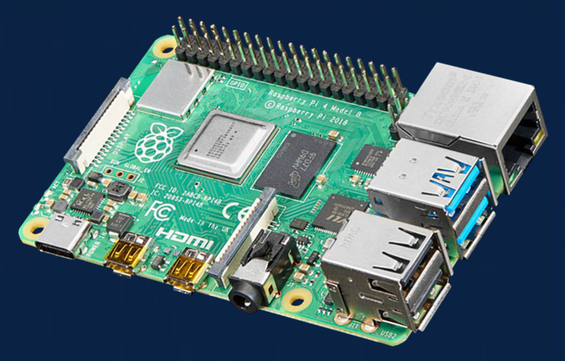
Florida International University

Spring 2022 Senior Design Project

Network Outage Call Home



Students: Michael Beall

Mentor: *Ryan Hutcheson*, *Product Owner (Kaseya)*

Instructor/Faculty: *Dr. Masoud Sadjadi*, Florida International University

Knight Foundation School of Computing and Information Sciences

PROBLEM

- As an MSP owner, I need visibility into my environments even when internet issues occur.
- Build and connect a small device (Raspberry Pi) to a network that upon losing connection will phone home to the user over a cellular network.
- The end user should then be able to remotely access this device over a cellular connection.

CURRENT SYSTEM

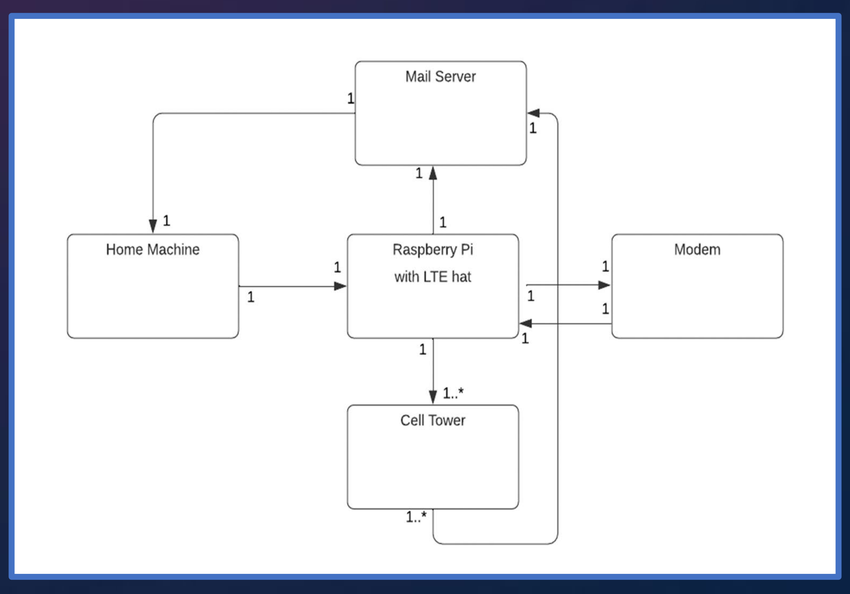
- In the case of an outage there is currently no automated system in place to notify technicians of down time.
- When an issue does arise, it would be up to someone on site to notice the issue and then contact technicians accordingly.

REQUIREMENTS

Hardware:
- Arduino R3 (Early Concept)
- Raspberry Pi 3 (Current Concept)
- Sixfab 4G/LTE Base Hat
- Sixfab IoT Simcard

Software:
- Python 3.10.4
- Visual Studio Code
- Raspberry Pi OS
- Raspberry Pi Terminal
- Sixfab Dashboard

SYSTEM DESIGN



The above diagram represents the layout of a successfully deployed device

SAMPLE CODE

```python
import os
import time
import phone_home

def eth_ping():
    # Recipient of ping test
    hostname = "8.8.8.8" # 8.8.8.8 = Google DNS
    # Specifies interface to conduct ping from
    interface = "wlan0"
    # This will run the linux ping command from the wlan0 interface
    response = os.system("ping -c 4 -I " + interface + " " + hostname)

    # If a response of 0 is received than the ping was successful and the network in up
    if response == 0:
        status = 'Up'
    # Else the ping has failed and the phone_home function will be called.
    # The phone home function will then alert the end user of the outage.
    else:
        status = 'Down'
        phone_home.phone_home()
    return status

# Function called in the run.py script to automate this script
def run():
    while(True):
        # Prints the results of the script for debugging purposes
        print(eth_ping())
        # As the script is automated, this will set a delay to run every half hour
        time.sleep(3000)
```

The above script showcases the code behind the primary function of the device

IMPLEMENTATION

- Python 3 - Primary Scripting language used for device tests/alerts
- Sixfab Dashboard - Used to configure network interface metrics
- Trello - Service used for tracking the development of user stories.
- Linux Terminal - Necessary for running test scripts and detecting outages.
- Raspberry Pi OS - The operating system deployed on the device.
- VNC Viewer - Software that allowed multiple users to access and manipulate the device remotely.

VERIFICATION

```
PING 8.8.8.8 (8.8.8.8) from 192.168.1.113 wlan0: 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=118 time=8.70 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=118 time=9.08 ms

--- 8.8.8.8 ping statistics ---
4 packets transmitted, 2 received, 50% packet loss, time 3033ms
rtt min/avg/max/mdev = 8.697/8.889/9.081/0.192 ms
Up
```

```
ping: connect: Network is unreachable
To: mbeal006@fiu.edu
From: mikephonehome@gmail.com
Subject: Python Test
Network outage detected
Down
```

SUMMARY

- Created a simple automating system for running test scripts:
  - General outage script.
  - Phone Home Script (Email)
- Assisted in the testing process for supplementary scripts.
- Obtained the necessary hardware to build the device:
  - Conducted research into the controller used (Raspberry Pi)
  - Configured the Raspberry Pi
  - Set up the LTE hat w/ cellular connection.
- Provided the network in which the device tests were performed.

REFERENCES

- *Reverse SSH tunneling - from start to end - jfrog connect (formerly UPSWIFT). JFrog Connect*. (n.d.). Retrieved April 17, 2022
- *Technical Details*. (n.d.). Sixfab Docs. Retrieved March 11, 2022
- *Raspberry Pi Documentation*. (n.d.). Raspberry Pi. Retrieved February 15, 2022
- *The Python Standard Library - Python 3.10.4 documentation. (n.d.).* Python.org. Retrieved February 16, 2022

ACKNOWLEDGEMENT

The material presented in this poster is based upon the work supported by Ryan Hutcheson (Kaseya). We thank Ryan Hutcheson for his assistance and mentorship that we received throughout the senior design project.