ECEN 5803: Mastering Embedded Systems Architecture

Project 3: VoIP Gateway Design Evaluation

Name: Erich Clever, Kanin McGuire,
Gaurang Rane, & Taher Ujjainwala
Instructor: Prof. Tim Scherr
Date: December 7th, 2021
Class: ECEN 5803
Assignment: Project 3: VoIP Gateway Design Evaluation

1. Executive Summary

The following document contains detailed descriptions and findings regarding the methods used to evaluate the BCM2837 MPU for use in Patton's e911 IP-PBX VoIP Gateway. Six testing modules were performed using the Raspberry Pi Model 3 B (RP3) development board, which is based around the BMC2837. The criteria for the testing modules included a comparison of Windows-based OSes (Windows 10 IoT and Windows CE 7) and Linux-based OSes (Debian) for suitability with telecom. and networking operations, as well as an analysis of the hardware capabilities and peripheral support.

The suitability of Windows 10 IoT Core was demonstrated through the implementation of a G.711 coder/decoder. Nevertheless, the fact that it is still in early stages of development for ARM was manifested in difficulties with setting up the environment. Debian, a Linux-based OS, allowed for much faster setup and development, and it demonstrated its suitability for a PBX implementation and other telecommunications operations that were implemented with some of the vast array of packages that are available for Linux distributions. Although Debian, at approximately 3 GB, is significantly larger than Windows IoT, both image sizes fall well within the 250 GB HDD budget for the Gateway. Finally, Windows 10 IoT Core Services has a fixed price of US$0.30 per month per device[1], which would result in $18 per device assuming a minimum product lifespan of five years. The cost savings can be allocated towards networking peripheral interfaces instead and the unlikely potential for third-party debugging contracts. Debian or another Linux distribution is clearly the best OS choice for Patton's IP-PBX gateway.

The BCM 2837 performed the preliminary PBX operations impressively and a hardware analysis revealed that it is capable of supporting all the necessary peripherals. Each core achieved a best-case score of 2404 DMIPS, which results in a total processing power of 9616 DMIPS, as the BCM 2837 contains four ARM Cortex A53 cores. This greatly exceeds the 500 DMIPS requirement. In addition to the verification of the interfaces included on the RP3 during development, Broadcom's datasheet[2] for the BCM2837 confirmed that it contains the required interfaces (SPI, HDMI, USB, I2S, GPIO) for the proposed PBX. Finally, the estimated cost of the solution based on the BCM2837 running a free Linux distribution is just under the $50 requirement. This estimate is projected to fall, as it is based on component quantities for a conservative figure of 1,000 units.

The BCM2837 running a Linux-based distribution for use in the Patton e911 IP-PBX VoIP Gateway is a GO.

2. Problem Statement and Objectives

Patton is seeking an MPU that is capable of performing telecommunications operations as a PBX. The qualifications for a viable candidate for this application include a minimum calls-per-second (CPS) capacity, the ability to process certain voice codec standards, and support for various communications standards peripherals. As the BCM2837 has been identified as a potential candidate, the following testing and analysis objectives were identified in order to determine this MPU's suitability for use in the e911 IP-PBX VoIP Gateway:

1. **Schedule**

   The product shall be available for the market within the next 18 months. Based on

lead times for components in the attached BOM (Appendix B), this timeline can be met.

2. **Budget**

   Budget for the embedded system is not to exceed $50 in production. This includes the PCB and connectors. Cost for the MCU should not be greater than $15.

   The proposed system, per the BOM, would cost $50. The MCU costs $3. This requirement is met.

3. **Environmental**

   Operating Temperature: 32 to 104° F (0 to 40°C), Operating Humidity: up to 90%, non-condensing, EMC: EN55022 and EN5502, Safety: EN 60950, CE, FCC Part 15 Class A; Part

   All components in the BOM, including the MCU, support the environmental requirements. Design guidelines and considerations provided by the above safety standards will be incorporated in the design process.

3. Approach and Methodology of Solutions

This section discusses the approaches and methodology the team has adopted in this project to design the prototype for evaluation. Both the high-level system design overview as well as evaluation measures are covered.

3.1 Block Diagram or Semi-Schematic diagram

See Figure 2 for the Block Diagram of the system.

3.2 Hardware Evaluation

- **Hardware evaluation based on availability of hardware peripherals (Required/Available):**
  1. GPIO -36 /54
  2. HDMI - 1/1
  3. SPI - 2/2
  4. USB - 1/1
  5. I2S - 1/1
- **Suggestion for interfacing sensor and user interface**
  1. Use GPIOs to interface with RAM memory, Switch and LED. Additionally, use GPIOs to provide control signals to RAM memory and E/T1/J1 Transceiver.
  2. Use I2S lines to interface Audio DAC to provide Audio output.
  3. Use SPI interface to interface ethernet PHY and ethernet switch for ethernet ports for debugging as well as for input signals .
  4. Use built-in HDMI interface for communicating with the HD display. Use HDMI to VGA converter COTS solution for VGA output.
  5. Use SPI interface for communicating with E1/T1 transceiver and FXS/FXO IC.
  6. Use built-in USB peripheral for 250GB HDD interface.
  7. Use parallel data and address line interface using GPIOs for RAM memory interface.
  8. Use a DC-DC buck converter for converting 12V to 5V. Use additional LDOs to provide 3.3V supply.

- **Hardware evaluation based on cost constraint**
  Proposed solution costing is USD 50 which is equal to the requirement in RFS. This includes costing of components, connectors, PCB manufacturing and assembly, and cables. Cost of components are taken from Octopart, findchips and aliexpress.
- **Hardware evaluation based on regulatory and testing standards -**
  Passive components, connectors to sensor integrated circuits used in the proposed solution will adhere to the ROHS requirement as well as required environmental standards. The product should have safety and design features in accordance with safety and EMI/EMC standards.

4. Module Test Results

Module 1 - Create the Windows CE Build Environment

Visual Basic 2008 was installed with the Platform Builder for Windows CE 7 plugin in addition to the other dependencies. A screenshot of the Platform Builder interface is included in the Appendix (Figure 1).

Module 2 - Boot Windows 10 IoT

A prebuilt image of Windows 10 IoT core was downloaded and used to boot the RP3 through the Windows 10 IoT Core Dashboard, which includes a performance-monitoring interface of the device running Windows 10 IoT (Figure 3, Appendix A). An FTDI Serial-to-USB cable was used to observe the debug messages during the boot sequence (Figure 4, Appendix A), which **output information like the primary image base, the module list, and ARM instructions which are used**. This required downloading WinDbg on the host machine to monitor the live kernel debug messages, and other configuration using Windows PowerShell commands (See link to video under “Module 2 - Serial Debugging” in GitHub repository MESA_Project3_VoIP_Gateway’s README).

The **memory used by the main OS is 1.4GB** (Figure 5, Appendix A) A **screenshot of the terminal window is included in the Appendix (**Figure 6, Appendix A**). After connecting the HDMI output to a monitor and rebooting the system, the Windows 10 IoT home page is displayed** (Link under “Module 2 - Booting Windows 10 IoT core” in GitHub repository MESA_Project3_VoIP_Gateway’s README).

To test the ability to create and perform telecommunications operations, a G.711 coder/decoder was written in C and used to decode an encoded file. The decoder function Snack_Mulaw2Lin() was implemented to convert U-law to 16-bit PCM values. The decoded file had 16-bit precision unlike the encoded file which had 8-bit precision. The original file size was 102KB, while its decoded size was 205KB (File “1449184213-A_eng_f6.wav” and “output.wav” in GitHub repository MESA_Project3_VoIP_Gateway -> Module 2 -> Coder/Decoder).

The behavior of Windows 10 IoT is different from Linux distributions in the following manners:

- The image size of Windows 10 IoT core (1.4 GB) is much smaller than Linux Debian (2.9 GB)

- Linux provides multiple options for access and administration like Putty, VNC Viewer, and Serial SSH Terminal, while Windows 10 IoT provides a streamlined GUI and PowerShell through the IoT Core Dashboard
- Linux distributions are much more stable than Windows 10 IoT core, and there is a much larger Linux community for troubleshooting and application configuration than with Windows 10 IoT Core

Module 3 - Scripting with Linux

An executable script was written to display various types of information, and a menu was created to allow the user to select which item to run. The various items are **date, disk, process, hardware and a simple quit**. The Date function displays the current date and time and the current version of the Linux kernel. The Disk function shows information about the disk partition and the current disk usage. The Process function displays all of the processes that are currently running, including the root user's. The Hardware function sends a list of available hardware peripherals on the Raspberry Pi and also displays RAM usage statistics. The script runs on startup and prints all of the information to a log file accessible on the desktop. The video link is attached with the name Module 3 output on the git repository.

Module 4 - Build your own PBX with Asterisk

Asterisk, an open-source PBX server, was downloaded to a Debian instantiation in order to test running a PBX server on the BCM2837. With Asterisk included, **the image size is 3.1GB**. In order to add phones and destinations to the network, an extensions.conf file was created. At extension 100, a voicemail message was configured to **playback "Hello World" when called**.

The file sip.conf was configured to add all the authentication details, ID and password of the softphone, network IP range, and necessary ports. This allows the user to add softphones and allows the authentication to connect to a single concentrated server which is the RP3 running Asterisk.

In order to make a phone to phone call, three softphones were established: One (7001) on a PC using the third-party software, MicroSIP (Figure 7), and another (7002) on a mobile device running Android using the MizuDroid application (Figure 8). A third softphone (100) was established on Iphone using Nathtel Voip (Figure 9).

A call was made from 7002 to 7001 initially to test the basic operation of VoIP calling (Figure 10). A call was then made from 7002 to 100 to check the connectivity and audio quality, then the Asterisk server was configured to play a "Hello World" voicemail message tone at extension 100.

**Note: The bonus portion of an SIP phone to SIP phone call was successfully made from 7002 to 100.*

Module 5 - Build an IoT Telecom Application

A Morse code translator/receiver was created and run on Raspbian. The input for the application is a string of text which is then output in Morse code. The input is provided by the user via the command line. An example of its usage is shown in Figure 11 in the Appendix. The purpose of

this Module was to explore the development experience and performance of another telecommunication operation with a Linux-based distribution.

Module 7 - Measure ARM Cortex-A53 DMIPS

The result of running the Dhrystone benchmark on Debian on the BCM2837 was a VAX DMIPS of 2404.97. A screenshot is included in Appendix A (Figure 12).

5. Recommendations: **GO**

The BCM2837 was thoroughly evaluated with respect to the RFS for the Patton e911 IP-PBX VoIP Gateway for both hardware and software requirements. A hardware evaluation of the BCM2837 was performed to determine peripheral availability, fulfillment of environmental and safety requirements, and fulfillment of budgetary restrictions. The BCM2837 can meet all these requirements and successfully perform all the processing associated with a PBX system. In fact, the MPU provides much higher processing power than required. This computing power is sufficient for monitoring the analog and digital I/O channels, encoding and decoding audio files, generating analog signals, and providing a user interface.

The BCM2837 supports a Linux-based OS, which has been identified as the optimal solution for operational purposes. Linux is open-source with a large developer community, and it has robust support for a vast array of software libraries and software development tools related to communications applications, like Asterisk, as well as user interfaces. This will accelerate the development process and reduce overall cost.

The BCM2837 contains built-in USB and HDMI interfaces for providing interface options, as well as an I2S interface which can be used to provide Audio output. Also included are SPI and UART interfaces for BRI ports, T1/E1 ports, and Ethernet Debugging ports. Ultimately, this project is feasible within the specified timeline and under the budgetary restrictions dictated by the RFS if it is based on the peripheral hardware specified in the BOM used to meet the various communications standards requirements. Therefore, **the BCM2837 running a Linux-based distribution for use in the Patton e911 IP-PBX VoIP Gateway is a GO.**

List of Deliverables

- GitHub repository MESA_Project3_VoIP_Gateway
  - Module 1 folder
    - Platform Builder interface screenshot
  - Module 2 folder
    - Coder/Decoder subfolder
      - Original encoded .wav files and decoded .wav file (output.wav)
      - decode.c - Code in which decoding of .wav files occurs
    - Serial Debugging subfolder
      - Various screenshots of debugging messages
  - Module 3 folder
    - menu - Executable script for custom functions in Linux
    - log1.log - Output from executing custom functions
  - Module 4 folder
    - Asterisk .conf files
    - Report screenshots
  - Module 5 folder
    - Report screenshots
  - Module 7 folder
    - Dhrystone benchmark result screenshots
  - README
    - Module 2 - Booting Windows 10 IoT core
      - Link to video of Windows 10 IoT Core boot sequence
    - Module 2 - Coder/Decoder
      - Link to video demonstrating decoding of encoded .wav file using G.711
    - Module 2 - Serial Debugging
      - Link to video demonstrating debug messages received over serial port on device running Window 10 IoT Core
    - Module 3 - Output
      - Link to video demonstrating output of executable script running on a Linux distribution
- **Block Diagram** of proposed PBX VoIP Gateway
- **Bill of Materials** with adequate peripheral communication hardware for PBX VoIP Gateway functionality
- **Evaluation report** - GO/No-GO for Broadcom BCM2837
  - Test results and design recommendations
  - Hardware Analysis

Project Staffing

1) Erich Clever
Graduate Student
University of Colorado Boulder
Email: erich.clever@colorado.edu
2) Kanin McGuire
Graduate Student
University of Colorado Boulder
Email: kamc9940@colorado.edu
3) Gaurang K. Rane
Graduate Student
University of Colorado Boulder
Email: gara8950@colorado.edu
4) Taher Ujjainwala
Graduate Student
University of Colorado Boulder
Email: tauj5361@colorado.edu

References

1. https://azure.microsoft.com/en-us/pricing/details/windows-10-iot-core/
2. Broadcom BCM2837 ARM Peripherals
3. How to Install Windows 10 IoT Core on Raspberry Pi 3 (makeuseof.com)
4. Windows Debugger - Windows IoT | Microsoft Docs
5. Download Debugging Tools for Windows - WinDbg - Windows drivers | Microsoft Docs
6. Asterisk VOIP Server Setup On Ubuntu 20 | Making Calls via SIP Soft Phone | Rocket System - YouTube
7. http://www.roylongbottom.org.uk/dhrystone%20results.htm#anchorAndroid
8. https://www.micron.com/products/dram/lpdram/part-catalog/mt53d512m16d1ds-046-wt
9. https://www.wiznet.io/product-item/w5500/
10. https://ww1.microchip.com/downloads/en/DeviceDoc/KSZ8895MQX-RQX-FQX-MLX-Integrated-5-Port-10-100-Managed-Ethernet-Switch-DS00002246B.pdf
11. https://www.monolithicpower.com/en/documentview/productdocument/index/version/2/document_type/Datasheet/lang/en/sku/NB679/
12. https://media.digikey.com/pdf/Data%20Sheets/Silicon%20Laboratories%20PDFs/Si3217x_291x_Brief.pdf
13. https://media.digikey.com/pdf/Data%20Sheets/Silicon%20Laboratories%20PDFs/Si3217x_291x_Brief.pdf
14. https://www.maximintegrated.com/en/products/comms/DS2148.html

Appendix A : Figures

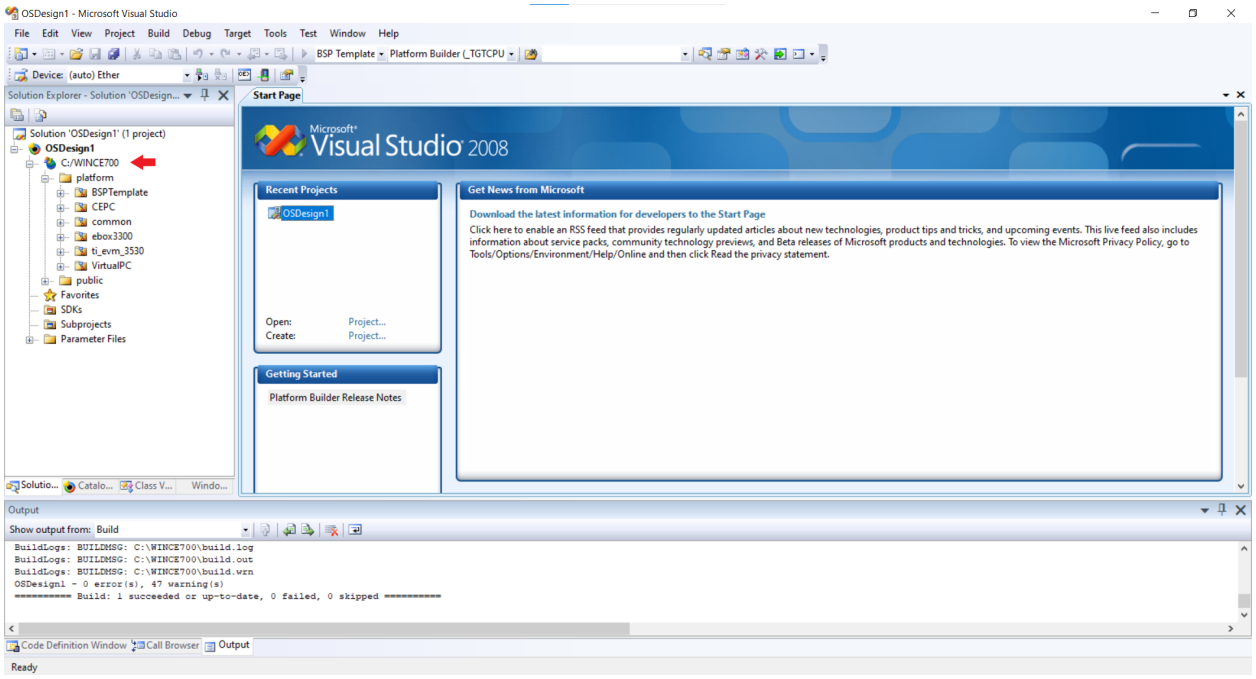

Figure 1 - Visual Basic 2008 with Windows CE 2007 Platform Builder Plugin

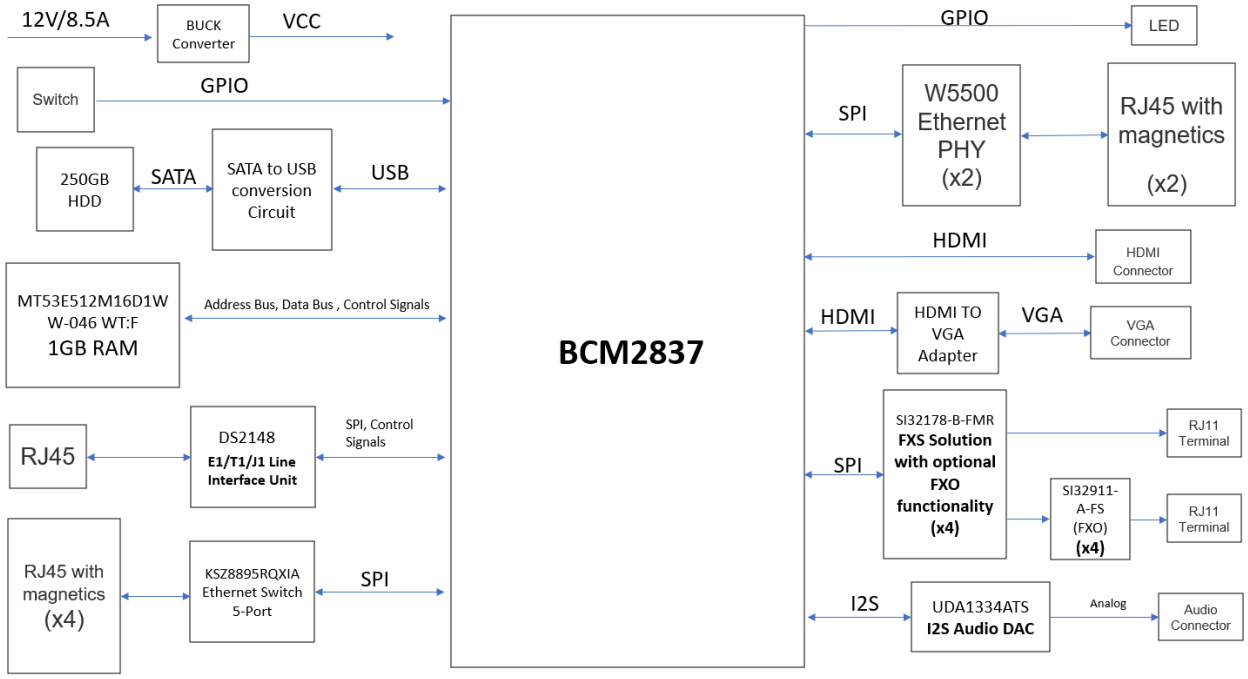

Figure 2 - Block Diagram of proposed PBX VOIP Gateway

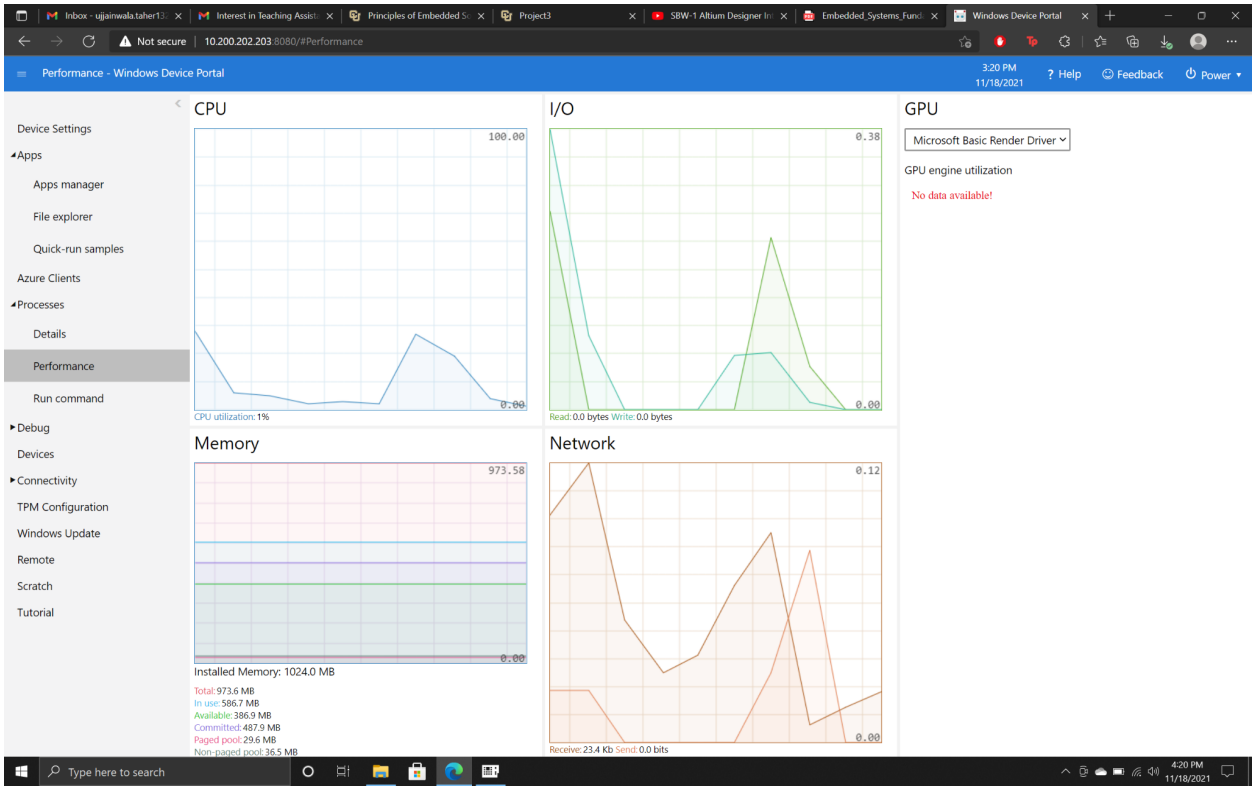

Figure 3: Performance of Windows 10 IoT core

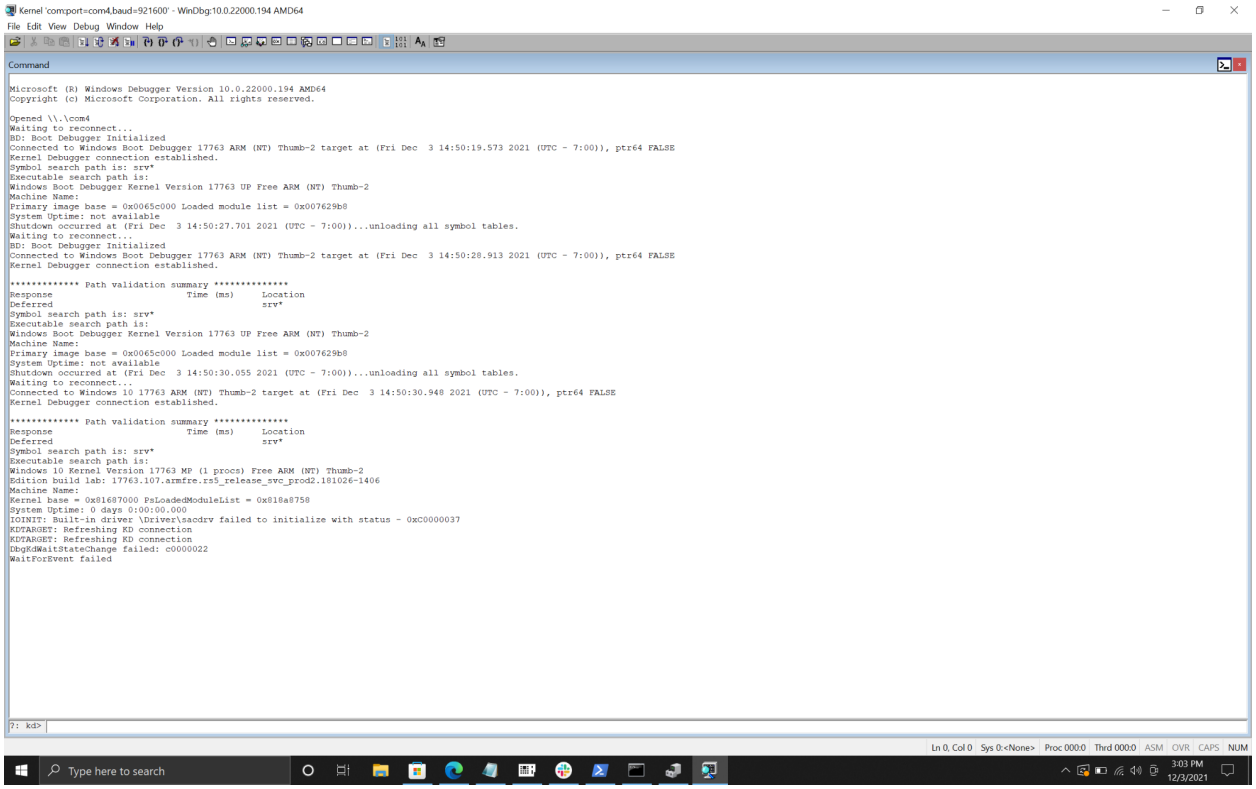

Figure 4 - Kernel Serial Debugging Output

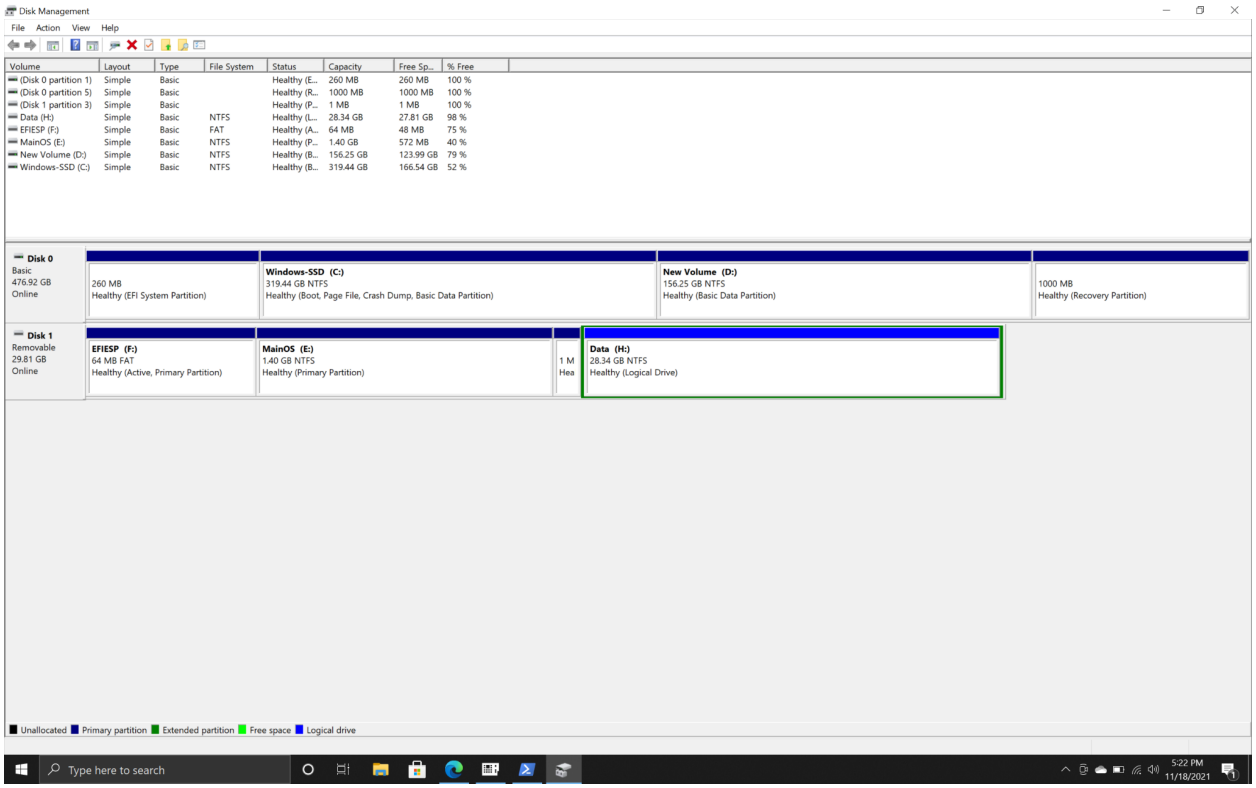

Figure 5 - Memory used by Windows 10 IoT Core

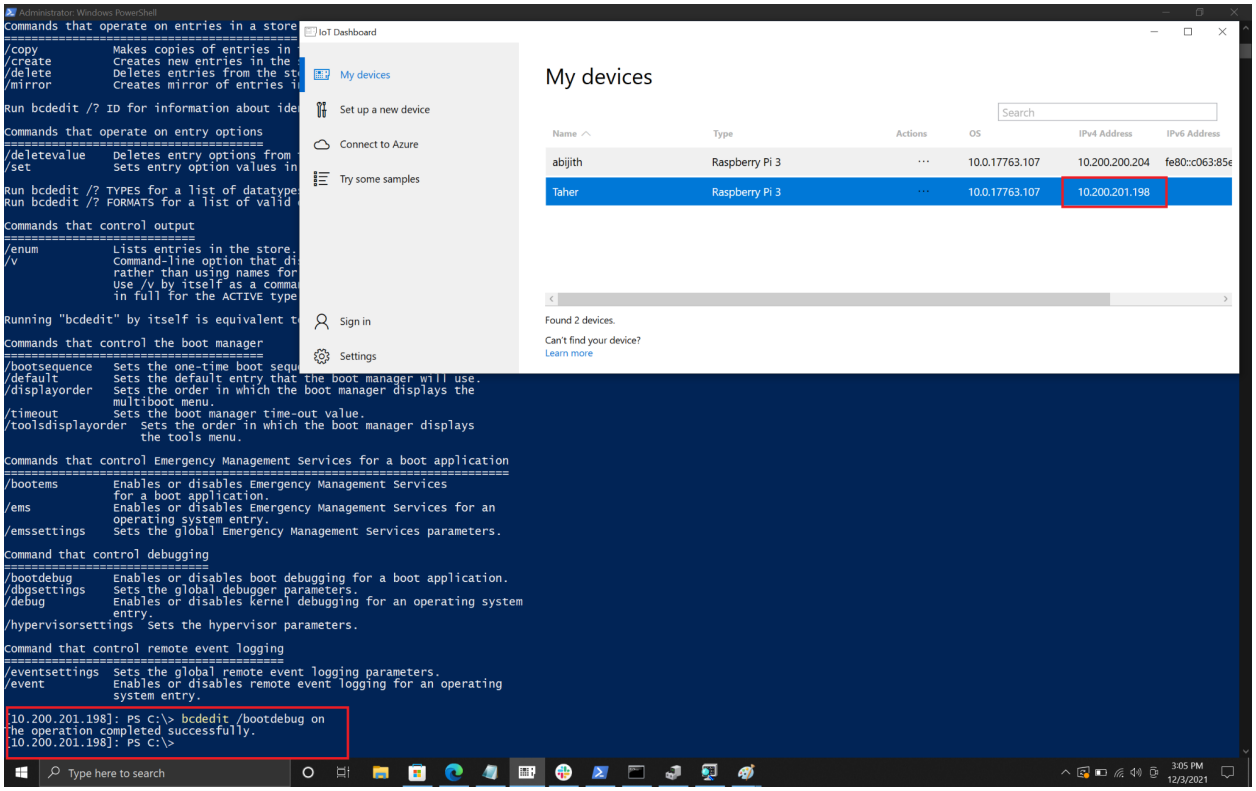

Figure 6 - PowerShell Window and Login IoT core Dashboard

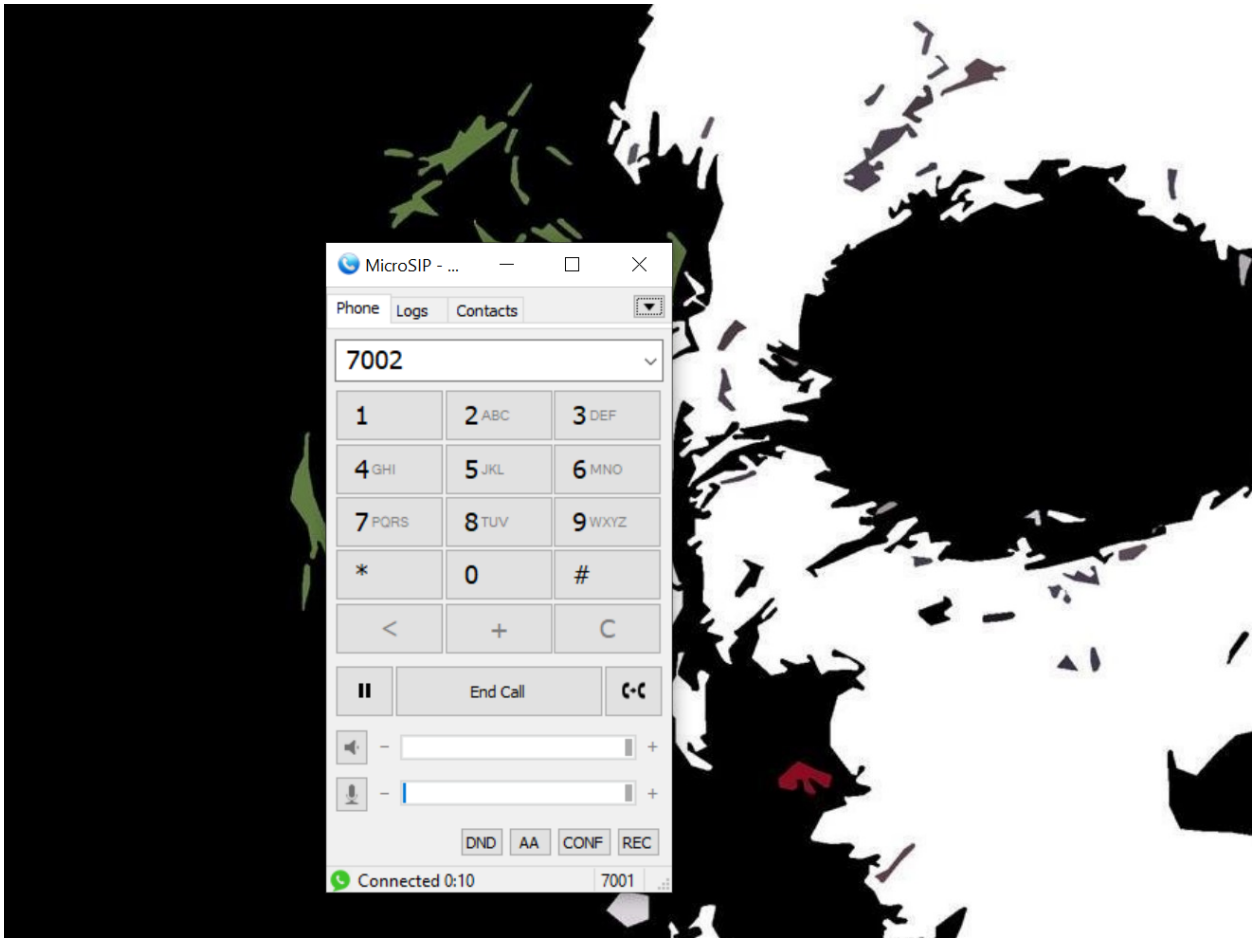

Figure 7 - Softphone 1 - 7001

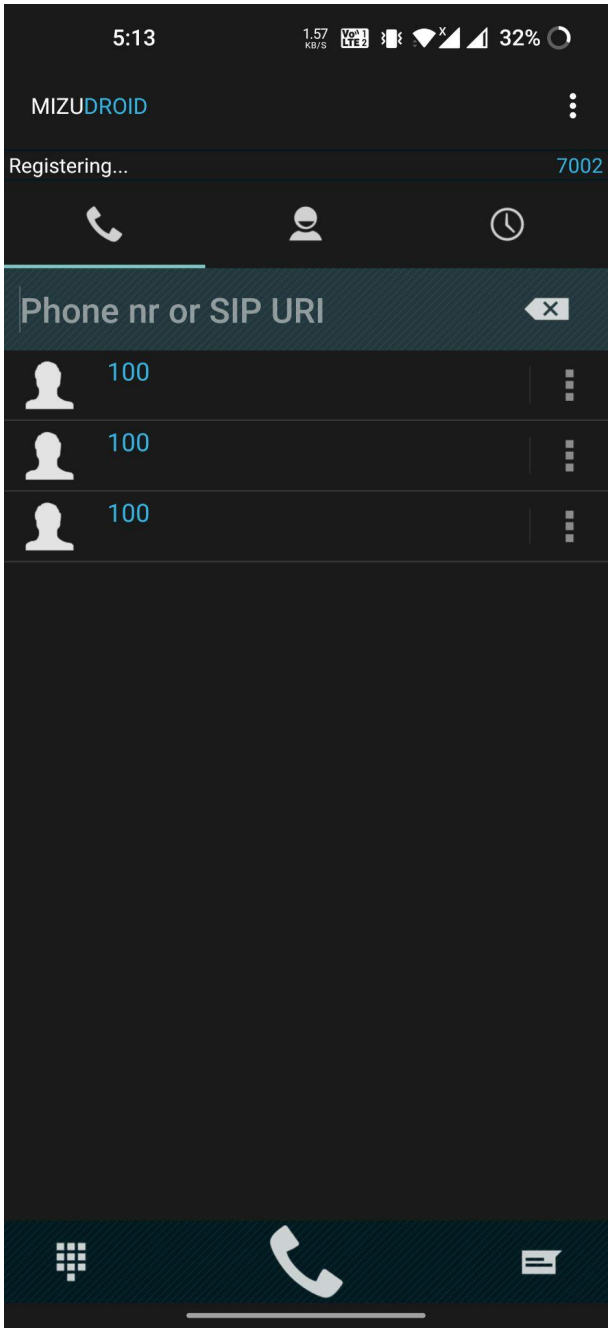

Figure 8 - Softphone 2 - 7002

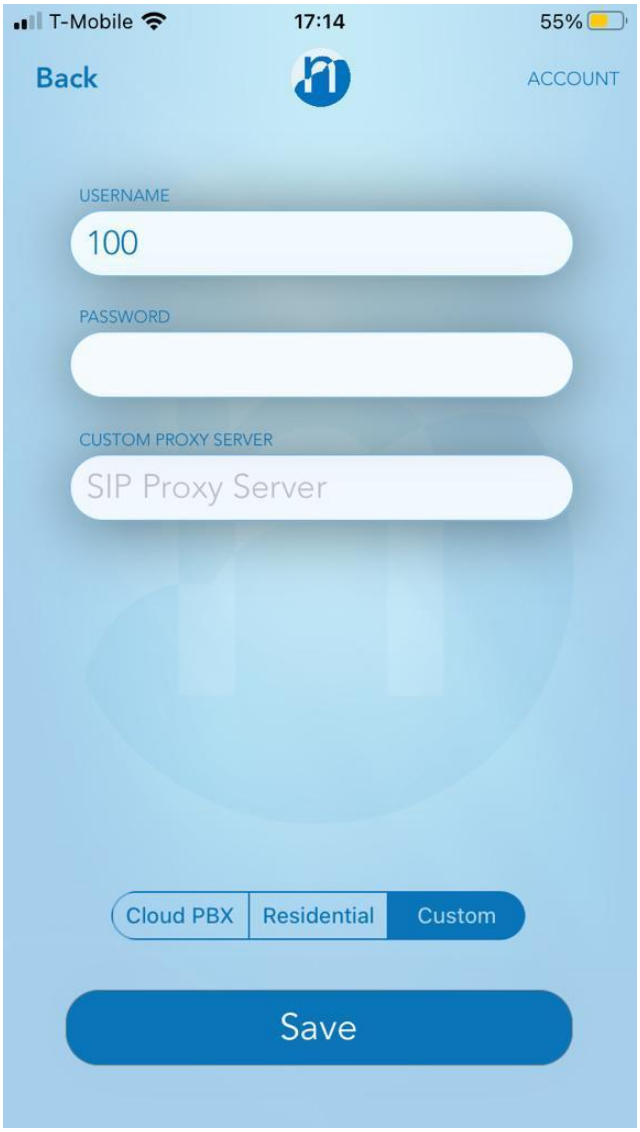

Figure 9 - Softphone 3 - 100

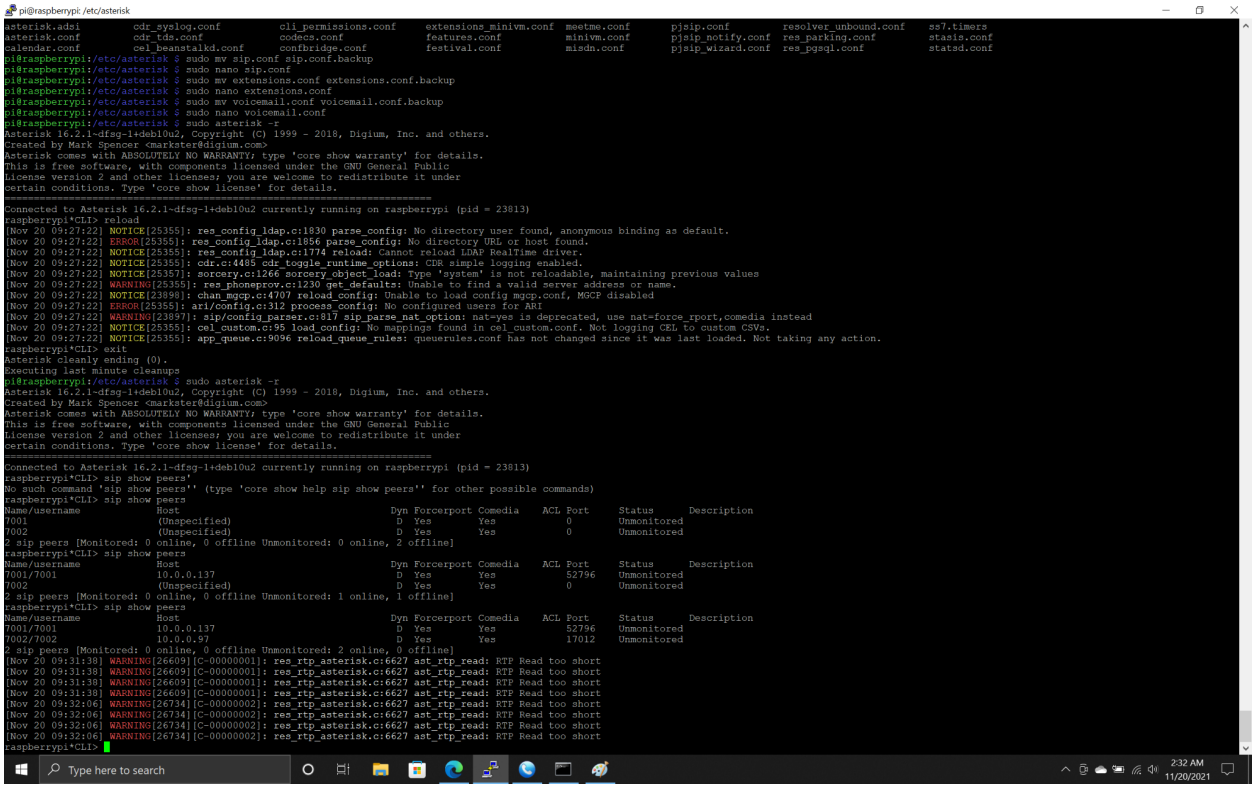

Figure 10 - SIP VoIP Peers

```
pi@raspberrypi:~/Desktop $ python3 morsecode.py
Arguments count: 1
Argument      0: morsecode.py
malformed input
pi@raspberrypi:~/Desktop $ python3 morsecode.py a a
Arguments count: 3
Argument      0: morsecode.py
Argument      1: a
Argument      2: a
malformed input
pi@raspberrypi:~/Desktop $ python3 morsecode.py "Translate"
Arguments count: 2
Argument      0: morsecode.py
Argument      1: Translate
Encoded Message: - .-. .- -. ... .-.. .- - .
Decoded Message: TRANSLATE
pi@raspberrypi:~/Desktop $ python3 morsecode.py "Translate this to and from morse code"
Arguments count: 2
Argument      0: morsecode.py
Argument      1: Translate this to and from morse code
Encoded Message: - .-. .- -. ... .-.. .- - . ..-.- - .... .. ... ..--.- - --- ..--.- .- -. -.. ..--.- ..-. .-. --- -- .
.--.- -- --- .-. ... . ..--.- -.-. --- -.. .
Decoded Message: TRANSLATE THIS TO AND FROM MORSE CODE
```

Figure 11 - Morse Code Output

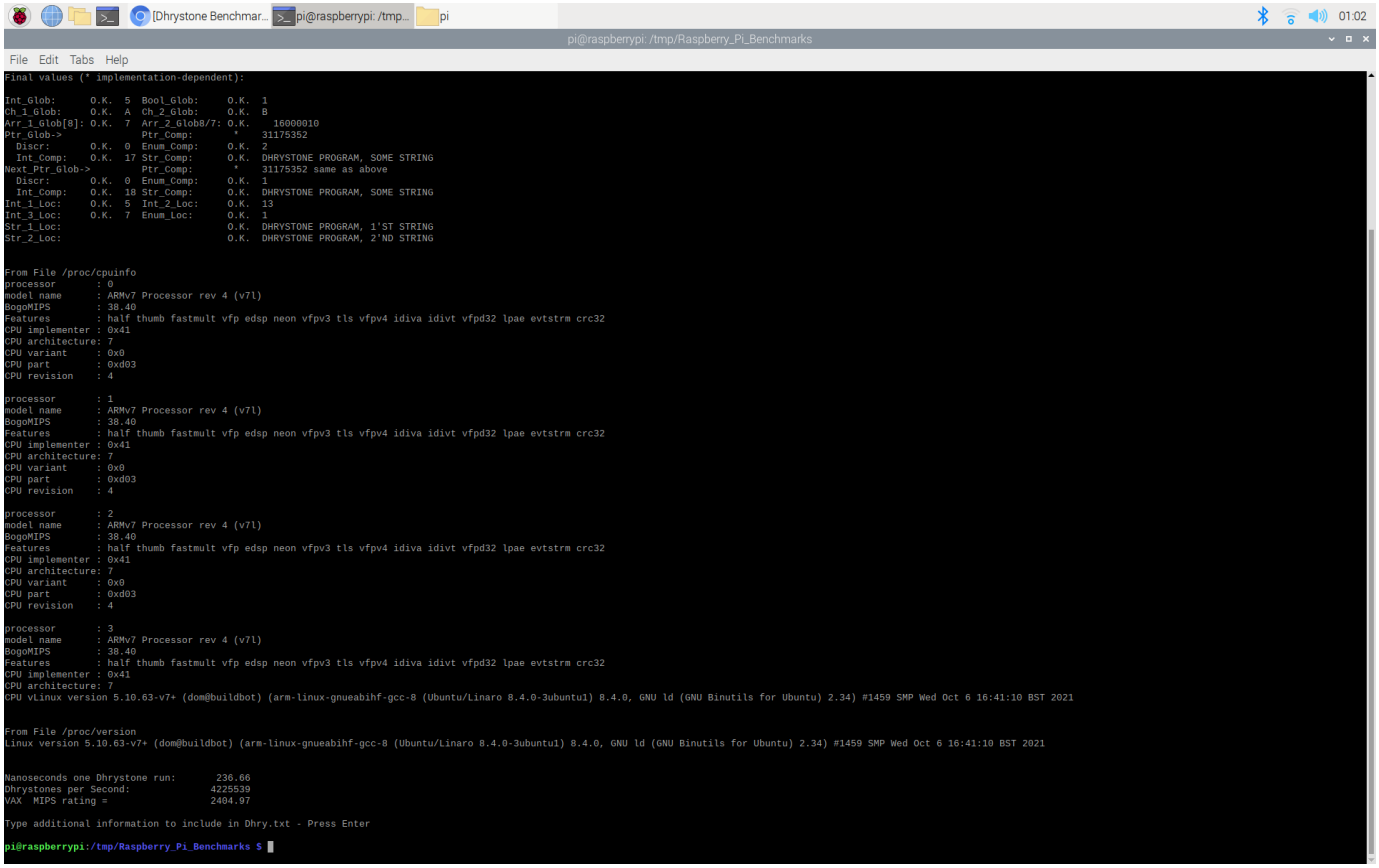

Figure 12 - Dhrystone Benchmark - Raspberry Pi 3

Appendix B : Bill of Materials

Item Number	Manufacturer Name	Part Number	Part Description	Order Quantity	Quantity/ device	Cost per Unit	Cost/Device in USD
1	Micron	MT53E512M16D1WW-046 WT:F	DRAM LPDDR4 8G X16 WFBGA	1000	1	5.3	5.3
2	WIZnet	W5500	Ethernet Controller 10/100 Base-T/TX PHY SPI Interface 48-LQFP (7x7)	1000	2	1	2
3	TOSHIBA	MK2555GSXF	TOS HD 250GB 5400RPM SATA	1000	1	8	8
4	Microchip	KSZ8895RQXIA	Ethernet Switch 5-Port 100Mbps Automotive 128-Pin PQFP Tray	1000	1	4.6	4.6
5	MPS	NB679GD-P	5.5-26V, Low Iq, High Current, Fixed 5.1V-8A	1000	1	0.9	0.9
6	Broadcomm	BCM2837	4x ARM Cortex-A53 @ 1.2 GHz	1000	1	3	3
7	Skyworks Solutions Inc.	SI32178-B-FMR	Telecom IC Subscriber Line Interface Concept (SLIC), CODEC 42-QFN (5x7)	1000	4	1.13	4.5
8	Skyworks Solutions Inc.	SI32911-A-FS	Telecom IC Subscriber Line Interface Concept (SLIC), CODEC 16-SOIC	1000	4	1.2	4.8
9	Maxim	DS2148	IC TELECOM INTERFACE 44TQFP	1000	1	6.4	6.4
10	WIZnet	J1B1211CCD	1 Port RJ45Through Hole 10/100 Base-T, AutoMDIX	1000	6	0.8	4.8
11	NXP	UDA1334ATS	IC DAC/AUDIO 24BIT 96K 16SSOP	1000	1	0.578	0.578
12	PCB Assembly cost with connector (15% of BOM cost)						5
TOTAL							49.878