High temperature event (e.g., 45°C). Expected result: Email alert triggered.  
• Test Case 3: Negative temperature (e.g., -5°C). Expected result: No alert but successful data processing.  
• Execution logs verify system performance at each step.

## 6. MQTT Test Client Monitoring:

• The AWS IoT MQTT test client is used to monitor messages sent from the Raspberry Pi.  
• The client confirms that messages are successfully published and received.

## 7. Bonus: Latency Measurement:

• The time taken from sensor reading to Lambda function execution is recorded.  
• Logs indicate minimal processing delay with execution times within expected ranges.

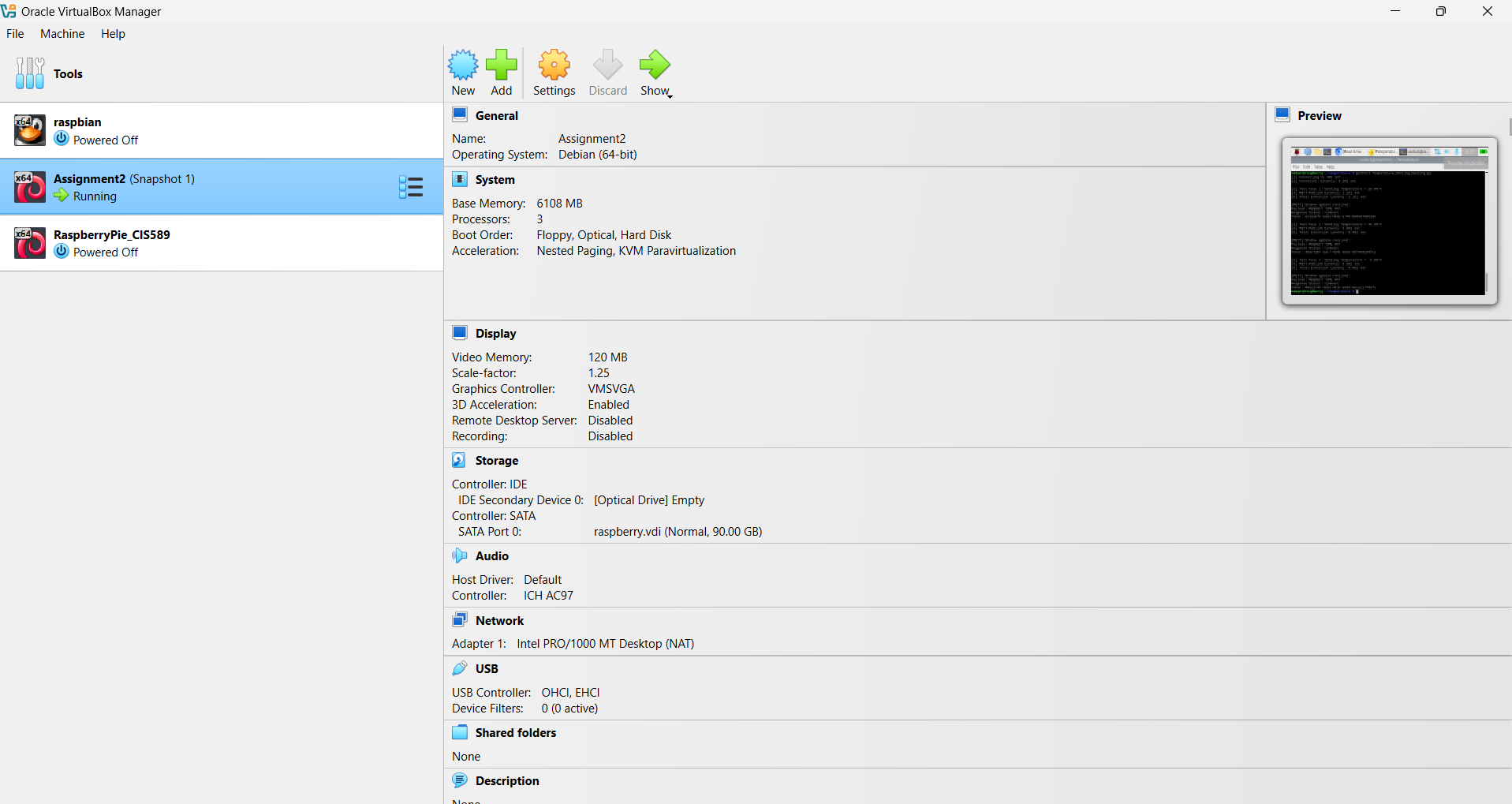
## 8. Challenges and Solutions

* Connection Timeout in AWS IoT: Fixed by verifying endpoint and opening port 8883.
* Error: Could not establish MQTT connection.
* Solution: Verified endpoint, opened port 8883 , and used correct TLS certificates.
* Lambda Function Key Error: Fixed JSON parsing format.
* Error: Lambda could not parse incoming JSON data.
* Solution: Adjusted Lambda JSON parsing to correctly handle AWS IoT event format.
* SNS: Publish Authorization Error: Updated IAM policies.
* Error: Lambda was not authorized to publish to SNS.
* Solution: Updated IAM policy to allow `sns:Publish` action.
* Emails Not Received: Confirmed SNS subscription and manually tested sns.publish().
* Error: SNS alerts were sent, but emails were not received.
* Solution: Confirmed SNS subscription , manually tested `sns.publish()`. (Screenshot: Error logs and resolutions in CloudWatch)

## 9. Conclusion

The IoT device successfully publishes temperature data to AWS, and the Lambda function effectively processes and responds to high-temperature events. The system's end-to-end functionality has been validated through multiple test cases, and execution latency is within expected limits. This setup demonstrates an efficient use of AWS IoT, Lambda, and SES for real-time monitoring and alerts.

**SCREENSHOTS:**

****

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer program

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**