7-18-2019

# sensor data collection and performance evaluation using a TK1 board

Saurabh Pahune

# Sensor data collection and performance evaluation using a TK1 board

by

Saurabh Pahune

A Thesis

Submitted in Partial Fulfillment of the

Requirements for the Degree of

Master of Science

Major: Computer Engineering

The University of Memphis

July 2019

# 1 Acknowledgment

My heartful gratitude to my thesis advisor, Dr. Bonny Banerjee, Associate Professor in the research-intensive Institute for Intelligent Systems and the Department of Electrical and Computer Engineering at the University of Memphis, whose insights and constant support helped me get through in the right direction. At any hour of need, he had his office doors open for me, alongside monitoring my progress periodically which helped me meet my deadlines.
I would like to thank Dr. Madhusudhanan Balasubramanian and Dr. Dipankar Dasgupta of Department of Computer science Engineering, for accepting to be a part of my Thesis Committee and for their timely suggestions, both academic and non-academic, which have helped me view things in different perspective.
I would also like to add appreciation to my colleagues at Computational Intelligence Laboratory, who made the work and study environment so lively, with constant support and help throughout my stay.
I would never forget the unconditional support I received from my parents in nitty gritty, and for the unfailing encouragement they gave me in moments of difficulty. I would also like to thank my family, friends and teachers who all backed me in my hour of need and without who I will not be where I am.

# Contents

# List of Figures

# List of Tables

# Abbreviations

**L4T** Linux For Tegra

**GPIO** General Purpose Input Output

**SATA** Serial Advanced Technology Attachment

**SDA** Serial Data Access Line

**SCL** Serial Clock Line

**I2C** Inter-integrated Circuit

**VSM** Virtual System Modeling

**CUDA** Compute Unified Device Architecture

**rtsp** Real Time Stream Protocol

**IP** Internet Protocol

**ISP** Image Signal Processor

**BSP** Board Support Package

**PLL** Phase Lock Loop

# 2 Introduction

## 2.1 Objective of Research

Applications involving a plethora of sensors are abundant in today's world. Our goal is to analyze the performance of a Nvidia Jetson TK1 board for sensor data collection. The Nvidia Jetson TEGRA-KEPLER(TK1) board is used as the processor as it is one of the most powerful processors for embedded applications with the flexibility to connect to a plethora of sensors. Data transfer for communication is facilitated via Bluetooth and Wireless Fidelity (Wi-Fi). Results on the performance of this setup is reported in experiments with different sensors such as cameras, microphone, gas sensor,temperature/pressure/humidity sensor, and Garmin smart health watch determined heart rate/distance/speed/altitude/- position latitude and longitude and using metrics such as read/write speed,heat generated of Central Processing Unit (CPU)and transmission delay.

## 2.2 Contribution of Research

In one of them work, Dr.Banerjee encouraged me to make manuscript of a practical device and tried to make survey of a different embedded board and sensors. Firstly we perform analytic calculation and simulation in software, to check device will work with respect to senors connection.We designed and measured performance of a board and sensors. We developed automation script to collect data automatically, and plotting data. We also evaluated different benchmark observations and results.

# 3 Literature Review

Iervolino (2017) proposed a system [22] using a wearable device for collecting data on board sensors and extract meaningful data. This system consist of small electronics board with capacity of SD card, accelerometer, gyroscope, Bluetooth wireless module, pressure and humidity sensor. This complete set of device author connected right ankle of a foot to analyze sport performance analysis [Walking/Running]. Here author totally focused on running and walking analysis by using separate accelerometer and gyroscope sensor.

Mardonova, M., & Choi, Y. (2018) provides a system [23] for overview of mining industry workers, using features of different bio-sensors and environmental sensors. In this system author used biological sensors on clothing of a person to measure body temperature, pulse rate of a mining workers .

In our research paper our contribution was the use of Garmin Smart-watch instead of wearable sensor on clothes of industry people. Smartwatch has all inbuilt biological sensors and GPS navigation system to track workers. Here all gadgets [smartwatch, Mobile] are connected to TK1 device via Bluetooth or Wi-Fi. As our system has Video cameras, which gives live streaming information to another person. As TK1 board is powerful board we can use this system for multiple real-time application by running algorithms. [ex: Health issues, Avoiding Critical emergency situations]

# 4 Device description

In our system a person wears the sensors and it will capture multiple surrounding environmental data as well as persons movements, heart rate, GPS location, calories, step movements of a person. If in any case if someone tries to attack/harm/ rob a person. This smart system can sense persons physiological data using smart watch which connected to TK1 board. If there is any abnormal behavior it can send an alert message to Police without alerting an attacker including video and location details in message. In Figure 1 shows TK1 board connected to multiple USB hub. This USB hub has connection of Garmin watch, USBHD cameras and connection of other peripheral devices such as Keyboard, Mouse. Multichannel gas sensor and BMP280 sensor connected to TKI General Purpose Input Output (GPIO) pins using I2C interface. Jetson TK1 has six I2C buses and four buses are available on TK1 expansion header[3]. TK1 board has total 125 GPIO pins are available. In our system we are using two I2C buses for BMP280 sensor [0x76] address and Multichannel gas sensor [0x04]address detect by TK1 board. BMP280 sensor detect Temperature/Pressure/ Humidity and Multichannel gas sensor detect Carbon Monoxide, Ethanol, Hydrogen, Ammonia, Methane, Propane, Iso-butane, Nitrogen dioxide in PPM [parts per million] unit.

## 4.1 Hardware Description

### 4.1.1 Nvidia Jetson TK1 board

This Jetson TK1 board maintained by the community NVIDIA. Jetson TK1 is NVIDIA's embedded Linux development platform. In a single chip it has a Tegra (CPU), Kepler(K1-GPU), Image Signal Processor (ISP) (Image signal processor), selling price 192 USD. This board works on Linux4Tegra (Linux For Tegra (L4T)) OS (example: Ubuntu 14.04 etc.). This Jetson board TK1 has similar features as Raspberry pi but also has PC oriented features such as Serial Advanced Technology Attachment (SATA) (Serial advanced technology attachment), mini-PCIe and fan to allow continues operation under heavy workload.[3]

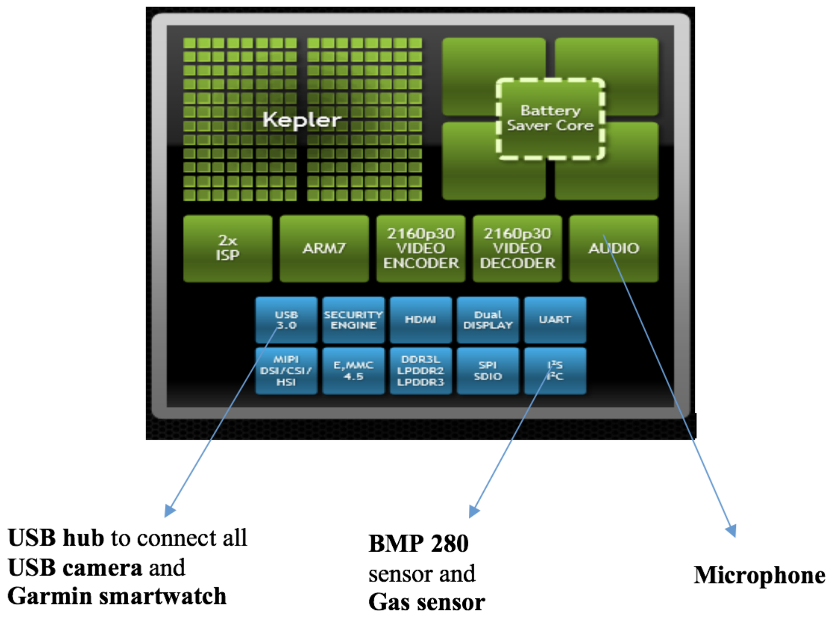


Figure 1: Overview of the system with Jetson TK1 board

### 4.1.2 BMP 280 sensor

BMP 280 sensors interface over I2C to Nvidia Jetson TK1 board. This sensor has four pins (Vcc, GND, SCL and SDA). This sensor has two I2C logic pins: SCL (Serial clock line), SDA (Serial data access). BMP280 sensor address 0x76.[8]

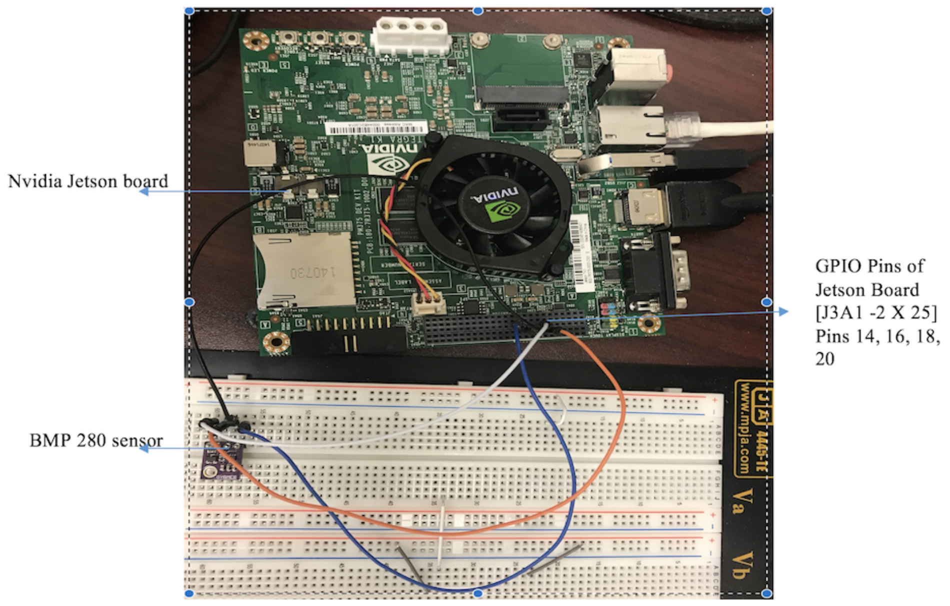

Nvidia Jetson board
GPIO Pins of Jetson Board [J3A1 -2 X 25] Pins 14, 16, 18, 20
BMP 280 sensor

Figure 2: BMP280 Sensor interface to Jetson board via I2C

1. BMP280 connect Vin pin to VCC (3.3V)-J3A1 (GPIO PIN: 16) – [Jetson Board].

2. BMP280 connect **GND!** (**GND!**) pin to GND–J3A1 (GPIO PIN:14) – [Jetson Board].
3. BMP280 connect Serial Clock Line (SCL) pin to SCL –J3A1 (GPIO PIN:18) – [Jetson Board].
4. BMP280 connect Serial Data Access Line (SDA) pin to SDA –J3A1 (GPIO PIN:20) – [Jetson Board].

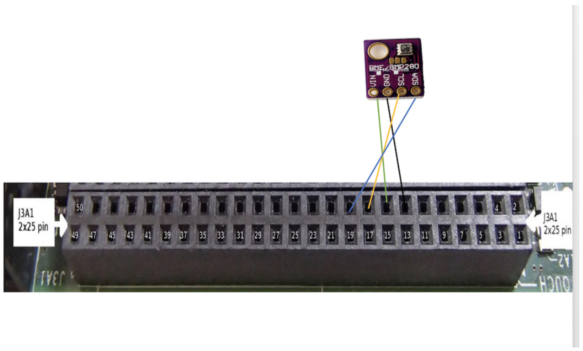


Figure 3: BMP280 sensor pin connection to a GPIO of TK1

Configure I2C interface for Jetson Board:

1. In order to use this module, must need to enable I2C interface on Nvidia Jetson board as it is not enable by default.[3][6]
2. Inter-integrated Circuit (I2C) is a multi-device bus used to connect low speed peripherals to computers and embedded system. [3]
3. Nvidia Jetson TK1 board supports this I2C interface using on its GPIO header.

Enable I2C using Jetson TK1 board Utility-Software Setup and Installation:

1. Once board is wired up with BMP280 sensor turn ON the Jetson TK1.
2. Need to install I2C tool.
3. Using some command on Linux terminal:

   **sudo apt-get update.**

   **sudo apt-get install-y i2c-tools.**

   **sudo apt install python-smbus.**

   **sudo apt-get update-y – python- smbus- i2c- tools.**

   **sudo adduser ubuntu i2c.**

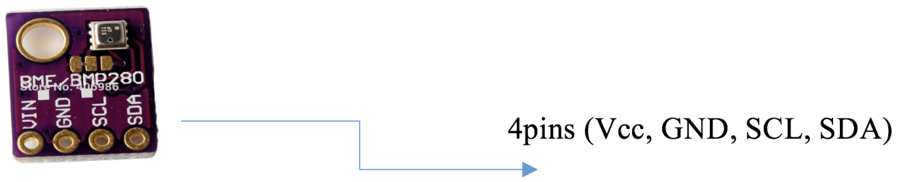


Figure 4: Overview of the system with Jetson TK1 board

### 4.1.3 USB Camera

In this system, we have used two cameras to Jetson TK1 board. One LogitechHD720 USB camera and another dual lens USB camera. All three lenses of a camera connected to Jetson TK1 board using multiple USB hub connection. Each lenses of camera has different viewing angle on to capture surrounding data in form of capturing wide angle. All three USB lenses has port 3.0 version. In our system two cameras are use to capture wide angle surrounding view. Using three lenses we have calculated reading and writing speed of camera with different frames per second.

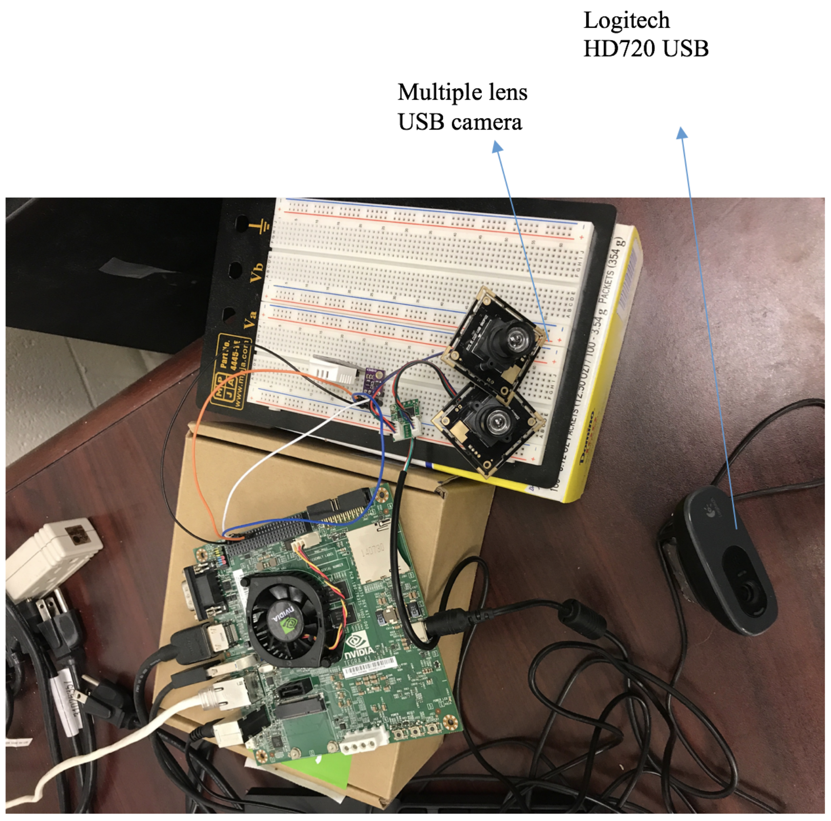


Figure 5: Dual lens and logitech webcam with Jetson TK1

### 4.1.4 Microphone

Microphone are used to convert input audio sound into output of an electric signal. The output of electric signal is applied to Single board computer of a Jetson TK1 board for storing audio data.

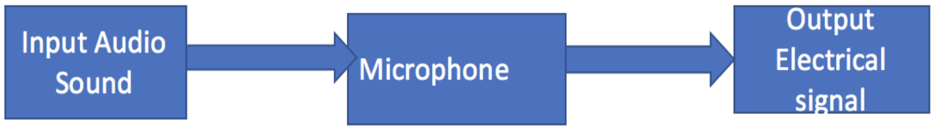


Figure 6: Block diagram of a microphone mechanism

In below script "arecord" is a command line for sound file recorder for soundcard driver. It supports several file formats and multiple soundcard with multiple devices. The audio file record with .wav extension [waveform audio file format]

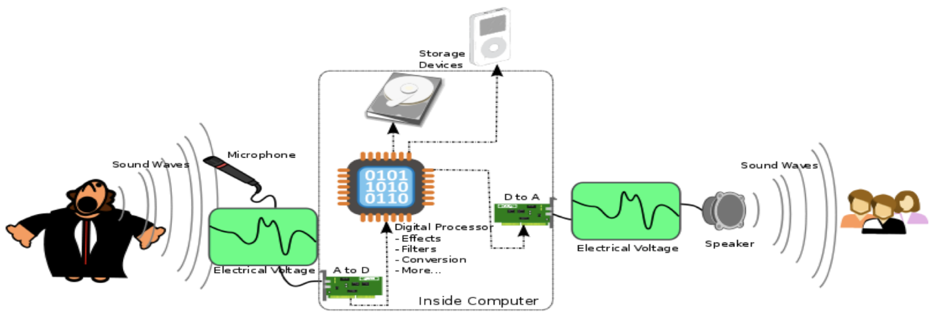
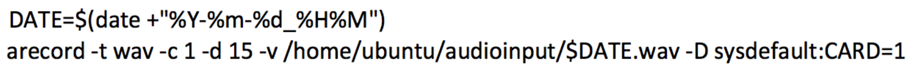


Figure 7: Microphone interfacing with single board computer



```
DATE=$(date +"%Y-%m-%d_%H%M")
arecord -t wav -c 1 -d 15 -v /home/ubuntu/audioinput/$DATE.wav -D sysdefault:CARD=1
```

Figure 8: Audio recording system inside Jetson TK1 board using arecord

### 4.1.5 Gas Sensor

Multichannel gas sensor is an environmental detecting sensor which detects unhealthy gases in an environment. This sensor has 4 pins VCC, GND, SCL and SDA. This multichannel sensor gas sensor detect multiple gases such as Carbon monoxide CO, Nitrogen dioxide NO2, Ethanol C2H6OH, Hydrogen H2, Ammonia NH3, Methane CH4, Propane C3H8, Iso-butane C4H10. All gases are measures in ppm [parts per molecule] unit.

Figure 9: Multichannel gas sensor

### 4.1.6 Intel 7260 Wi-Fi and Bluetooth card

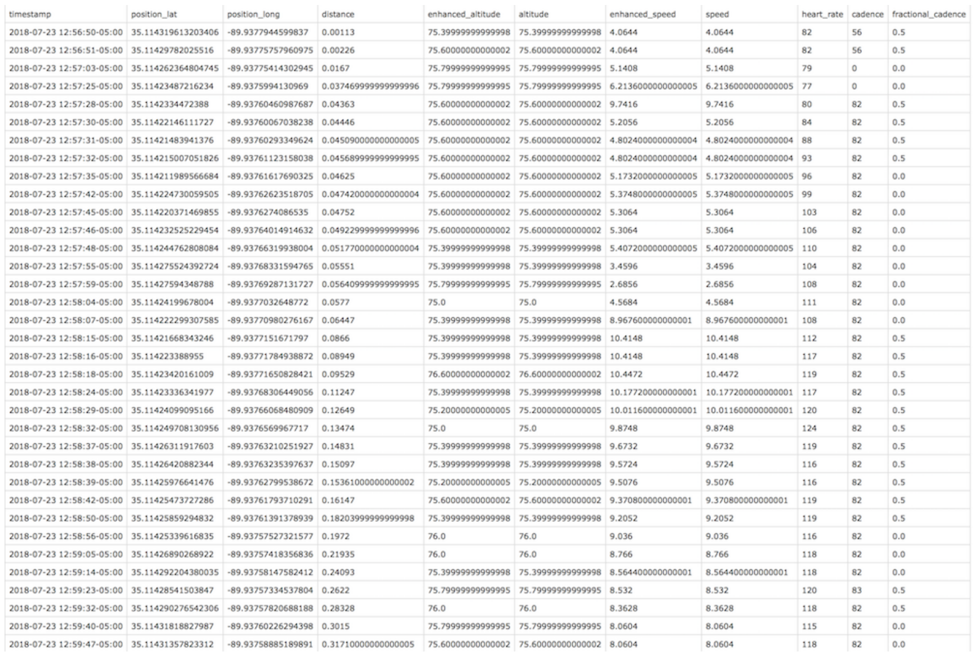
Jetson TK1 board supports Wifi which can be added using Minipcie card slot in TK1 board. In our system we have used wifi card intel 7260 for wireless connection. Generally it needs two things in order to work such as: Antenna pairs are required to connect into network and wifi ['iwlwifi'] driver which supports linux kernel. 'iwlwifi' is a wireless driver for intel wireless chip. Here, antenna pair has two pigtail adapters to connect to tiny antenna ports on Wifi card with standard antenna mount which shown in given figure.

### 4.1.7 Garmin Vivoactive-HR smartwatch

In our system, we have used Garmin vivoactive HR model as a smartwatch. We have connected vivoactive HR watch to Jetson TK1 board via USB cable. This watch has multiple features such as heart rate, step counter, skin temperature, GPS system, gyroscope, accelerometer, distance traveled, calories burned etc.

Figure 10: Garmin Smart-watch

Garmin watch stored all health related in data in .FIT file format and sync .FIT files with Jetson TK1 board. using python it use to convert .FIT files into .CSV file.[4]



| timestamp | position_lat | position_long | distance | enhanced_altitude | altitude | enhanced_speed | speed | heart_rate | cadence | fractional_cadence |
|---|---|---|---|---|---|---|---|---|---|---|
| 2018-07-23 12:56:50-05:00 | 35.114319613203406 | -89.9377944599837 | 0.00113 | 75.39999999999998 | 75.39999999999998 | 4.0644 | 4.0644 | 82 | 56 | 0.5 |
| 2018-07-23 12:56:51-05:00 | 35.11429782025516 | -89.93775757960975 | 0.00226 | 75.60000000000002 | 75.60000000000002 | 4.0644 | 4.0644 | 82 | 56 | 0.5 |
| 2018-07-23 12:57:03-05:00 | 35.114262364804745 | -89.93775414302945 | 0.0167 | 75.79999999999995 | 75.79999999999995 | 5.1408 | 5.1408 | 79 | 0 | 0.0 |
| 2018-07-23 12:57:25-05:00 | 35.11423487216234 | -89.9375994130969 | 0.037469999999999996 | 75.79999999999995 | 75.79999999999995 | 6.2136000000000005 | 6.2136000000000005 | 77 | 0 | 0.0 |
| 2018-07-23 12:57:28-05:00 | 35.1142334472388 | -89.93760460987687 | 0.04363 | 75.60000000000002 | 75.60000000000002 | 9.7416 | 9.7416 | 80 | 82 | 0.5 |
| 2018-07-23 12:57:30-05:00 | 35.11422146111727 | -89.93760067038238 | 0.04446 | 75.60000000000002 | 75.60000000000002 | 5.2056 | 5.2056 | 84 | 82 | 0.5 |
| 2018-07-23 12:57:31-05:00 | 35.11421483941376 | -89.93760293349624 | 0.045090000000000005 | 75.60000000000002 | 75.60000000000002 | 4.8024000000000004 | 4.8024000000000004 | 88 | 82 | 0.5 |
| 2018-07-23 12:57:32-05:00 | 35.114215007051826 | -89.93761123158038 | 0.045689999999999995 | 75.60000000000002 | 75.60000000000002 | 4.8024000000000004 | 4.8024000000000004 | 93 | 82 | 0.5 |
| 2018-07-23 12:57:35-05:00 | 35.114211989566684 | -89.93761617690325 | 0.04625 | 75.60000000000002 | 75.60000000000002 | 5.1732000000000005 | 5.1732000000000005 | 96 | 82 | 0.0 |
| 2018-07-23 12:57:42-05:00 | 35.114224730059505 | -89.93762623518705 | 0.047420000000000004 | 75.60000000000002 | 75.60000000000002 | 5.3748000000000005 | 5.3748000000000005 | 99 | 82 | 0.0 |
| 2018-07-23 12:57:45-05:00 | 35.114220371469855 | -89.9376274086535 | 0.04752 | 75.60000000000002 | 75.60000000000002 | 5.3064 | 5.3064 | 103 | 82 | 0.0 |
| 2018-07-23 12:57:46-05:00 | 35.114232525229454 | -89.93764014914632 | 0.049229999999999996 | 75.60000000000002 | 75.60000000000002 | 5.3064 | 5.3064 | 106 | 82 | 0.0 |
| 2018-07-23 12:57:48-05:00 | 35.114244762808084 | -89.93766319938004 | 0.051770000000000004 | 75.39999999999998 | 75.39999999999998 | 5.4072000000000005 | 5.4072000000000005 | 110 | 82 | 0.0 |
| 2018-07-23 12:57:55-05:00 | 35.114275524392724 | -89.93768331594765 | 0.05551 | 75.39999999999998 | 75.39999999999998 | 3.4596 | 3.4596 | 104 | 82 | 0.0 |
| 2018-07-23 12:57:59-05:00 | 35.11427594348788 | -89.93769287131727 | 0.056409999999999995 | 75.79999999999995 | 75.79999999999995 | 2.6856 | 2.6856 | 108 | 82 | 0.0 |
| 2018-07-23 12:58:04-05:00 | 35.11424199678004 | -89.9377032648772 | 0.0577 | 75.0 | 75.0 | 4.5684 | 4.5684 | 111 | 82 | 0.0 |
| 2018-07-23 12:58:07-05:00 | 35.114222299307585 | -89.93770980276167 | 0.06447 | 75.39999999999998 | 75.39999999999998 | 8.967600000000001 | 8.967600000000001 | 108 | 82 | 0.0 |
| 2018-07-23 12:58:15-05:00 | 35.11421668343246 | -89.9377151671797 | 0.0866 | 75.39999999999998 | 75.39999999999998 | 10.4148 | 10.4148 | 112 | 82 | 0.5 |
| 2018-07-23 12:58:16-05:00 | 35.114223388955 | -89.93771784938872 | 0.08949 | 75.39999999999998 | 75.39999999999998 | 10.4148 | 10.4148 | 117 | 82 | 0.5 |
| 2018-07-23 12:58:18-05:00 | 35.11423420161009 | -89.93771650828421 | 0.09529 | 76.60000000000002 | 76.60000000000002 | 10.4472 | 10.4472 | 119 | 82 | 0.5 |
| 2018-07-23 12:58:24-05:00 | 35.11423336341977 | -89.93768306449056 | 0.11247 | 75.39999999999998 | 75.39999999999998 | 10.177200000000001 | 10.177200000000001 | 117 | 82 | 0.5 |
| 2018-07-23 12:58:29-05:00 | 35.11424099095166 | -89.93766068480909 | 0.12649 | 75.20000000000005 | 75.20000000000005 | 10.011600000000001 | 10.011600000000001 | 120 | 82 | 0.5 |
| 2018-07-23 12:58:32-05:00 | 35.114249708130956 | -89.9376569967717 | 0.13474 | 75.0 | 75.0 | 9.8748 | 9.8748 | 124 | 82 | 0.5 |
| 2018-07-23 12:58:37-05:00 | 35.11426311917603 | -89.93763210251927 | 0.14831 | 75.39999999999998 | 75.39999999999998 | 9.6732 | 9.6732 | 119 | 82 | 0.5 |
| 2018-07-23 12:58:38-05:00 | 35.11426420882344 | -89.93763235397637 | 0.15097 | 75.39999999999998 | 75.39999999999998 | 9.5724 | 9.5724 | 116 | 82 | 0.5 |
| 2018-07-23 12:58:39-05:00 | 35.11425976641476 | -89.93762799538672 | 0.15361000000000002 | 75.20000000000005 | 75.20000000000005 | 9.5076 | 9.5076 | 116 | 82 | 0.5 |
| 2018-07-23 12:58:42-05:00 | 35.11425473727286 | -89.93761793710291 | 0.16147 | 75.60000000000002 | 75.60000000000002 | 9.370800000000001 | 9.370800000000001 | 119 | 82 | 0.5 |
| 2018-07-23 12:58:50-05:00 | 35.11425859294832 | -89.93761391378939 | 0.18203999999999998 | 75.39999999999998 | 75.39999999999998 | 9.2052 | 9.2052 | 119 | 82 | 0.5 |
| 2018-07-23 12:58:56-05:00 | 35.11425339616835 | -89.93757527321577 | 0.1972 | 76.0 | 76.0 | 9.036 | 9.036 | 116 | 82 | 0.0 |
| 2018-07-23 12:59:05-05:00 | 35.11426890268922 | -89.93757418356836 | 0.21935 | 76.0 | 76.0 | 8.766 | 8.766 | 118 | 82 | 0.0 |
| 2018-07-23 12:59:14-05:00 | 35.114292204380035 | -89.93758147582412 | 0.24093 | 75.39999999999998 | 75.39999999999998 | 8.564400000000001 | 8.564400000000001 | 118 | 82 | 0.0 |
| 2018-07-23 12:59:23-05:00 | 35.11428541503847 | -89.93757334537804 | 0.2622 | 75.79999999999995 | 75.79999999999995 | 8.532 | 8.532 | 120 | 83 | 0.5 |
| 2018-07-23 12:59:32-05:00 | 35.114290276542306 | -89.93757820688188 | 0.28328 | 76.0 | 76.0 | 8.3628 | 8.3628 | 118 | 82 | 0.5 |
| 2018-07-23 12:59:40-05:00 | 35.11431818827987 | -89.93760226294398 | 0.3015 | 75.79999999999995 | 75.79999999999995 | 8.0604 | 8.0604 | 115 | 82 | 0.0 |
| 2018-07-23 12:59:47-05:00 | 35.11431357823312 | -89.93758885189891 | 0.31710000000000005 | 75.60000000000002 | 75.60000000000002 | 8.0604 | 8.0604 | 118 | 82 | 0.0 |

Figure 11: Garmin Smart-watch .Fit files to .CSV file

## 4.2 Software Description

### 4.2.1 Linux OS in TK1 board

Linux is a free open source operating system and it belongs to UNIX like operating systems. In Linux operating system kernel is the heart of operating system and handles the communication between user and hardware. When comparing Linux and windows system OS, one of the major difference is Linux is an open source project platform and windows is close source platform. The Jetson TK1 is Nvidias embedded Linux platform device on Ubuntu 14.04.

### 4.2.2 Proteus Simulation Software

Proteus Virtual System Modeling(Virtual System Modeling (VSM)) combines mixed mode SPICE circuit simulation, animated components and microprocessor models to facilitate co-simulation of complete micro controller based embedded designs. Proteus comes equipped with a rich suite of instrumentation and analysis tools, from DSO to Logic Analyzer to I2C and SPI Protocol Analyzers. This makes it a mobile electronic workbench and reduces the need for expensive physical equipment.

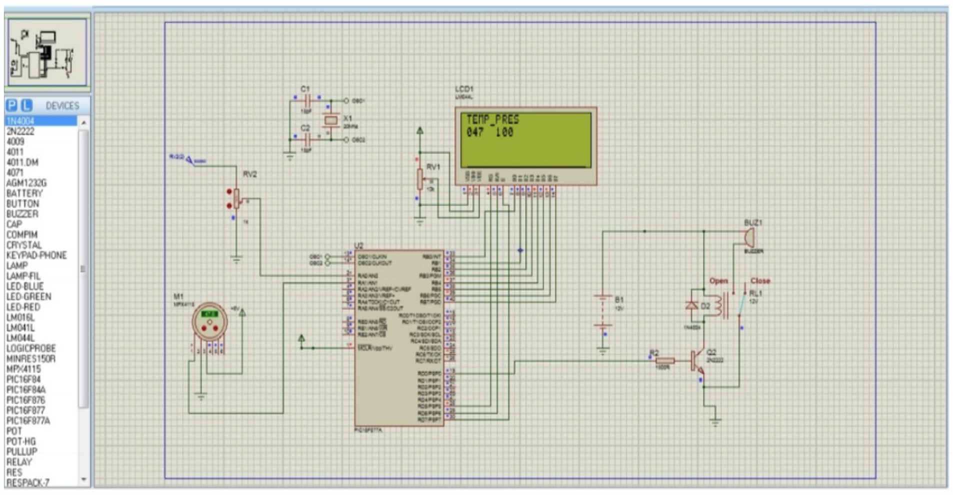


Figure 12: Proteus simulation

### 4.2.3 Flash OS in TK1 board

1. To configure a new board “flash” (wipe it clean and install Linux into it) the Jetson TK1 Board. 2. It flashes all the files into the Jetson TK1, eMMC card of 16GB. 3. It takes roughly 1 hours to flash the whole eMMC. 4. Software development package for the Jetson just like (Compute Unified Device Architecture (CUDA), Open CV, Cudnn etc.) need to install Jetpack (Jetson development package) from the official website.

Directing access the board using from peripheral devices:

1. First things to attach the Jetson board to a HDMI Monitor, Keyboard Mouse to Jetson board. Remote access to Jetson Board from Ethernet cable port:

2. Ethernet cable between host PC and Jetson Board and access it through the network.

[Installation of L4T (Linux for Tegra) onto NVIDIA JETSON TK1 development kit]:

1. Host desktop running on Ubuntu (14.04) is officially recommend and Micro USB Cable provided with Jetson TK1. 2. Download NVIDIA Linux 4Tegra(L4T) board support package(Board Support Package (BSP)) and sample root file (rootfs) from NVIDIA to a host PC (running on the Virtual machine). 3. Extract NVIDIA files (BSP and rootfs). 4. Apply binaries. 5. Connect Jetson TK1 to host computer via the micro-usb cable and place Jetson TK1 to recovery mode.

### 4.2.4 Crontab automation tool

Crontab is a powerful task scheduler, which can be used to perform task automatically. It uses several parameters to perform task automatically.

Crontab can be displayed and edited by using commands in terminal. Cronjobs are commands that runs at specified interval of a time. Cron job executing multiple sensor scripts simultaneously at particular interval of a time for capturing surrounding data inside Jetson TK1 board.

1. crontab -l , which display the cron table.
2. crontab –e ,which opens the crontable editor.
3. crontab -r ,which remove the crontable task.
4. crontab -v ,which display last time when you edited crontab file.

### 4.2.5 Python modules, drivers and packages

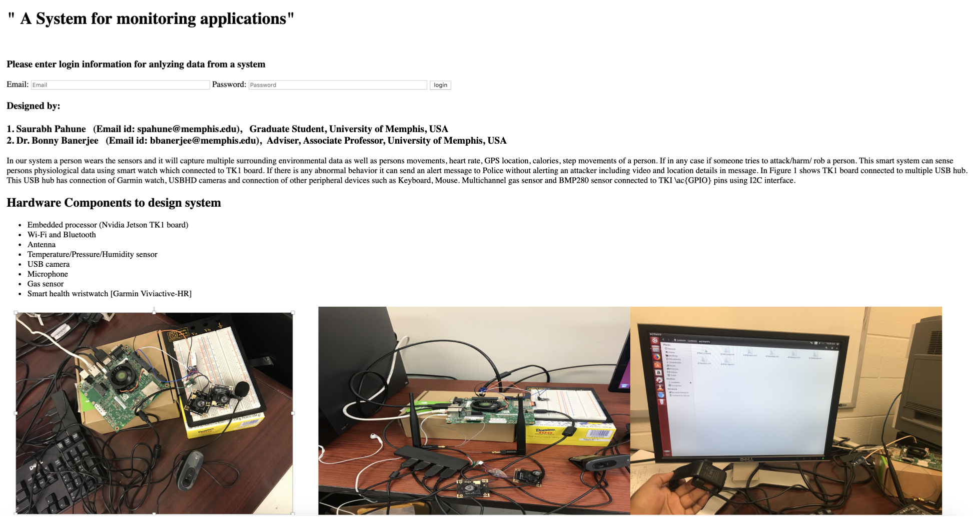
Python is a flexible and powerful programming language and it’s easy to learn and follow. The clear syntax of python makes a valuable tool for users who wants to learn programming. Python runs on Linux OS, Windows, MacOS. For instance, python is addressed to use specific hardware such as Nvidia’s GPIO port. Many single board computers such as Raspberry pi, Arduino, Intel Edison prefer the Python language.

# 5 Wi-Fi/Bluetooth networking protocol

## 5.1 Intel 7260 Wi-Fi/Bluetooth card driver in TK1

Jetson TK1 board has Wifi can be added using Minipcie card slot in TK1 board. In our system we have used wifi card intel 7260 for wireless connection. To work wifi generally it needs two things in order to work such as: Antenna pairs require to connect into network, wifi ['iwlwifi'] driver which supports linux kernel. 'iwlwifi' is a wireless driver for intel wireless chip. Here, antenna pair has two pigtail adapters to connect to tiny antenna ports on Wifi card with standard antenna mount which shown in given figure.

## 5.2 Live streaming introduction and display data on Web page


" A System for monitoring applications"

Please enter login information for anlyzing data from a system

Email: Email Password: Password login

Designed by:

1. Saurabh Pahune (Email id: spahune@memphis.edu), Graduate Student, University of Memphis, USA
2. Dr. Bonny Banerjee (Email id: bbanerjee@memphis.edu), Adviser, Associate Professor, University of Memphis, USA

In our system a person wears the sensors and it will capture multiple surrounding environmental data as well as persons movements, heart rate, GPS location, calories, step movements of a person. If in any case if someone tries to attack/harm/ rob a person. This smart system can sense persons physiological data using smart watch which connected to TK1 board. If there is any abnormal behavior it can send an alert message to Police without alerting an attacker including video and location details in message. In Figure 1 shows TK1 board connected to multiple USB hub. This USB hub has connection of Garmin watch, USBHD cameras and connection of other peripheral devices such as Keyboard, Mouse. Multichannel gas sensor and BMP280 sensor connected to TKI \ac{GPIO} pins using I2C interface.

Hardware Components to design system

- Embedded processor (Nvidia Jetson TK1 board)
- Wi-Fi and Bluetooth
- Antenna
- Temperature/Pressure/Humidity sensor
- USB camera
- Microphone
- Gas sensor
- Smart health wristwatch [Garmin Viviactive-HR]

Figure 13: Front end webpage

In above figure, a screen shot of a Web page. Web page made up of HTML,CSS languages. Whatever data is captured from the sensors, it will be store on server so that anyone can access or watch data capture by TK1 board. But to access data on server,it must needs login id and password to make data confidential and secure hence authenticate person can see data on webpage.

### Live stream data of all sensors where all data can access using RTSP

RTSP is a real time stream protocol. Using IP address and port number, we can access all data at any mobile device or PC. To run the Gst-rtsp server-1.8.1 below three process require for installation in host PC of Jetson TK1. Here rtsp [Real time stream protocol] server is used for live streaming. This server uses 8554 port number.

### 1. Gstreamer-1.8.0.tar.xz 2. Gst-plugins-base-1.8.0.tar.xz 3. Gst-rtsp-server-1.8.1.tar.xz

Need to install above three packages on Jetson TK1. Gstreamer [**Gst**] is a pipeline based multimedia [Video playback, audio playback, live streaming etc.] framework that links together a wide variety of media processing system. Gstreamer has different versions Gstremaer-0.10, Gstreamer-1.0, Gstreamer-1.8.

# 6 Installing process of Gstremaer-1.80:

1. sudo tar xpf gstreamer-1.8.0.tar.xz
2. cd gstreamer -1.8.0
3. ./configure –libdir=/usr/lib/arm-linux-gnueblihf
4. make
5. sudo make install

## 6.1 Installing process of Gst-plugins-base-1.80:

1. sudo tar xpf gst-plugins-base-1.8.0.tar.xz
2. cd gstreamer -1.8.0
3. ./configure –libdir=/usr/lib/arm-linux-gnueblihf
4. make
5. sudo make install

## 6.2 Installing process of Gst-rtsp-server-1.80

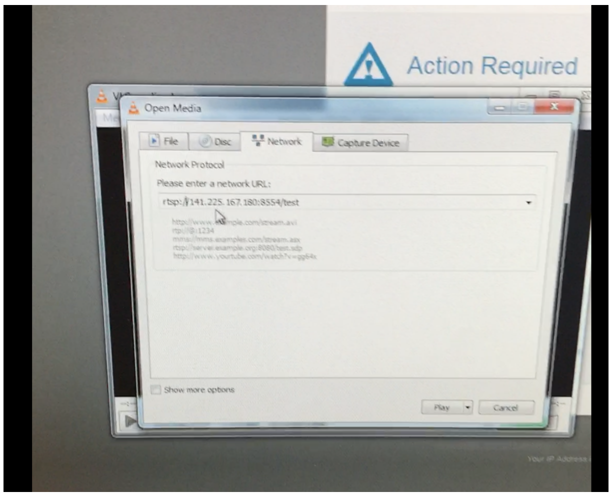
1. sudo tar xpf gst-rtsp-server-1.8.0.tar.xz
2. cd gstreamer -1.8.0
3. ./configure –libdir=/usr/lib/arm-linux-gnueblihf
4. make
5. sudo make install

**Once all three packages are install then need to run test app inside rtsp server installation folder:**

1. cd gst-rtsp-server -1.8.1
2. cd examples
3. ./test-launch"v4l2src-device=/dev/video2!video/x-raw,format=YUY2,width=640,height=480 ! videoconvert ! video/x-raw,format=I420 ! omxh264enc ! rtph264pay name=pay0 pt=96"

4. Above command to run the test file inside rtsp server with YUY2 pixel format in video capturing camera with h.264 encoder.

5. **Real Time Stream Protocol (rtsp)://IP address of TK1 board:8554/test** [This command shows stream ready for the live streaming-run this in client PC we can get live streaming of a camera]

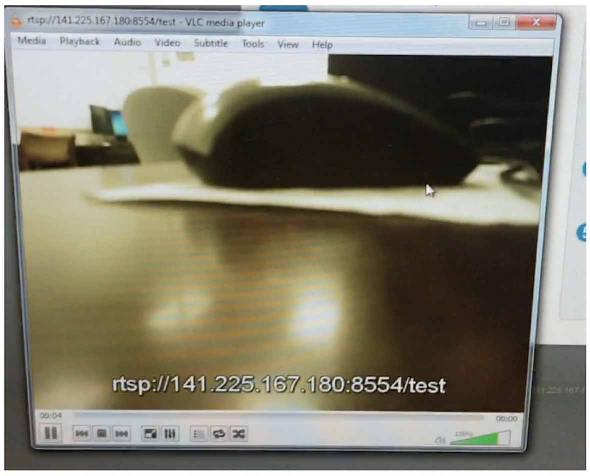


Network Protocol

Please enter a network URL:

rtsp://141.225.167.180:8554/test

Figure 14: Client PC stream ready to get live streaming



rtsp://141.225.167.180:8554/test - VLC media player

rtsp://141.225.167.180:8554/test

Figure 15: Client PC live streaming of Camera view using rtsp [Real time stream protocol]

# 7 Evaluation of System

CPU is the main chip inside the computer responsible for day to day activity. It run all application program. Heat is a natural product of electricity, as components inside the computer such as CPU, GPU etc. electricity carried across this circuits also creates heat.

As CPU operates on clock speed/ clock frequency, sometimes operating CPU at higher clock speed generates excessive heat [CPU temperature/heat increases]. In an effort to lower the heat build up in CPU, crystal oscillator inside components is under-clocked [Lower the CPU frequency] but naturally it decreases system efficiency. Therefore, to cool down CPU temperature it has internal fan uses to cooling down the excessive heat [dissipate the heat].

As it gets heater(CPU) it lowers the clock speed called under clock, using crystal oscillator mechanism. In whole processes called thermal throttling. Thermal throttling is the process to protect chip out of heat damage in case of a without fan. In presence of fan overclock and under clock results are clipped because of internal fan which cool down the CPU temperature. In this case temperature inside chip not get high because of cooling fan and due to which there is a very less chance of thermal throttling. If temperature increases beyond capabilities the board gets automatically shut down.

## 7.1 Time vs CPU temperature results in TK1 with/without Fan with all sensor

In this system, we have calculated CPU temperature characteristics of a Jetson TK1 board with respect to by running multiple sensors [BMP280,Microphone, Gas sensor, multiple lens camera] running in TK1 board with different frame per seconds of a camera [@10fps,@20fps,@30fps,@40fps]and analyze the CPU temperature with fan and without fan of a TK1 board.

## 7.2 Time vs CPU temperature results in TK1 with/without Fan with all sensor with respect to 10,20,30,50fps

In this system, we have calculated CPU temperature characteristics of a Jetson TK1 @10fps and analyze the CPU temperature with fan and without fan of a TK1 board.

In 10 fps@ multiple data collection using sensor CPU temperature is **maximum 40°C** in case of without fan.

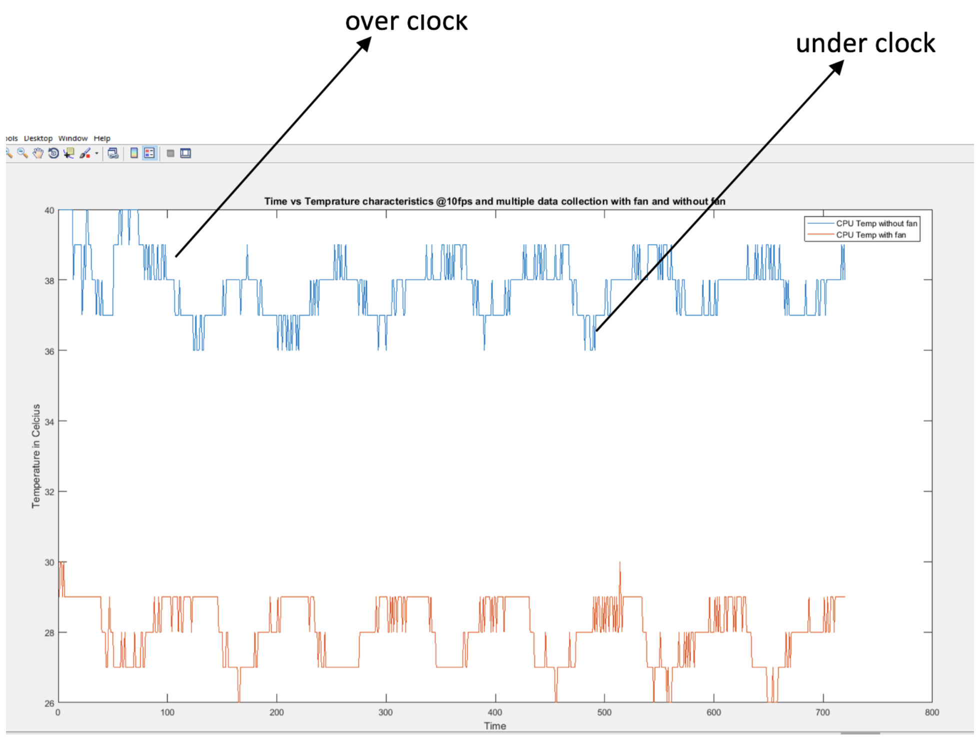


Figure 16: Time vs CPU temperature characteristics @10fps with multiple sensor data collection with respect to with fan and without fan

In 20 fps@multiple data collection sensor CPU temperature **maximum 42°C** in case of without fan. Inside Jetson TK1 the video decoder engine implementing video codec in H.264, or MPEG4 video file and also processing audio files. In our case we are capturing 3 videos simultaneously every minute with 20fps for duration 5 second video, which also causes more temperature dissipation in TK1 board with respect to with fan and without fan cases. In every mechanism PLL clock domain inside Jetson TK1 board “Phase lock loop” are responsible for clock and clock speed for multiplication and division.

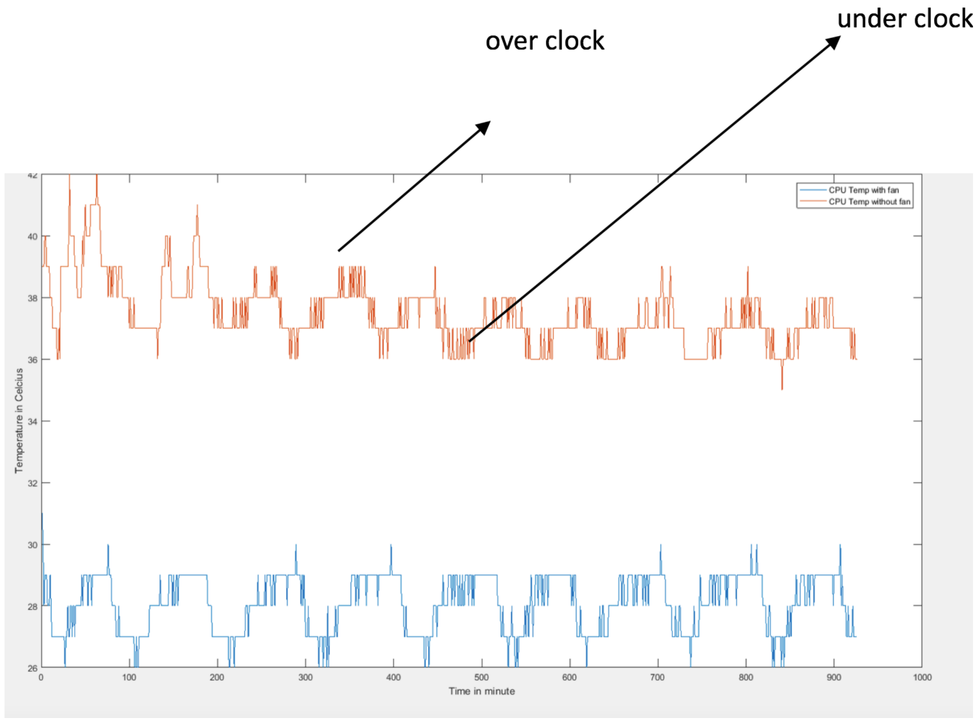


Figure 17: Time vs CPU temperature characteristics @20fps with multiple sensor data collection with respect to with fan and without fan

In Fig:18, CPU generates overclock frequency (higher heat) and under clock frequency (to lower the heat) in case of without fan plot. In presence of fan overclock and under clock results are clipped because of internal fan which cool down the CPU temperature. In 30 fps@multiple data collection sensor CPU temperature **maximum 45°C**.

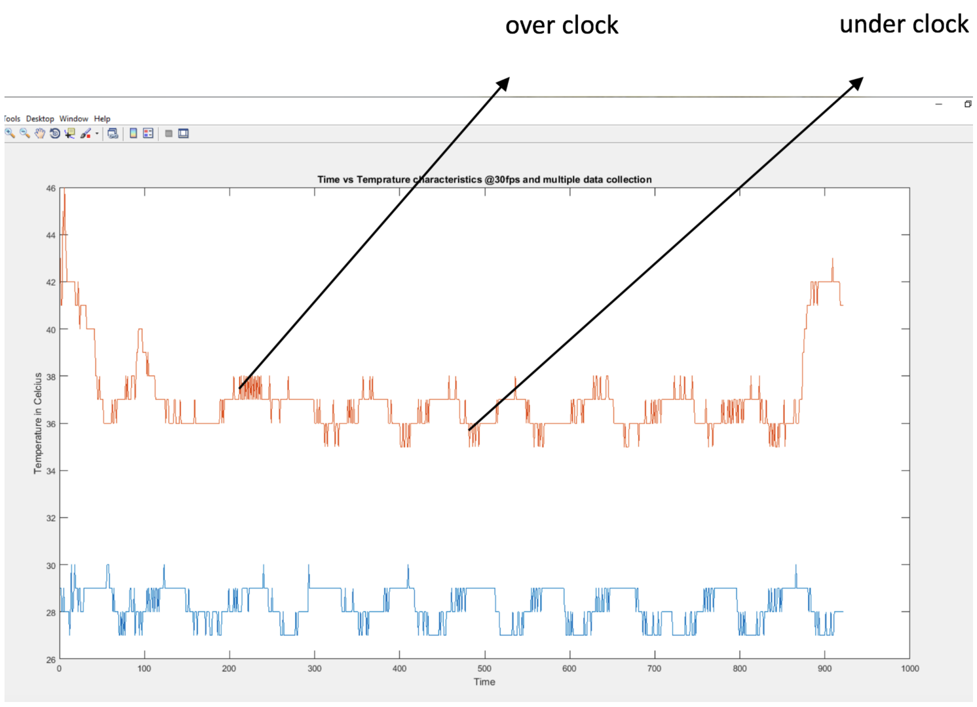


Figure 18: Time vs CPU temperature characteristics @30fps with multiple sensor data collection with respect to with fan and without fan

In Fig:19, CPU generates overheating system (higher heat) and throttling CPU frequency (to reduce the heat) in case of without fan plot [red plot shows in graph]. In presence of fan overclock and under clock results are clipped because of internal fan which cool down the CPU temperature. In 50 fps@multiple data collection sensor CPU temperature **maximum 47°C**.

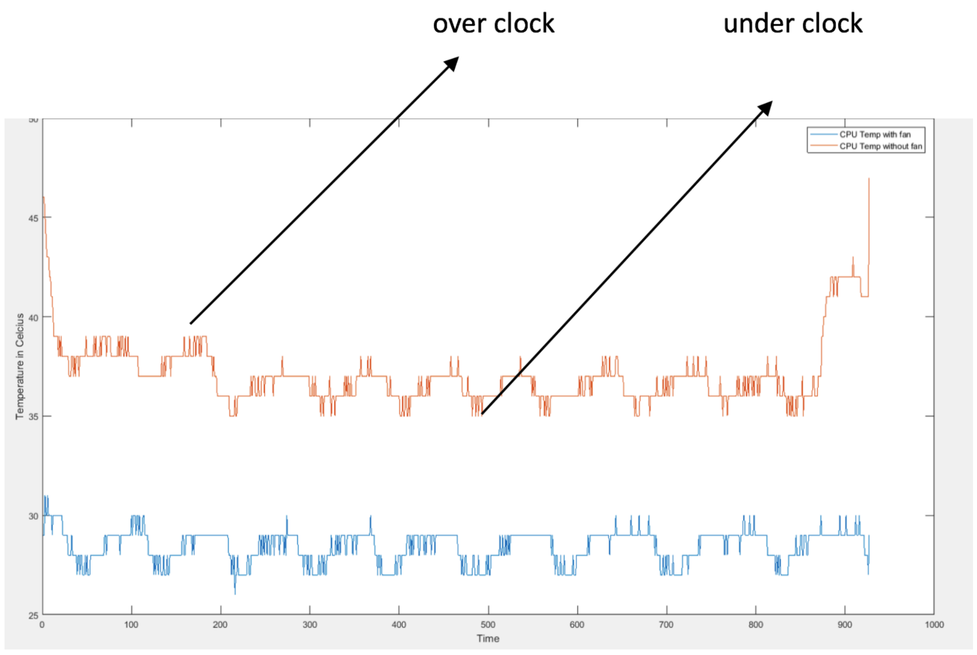


Figure 19: Time vs CPU temperature characteristics @50fps with multiple sensor data collection with respect to with fan and without fan

| fps (frames per second) | CPU Temp max |
|---|---|
| @50 fps | 47 °C |
| @30 fps | 45 °C |
| @20 fps | 42 °C |
| @10 fps | 40 °C |

Table 1: CPU temperature in absence of a Fan in TK1 board

By analyzing the results and above table, it can be seen that as fps of camera increases, CPU temperature of a TK1 board also increases.

## 7.3 Memory card vs Writing speed results in TK1 with different fps

In each case of every minute while computing @50fps, Jetson TK1 all thermal zones are changed and produced more heat generation in CPU temperature, board temperature which causes more heat while computing more FPS (frames per second) data. As heat generated increases, it makes system sluggish due to which @50fps reading speed and writing speed takes more time and vice-versa.When we analyzed result it shows while computing data@50fps at every minute of time causes more heat generation and has more power consumption.

### 7.3.1 Writing speed characteristics of 10fps frames video for three lenses [ 2Dual lens camera, Logitech cam] encoded in TK1 board to capture video of 5 seconds and stored in SD card

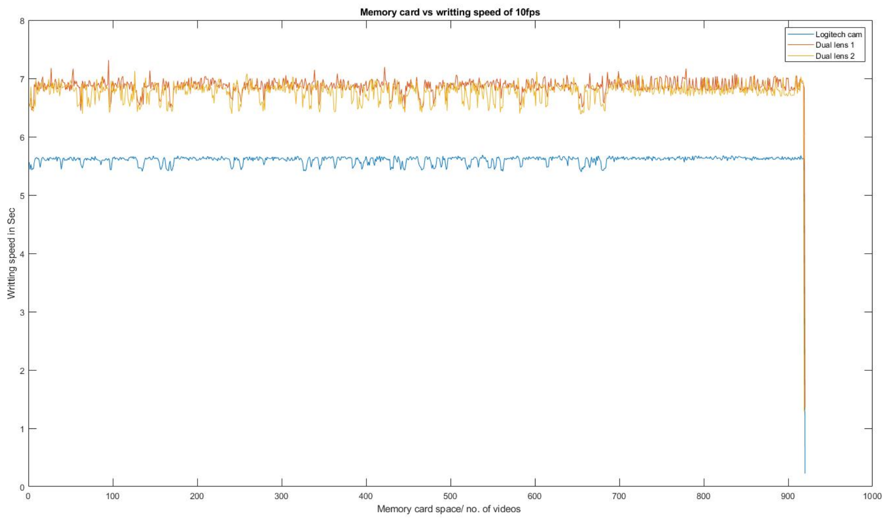


Figure 20: Memory card vs Writing speed characteristics@10fps

In Fig:20, 10fps has: 930 videos [after that memory card get full occupied] each video has each size of 11.3 MB. Which has less size of video which shows less writing speed to store/write into SD card in between 5.5 to 7.2 MB/sec. In above graph we used 2Dual lens camera and one Logitech webcam. The two dual lens camera have almost same hardware specifications hence their writing speed calculation almost equal @10fps according to above result. While computing data @10fps which causes less thermal zone effects in all parts of Jetson TK1 board on chip system [CPU temperature, GPU and tegra board temperature]. While capturing video@10fps inside video encoder of TK1 implementing H.264, MPEG format generated of 11.5 MB video size for each lens which takes less time to write data into SD Card.

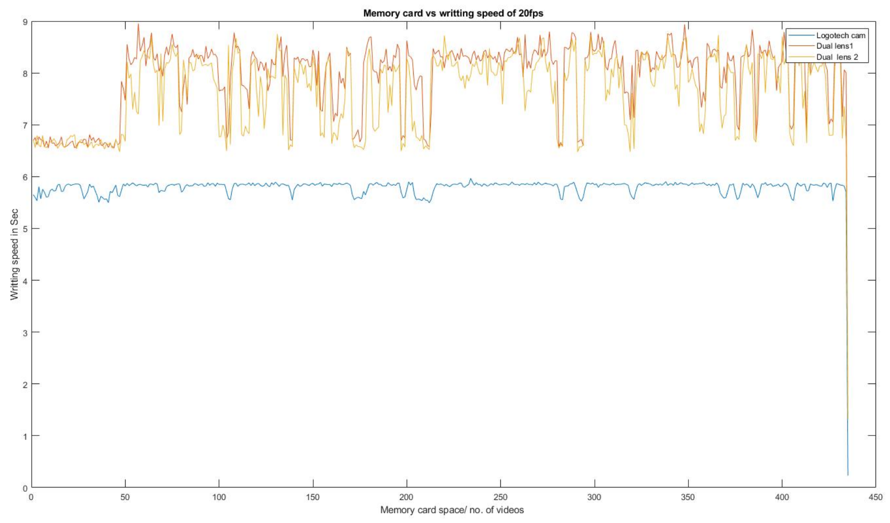


Figure 21: Memory card vs Writing speed characteristics@20fps

In Fig:21, 20fps has: 420 videos [after that memory card get full occupied] each video has each size of 22.3 MB. Which has less size of video than 30fps,50fps which shows less writing speed comparatively 30fps,50fps to store/write into SD card in between 5.8 to 9MB/sec. In above graph we used 2Dual lens camera and one Logitech webcam. The two dual lens camera has almost same hardware specifications hence their writing speed calculation almost equal according to above result.

While computing data @20fps which causes less thermal zone effects than [@30,50 fps] in all parts of Jetson TK1 board on chip system [CPU temperature, GPU and tegra board temperature]. While capturing video@20fps inside video encoder of TK1 implementing H.264, MPEG format and generated of 22.5 MB video size for each lens which takes less time to write data into SD Card as compare to more computing data on @30,50fps video.

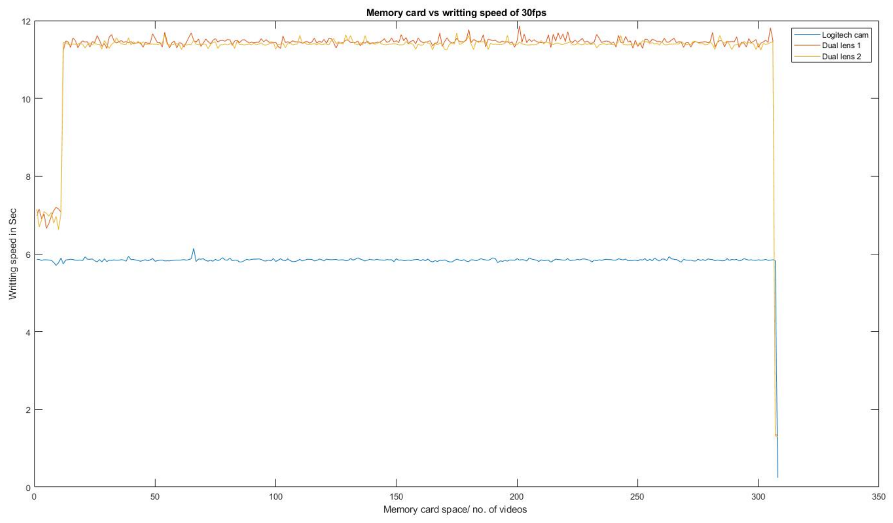


Figure 22: Memory card vs Writing speed characteristics@30fps

In Fig:22, 30fps has: 310 videos [after that memory card get full occupied] each video has each size of 34.5 MB. Which has less size of video than 50fps which shows less writing speed comparatively 50fps to store/write into SD card in between 5.8 to 12MB/sec. In above graph we used 2Dual lens camera and one Logitech webcam. The two dual lens camera has almost same hardware specifications hence their writing speed calculation almost equal according to above result.

While computing data @30fps which causes less thermal zone effects than [@50 fps] in all parts of Jetson TK1 board on chip system [CPU temperature, GPU and tegra board temperature]. While capturing video@30fps inside video encoder of TK1 implementing H.264, MPEG format and generated of 33.5 MB video size for each lens which takes less time to write data into SD Card as compare to more computing data on @30fps video.

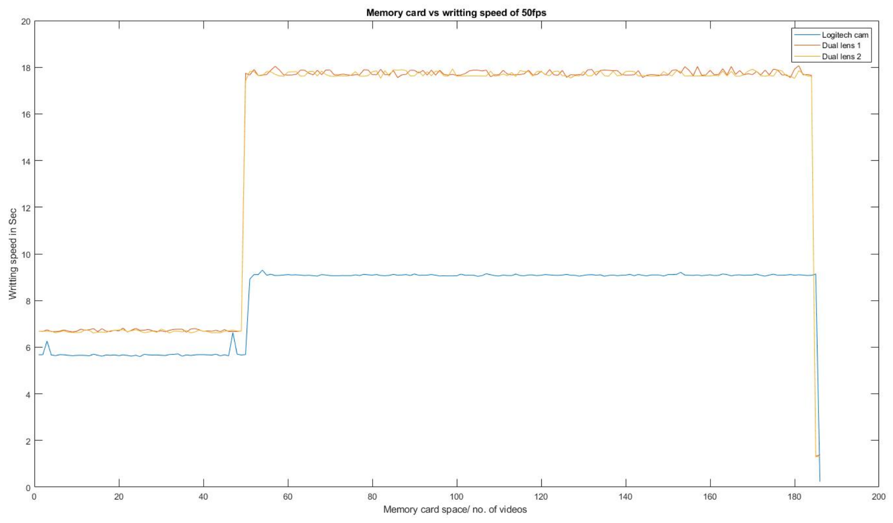


Figure 23: Memory card vs Writing speed characteristics@50fps

In Fig:23, 50fps has: 190 videos [after that memory card get full occupied] each video has each size of 53.3 MB. Which has larger size of video which shows writing speed to store/write into SD card in between 5.9 to 18MB/sec. In above graph we used 2Dual lens camera and one Logitech webcam. The two dual lens camera has almost same hardware specifications hence their writing speed calculation almost equal according to above result. While computing data @50fps which causes more thermal zone effects than in all parts of Jetson TK1 board on chip system [CPU temperature, GPU and tegra board temperature]. While reading and capturing video@50fps inside video encoder of TK1 implementing H.264, MPEG format and generated of 55.5 MB video size file for each lens which takes more time to write data into SD Card as compare to @10,20,30 fps computing data.

Writing speed of combining all 10fps,20fps,30fps,50fps for three lenses to capture video of 5 seconds and stored in SD card.

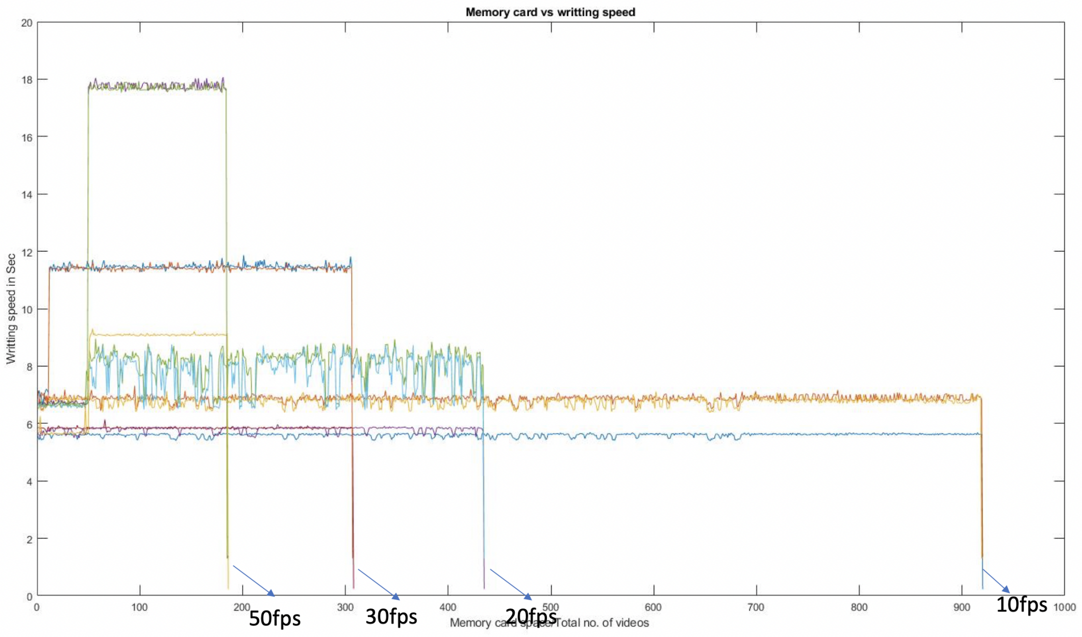


Figure 24: Memory card vs Writing speed characteristics@10,20,30,50fps

| fps (frames per second) | Writing speed (sec) |
|---|---|
| 10 | 5.5 to 7.2 sec |
| 20 | 5.8 to 9 sec |
| 30 | 5.8 to 12 sec |
| 50 | 5.9 to 18 sec |

Table 2: fps vs writing speed

From above table it reflects that as reading and computing on data @ 50 fps causes more heat rise in all thermal zones of Jetson TK1 board which makes system sluggish and takes more writing speed time as more data of fps computing by Jetson TK1 board.

## 7.4 Memory card vs Reading speed results in TK1 with different fps

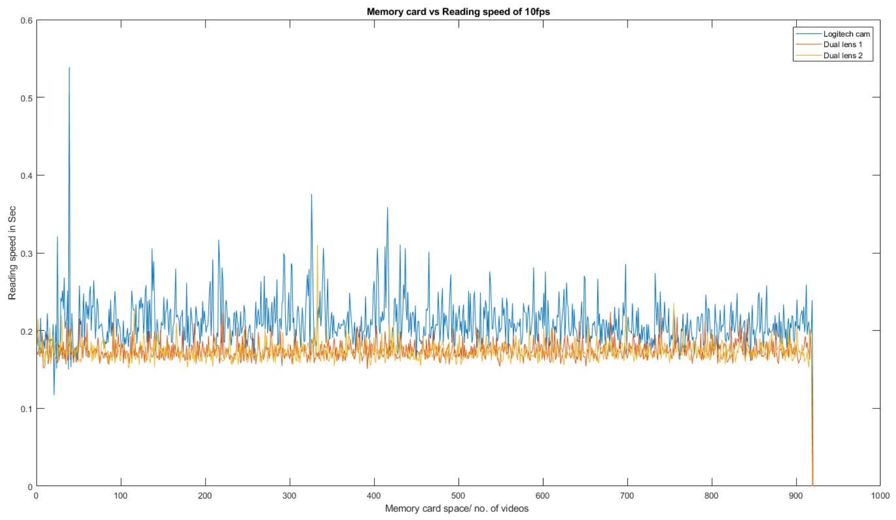


Figure 25: Memory card vs Reading speed characteristics@10fps

In Fig:25, 10fps has: 930 videos to read each video has size of 11.3 MB. Each of video has less size which shows less reading speed from SD card in between 0.1 to 0.38 second speed.

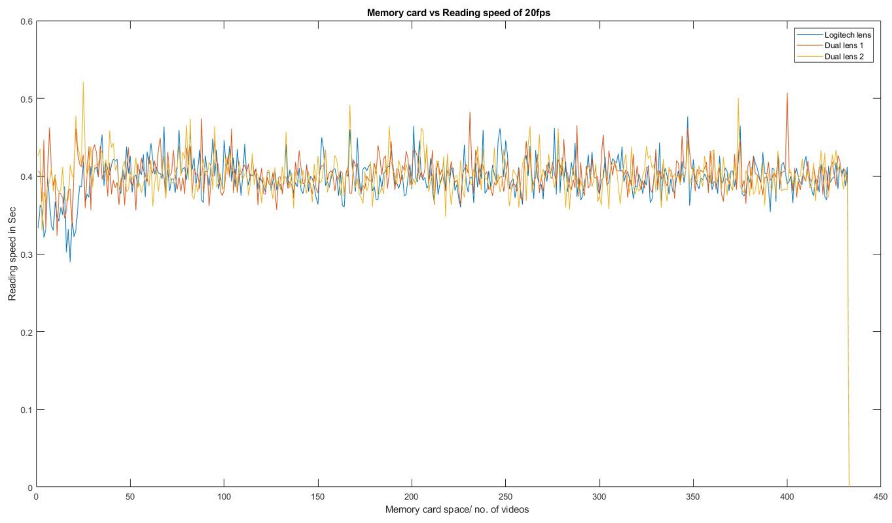


Figure 26: Memory card vs Reading speed characteristics@20fps

In Fig:26, 20fps has: 420 videos to read each video has size of 22.3 MB. Each of video has less size than 30,50 fps which shows less reading speed than 30,50fps from SD card in between 0.3 to 0.5 second speed.

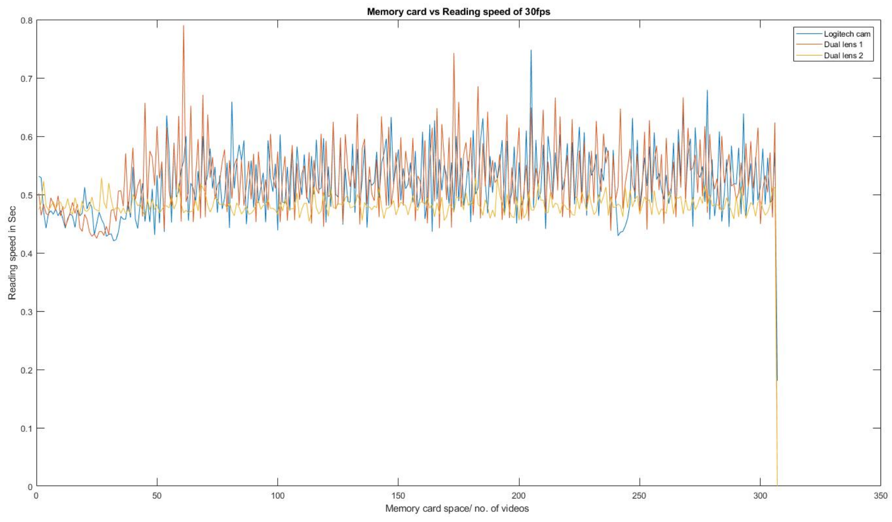


Figure 27: Memory card vs Reading speed characteristics@30fps

In Fig:27, 30fps has: 310 videos to read each video has size of 34.5 MB. Each of video has less size than 50 fps which shows less reading speed than 50fps from SD card in between 0.4 to 0.78 second speed.

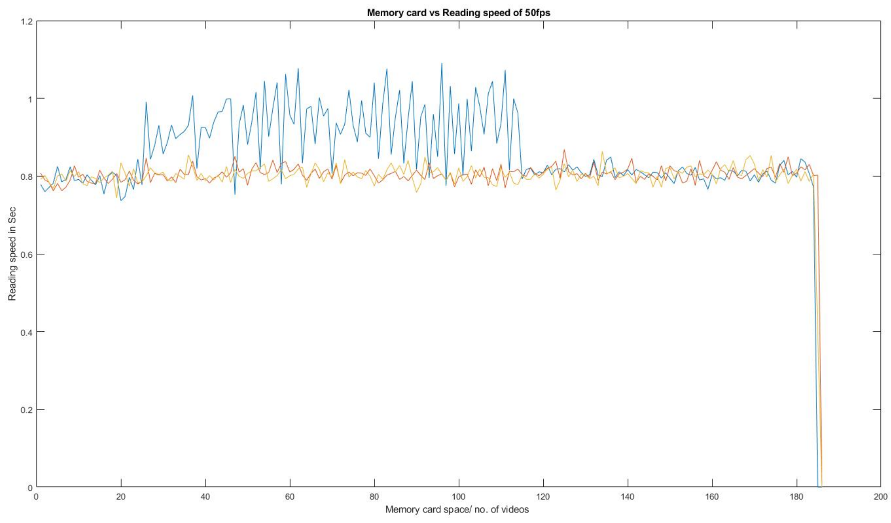


Figure 28: Memory card vs Reading speed characteristics@50fps

In Fig:28, 50fps has: 190 videos to read each video has size of 55.3 MB. Each of video has bigger which shows more reading speed than 50fps from SD card in between 0.79 to 1 second speed.

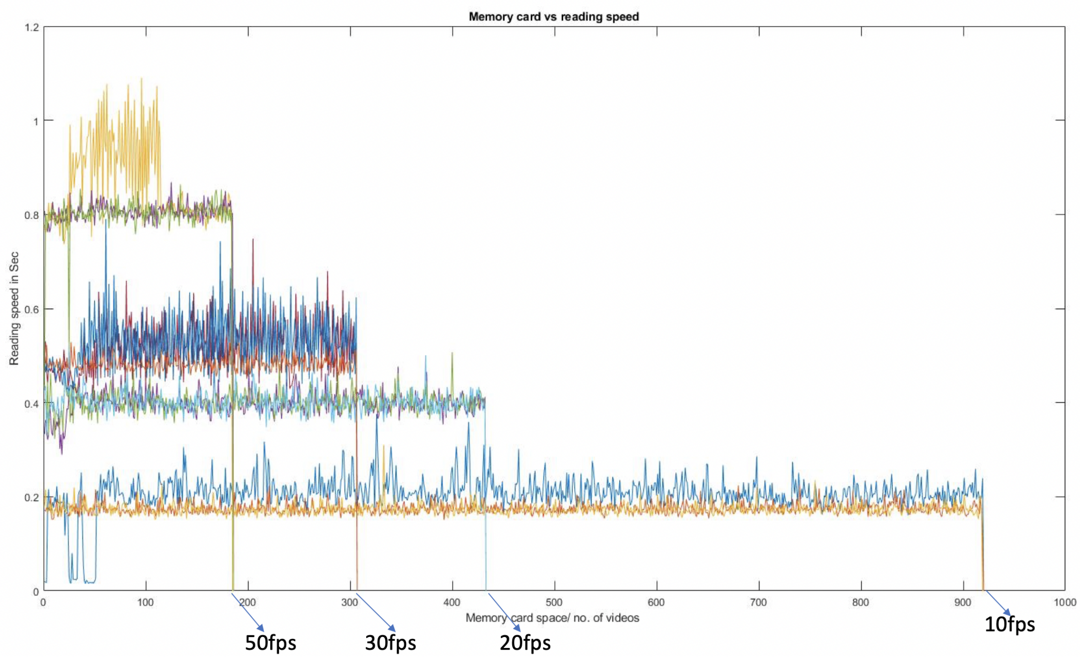


Figure 29: Memory card vs Reading speed characteristics@10,20,30,50fps

| fps (frames per second) | Reading speed (sec) |
|---|---:|
| 10 | 0.1 to 0.38 |
| 20 | 0.3 to 0.5 |
| 30 | 0.4 to 0.78 |
| 50 | 0.79 to 1 |

Table 3: fps vs reading speed

## 7.5 Time vs Temperature-Humidity results in TK1

This graph shows, as relative humidity changes temperature also changes. Which conclude that as relative humidity falls, temperature rises and vice versa. Blue-line shows temperature and red line indicate humidity. Here we have plotted relation between temperature and humidity by collecting atmospheric data of a temperature and humidity using BMP280 sensors via Jetson TK1 board. x axis plot shows time collection of a data every minute while y axis plot shows temperature and humidity plot. Here temperature reading measure in Fahrenheit

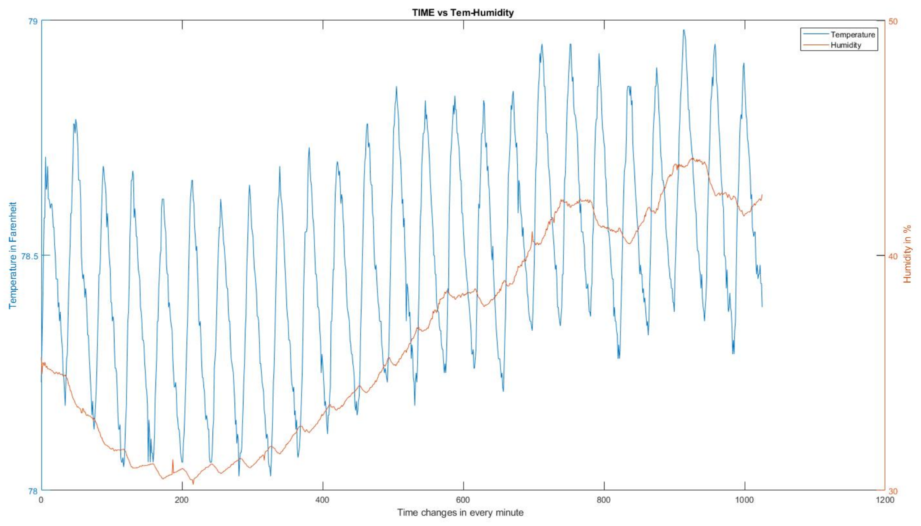


Figure 30: Time vs Temperature-Humidity characteristics

.

## 7.6 Time vs Pressure-Humidity results in TK1

Humidity varies with temperature, this plot shows a relationship between pressure and humidity by plotting data by using BMP280 sensor for data collection using Jetson TK1 board. Here pressure measured in hPa [hectopascals] unit.

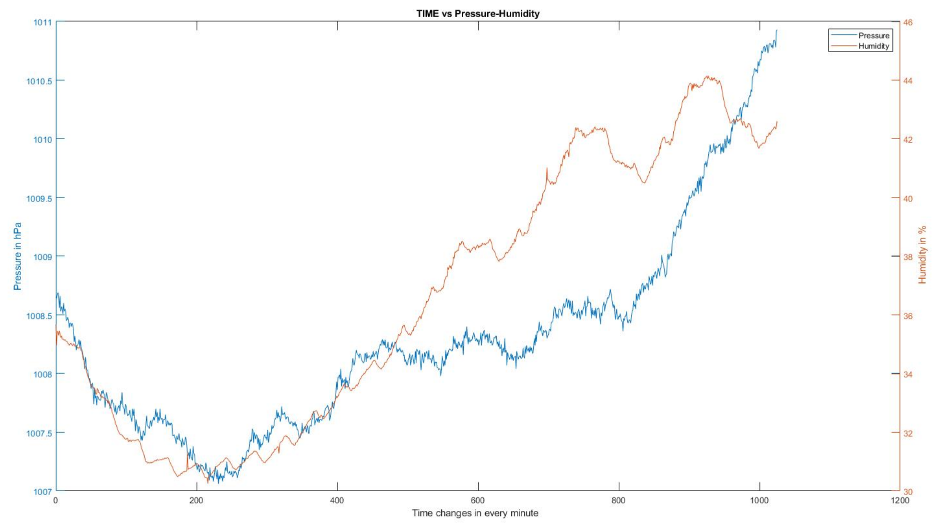


Figure 31: Time vs Pressure-Humidity characteristics

## 7.7 Time vs Pressure-Temperature results in TK1

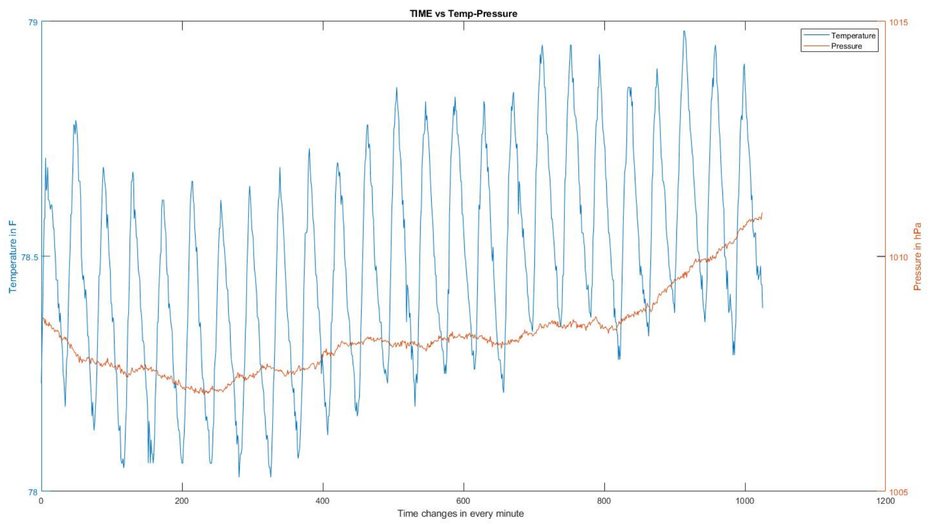


Figure 32: Time vs Temperature-Pressure characteristics

According to plot relation, temperature and pressure are inversely proportional to each other. Which shows that as temperature increases and pressure decreases and vice-versa. Here we collected data from atmosphere by measuring value of temperature and pressure using Bmp280 sensor inside TK1 board. Here pressure measured in hPa [hectopascals] unit. and Temperature in Fahrenheit unit.

## 7.8 Each CPU cores [core0,1,2,3] frequency results w.r.t different fps and all sensor

TK1 CPU has two clusters [4 +1] core configuration:

**1.High-performance [HP] clusters:**

HP clusters consist of four Cortex –A15 core processor [Core 0,1,2,3]. • Three of cores can be individually shut down. • All the sensors and application distributed all workload by CPU clusters [Core 0,1,2,3]. • The Jetson TK1 has 22 configurable CPU frequency [between 51MHZ to 2.32GHZ/2300000].

**2.Low-performance [LP] clusters:**

LP clusters with single Cortex-A 15 core processor. • This is a low power companion core. • 1GHz Cortex-A15 core.

$$P_{core} = \alpha C V_{core}^2 f_{core} \tag{1}$$

According to above formula processor frequency can be responsible to increase power consumption.[1]

Where,

C= core switching capacitance.
V$_{core} = corevoltage.$
$f_{core} = corefrequency.$
$\alpha = coreutilizationlevel.$

To reduce power draw by Jetson board use this below 4 steps:

1. Follow the various sections in Controlling the CPU performance to reduce the CPU speed of the 4 cores [core frequency] to be as low as possible for your application.
2. Follow Controlling GPU performance to reduce the GPU speed as much as is possible for your application.
3. Also, if you don't need anything displayed on a screen then make sure nothing is plugged into your HDMI port, and possibly also disable it.
4. And if you won't be running intense code for long periods, then replace the fan with a heatsink or heat spreader.
5. In my case I am running hardware platform of TK1 board with different applications. But default setting of Jetson TK1 gives high performance and produce less power draw in light tasks.
6. In some cases, if you want to run higher performance of board or run as lower power draw.
7. The automatic turning on/off of the 4 main CPU cores and 5th companion core is mostly done in the L4T kernel using "cpuquiet", a mechanism for dynamically hot-plugging CPU cores based upon workload/policy.

8. Limit the max clock-rate of the 4 main CPU cores to a low speed to reduce the power (it will still automatically switch them on/off and switch to the 5th low-power companion core when suitable).

9. Turn some CPU cores on and some off, based on what you know works best for a particular use-case.

## 7.9 Variations of Core frequency with different fps:

### 7.9.1 While running all sensors simultaneously and computing data with different fps and we calculated each core [core 0,1,2,3] frequency in TK1 board:

Here we evaluated each core individual frequency while running all sensors simultaneously and computing data @50fps and audio data process to check whether this core frequency will responsible for heat generation or it shows that whether this board consumes more energy consumption or not.

As CPU distributed work load in the form cores [Core0,1,2,3]. If more background application running in that case all cores can distributed workload of a system.

$$P_{core} = \alpha C V_{core}^2 f_{core} \tag{2}$$

Where,

C= core switching capacitance.
V$_{core} = corevoltage.$
$f_{core} = corefrequency.$
$\alpha = coreutilizationlevel.$

**While running all sensors simultaneously and computing data @50fps and we calculated core 0 frequency:**

1. The Jetson TK1 board has 22 configurable frequencies in range from 51MHz to 2.3 GHz.

2. The above plot shows the core 0 distributed frequencies from this plot it concludes that minimum frequency of core 0 goes to 1224 KHz [1224000 Hz] and maximum frequency of core 0 goes to 2320 KHz [2320000 Hz].

3. In our system core 0 frequency ranges from [1224KHz to 2320KHz], which is totally depends on running all sensors and computing data @50 fps and audio data process.

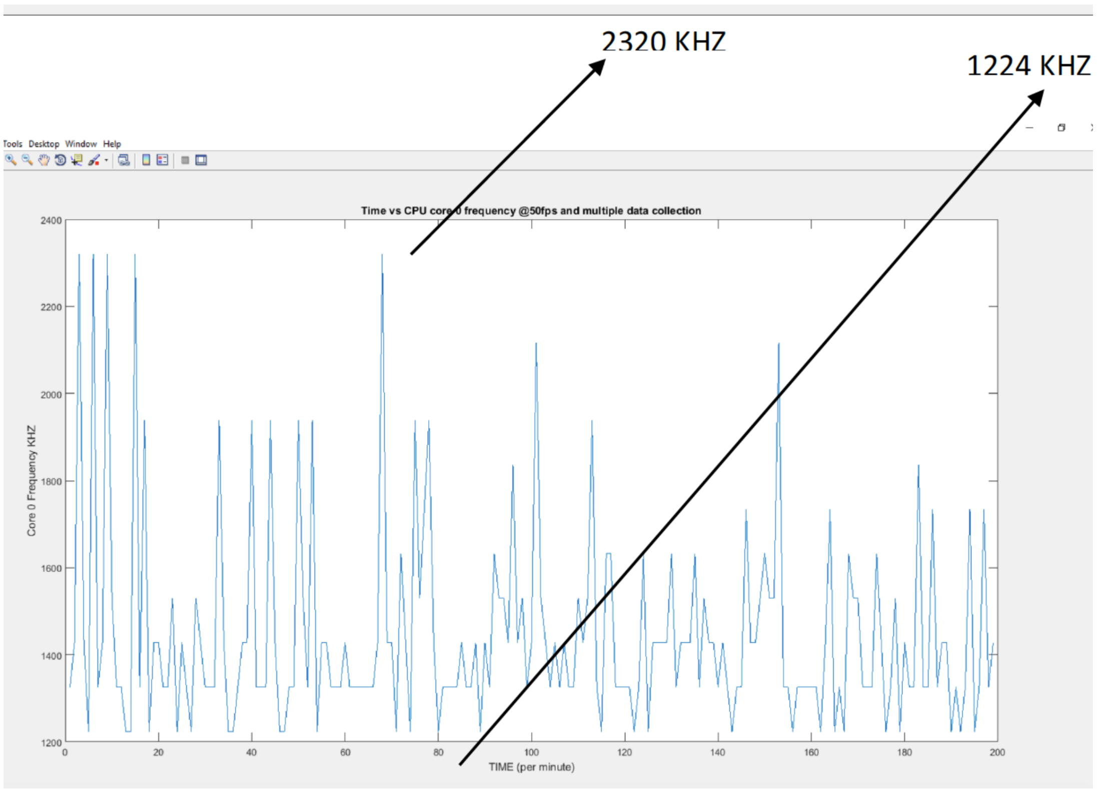


Figure 33: CPU core 0 frequency operation with respect to time@50fps

**While running all sensors simultaneously and computing data @50fps we calculated core 1 frequency:**

1. The above plot shows that core 1 distributed frequencies from this plot it concludes that minimum frequency of core 1 goes to 1224 KHz [1224000 Hz] and maximum frequency of core 1 goes to 2320 KHz [2320000 Hz].

2. The above plot shows same minimum and maximum frequency in both Core 0 and Core 1. It proves that while running our system [multiple sensor data, computing data@50fps and audio data to process] both core 0 and core 1 uses same distributed load work.

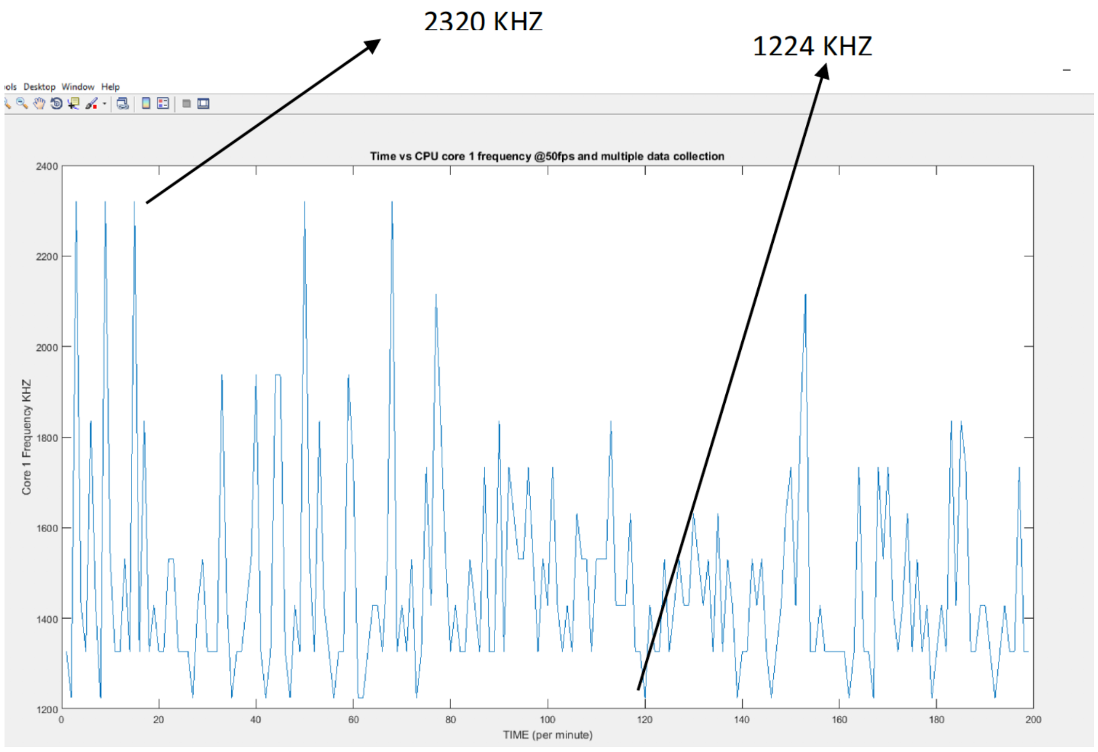


Figure 34: CPU core 1 frequency operation with respect to time@50fps

**While running all sensors simultaneously and computing data @50fps we calculated core 2 frequency:**

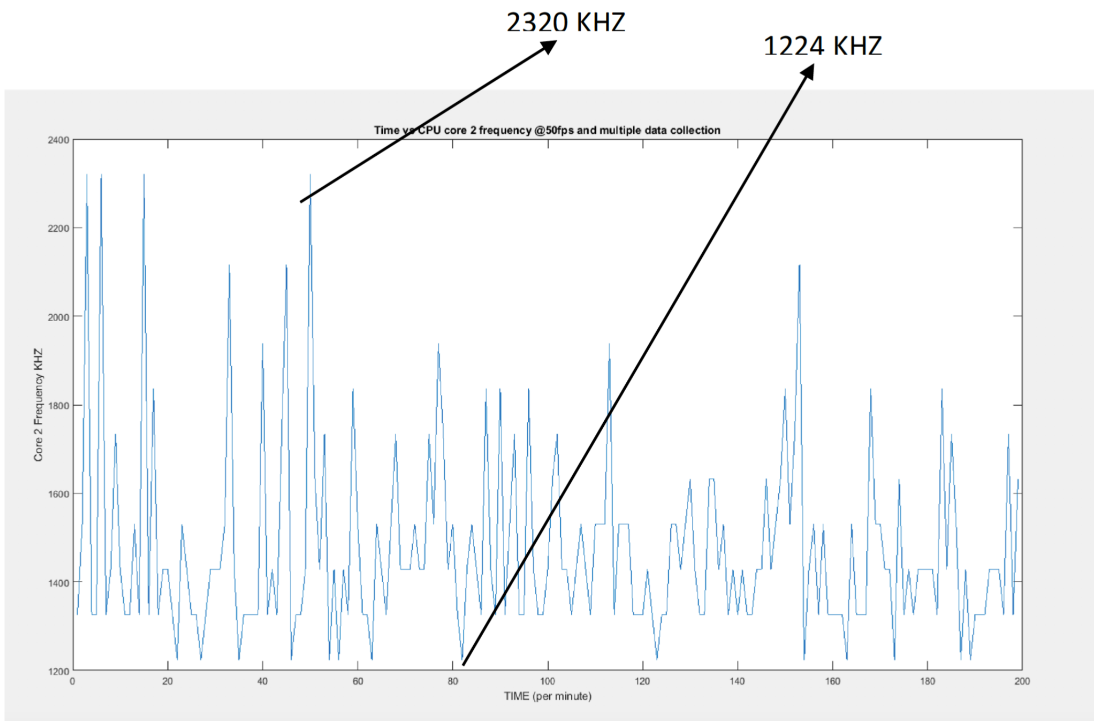


Figure 35: CPU core 2 frequency operation with respect to time@50fps

1. The above plot shows that core 2 distributed frequencies from this plot it concludes

that minimum frequency of core 2 goes to 1224 KHz [1224000 Hz] and maximum frequency of core 2 goes to 2320 KHz [2320000 Hz].

2. The above plot shows same minimum and maximum frequency in both Core 0 and Core 1 and core2. It proves that while running our system [multiple sensor data, computing data@50fps and audio data to process] both core 0 and core 1 and core2 uses same distributed load work.
3. But in core 2 plot there are only 4 peaks of maximum frequency while in core 0 and core1 has 5 peaks of maximum frequency.

**While running all sensors simultaneously and computing data @50fps and we calculated core 3 frequency:**

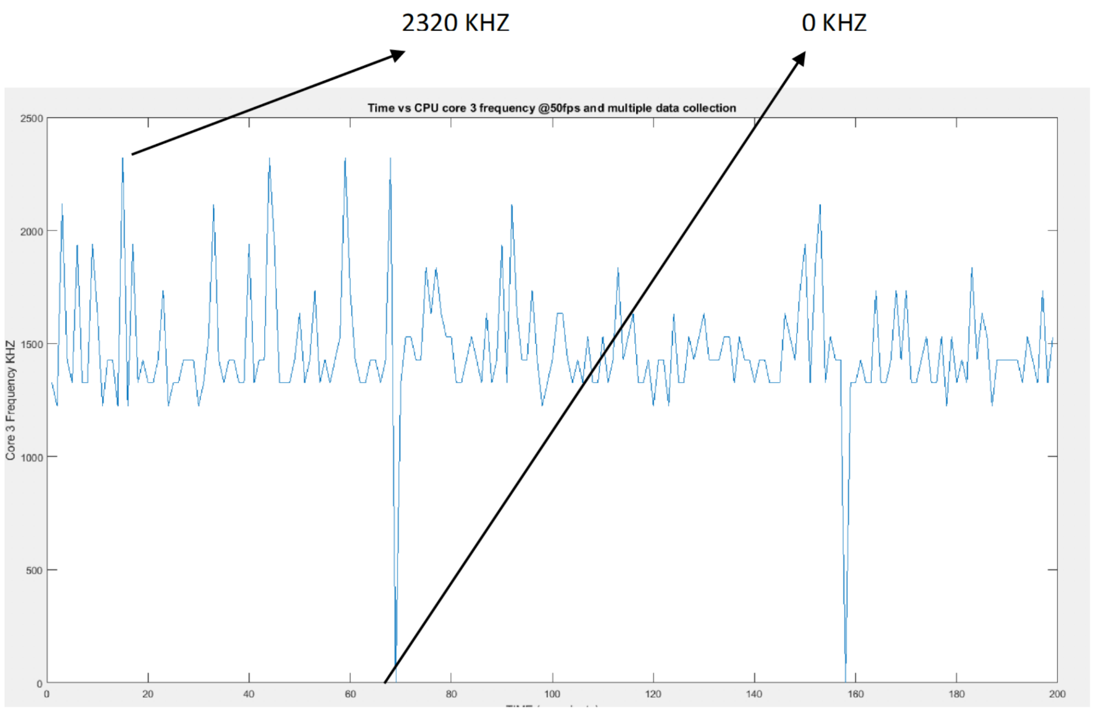


Figure 36: CPU core 3 frequency operation with respect to time@50fps

1. The above plot shows that core 3 distributed frequencies from this plot it concludes that minimum frequency of core 3 goes to 0 KHz and maximum frequency of core 3 goes to 2320 KHz [2320000 Hz].
2. From this graph it concludes that at some instant of time core 3 goes to offline at 69 minutes and 158 minutes.
3. By analyzing core frequency load work according to running all multiple sensor data and computing data @50fps and audio data process, which conclude that core0, core 1, core 2 uses all the time while running background application but core 3 goes offline at two times.

**While running all sensors simultaneously and computing data @50fps and combine all core [0,1,2,3] frequency:**

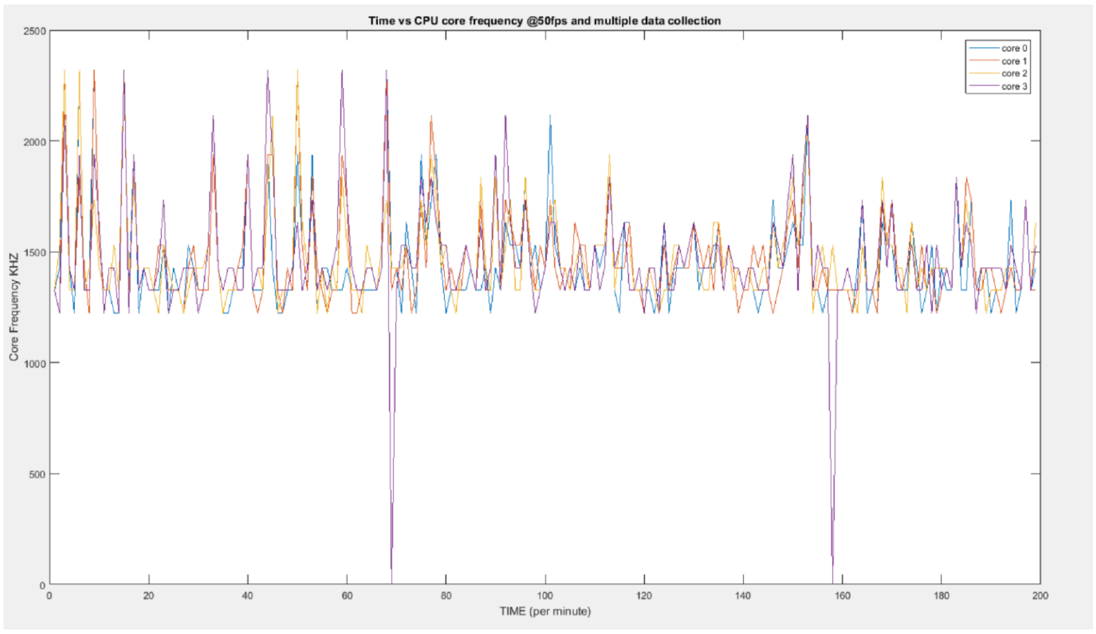
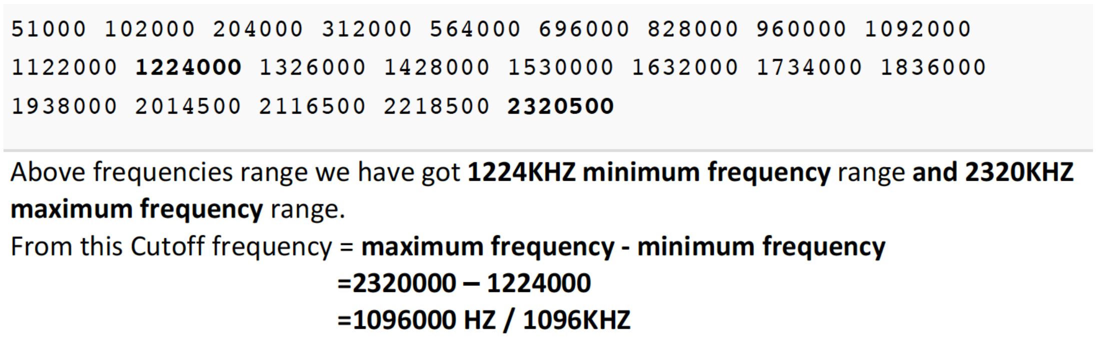


Figure 37: CPU core0,1,2 3 frequency operation with respect to time@50fps

1. The above graph shows frequency distribution of all cores by running all multiple sensors and computing data @50fps.
2. This are all above CPU distributed work load frequency, while running all the process we analyzed that Core3 is not used two times, which makes that all load work distributed by only core0,1,2 and very less use of core3.
3. The range of frequency we analyzed from each cores all are matching with Tegra board data sheet frequencies which has 22 configurable CPU frequencies range from [51MHz to 2.32GHz].



```
51000 102000 204000 312000 564000 696000 828000 960000 1092000
1122000 1224000 1326000 1428000 1530000 1632000 1734000 1836000
1938000 2014500 2116500 2218500 2320500
```

Above frequencies range we have got **1224KHZ minimum frequency** range **and 2320KHZ maximum frequency** range.
From this Cutoff frequency = **maximum frequency - minimum frequency**
**=2320000 – 1224000**
**=1096000 HZ / 1096KHZ**

Figure 38: Tegra board data sheet frequencies which has 22 configurable CPU frequencies @50fps

4. Above frequencies range we have got 1224KHz minimum frequency range and 2320KHz maximum frequency range.
5. From this Cutoff frequency = maximum frequency - minimum frequency =2320000 - 1224000 =1096000 Hz / 1096KHz

**While running all sensors simultaneously and computing data @30fps and combine all core0 frequency**

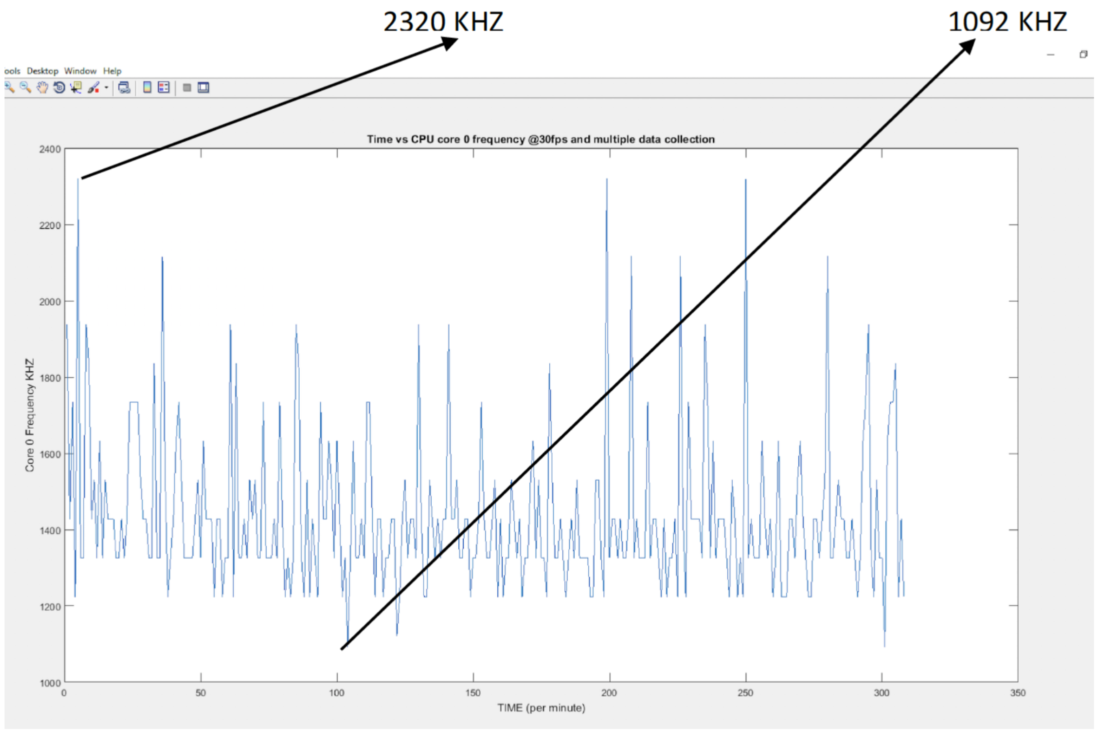


Figure 39: CPU core 0 frequency operation with respect to time @30fps

1. The above plot shows the core 0 distributed frequencies from this plot it concludes that minimum frequency of core 0 goes to 1092 KHZ [1092000 Hz] and maximum frequency of core 0 goes to 2320 KHz [2320000 Hz].

2. In our system core 0 frequency ranges from [1092KHz to 2320KHZ], which is totally depends on running all sensors and computing data @50 fps and audio data process.

**While running all sensors simultaneously and computing data @30fps and calculated core 1 frequency**

1. The above plot shows that core 1 distributed frequencies from this plot it concludes that minimum frequency of core 1 goes to 1092 KHz [1092000 Hz] and maximum frequency of core 1 goes to 2320 KHz [2320000 Hz].

2. The above plot shows same minimum and maximum frequency in both Core 0 and Core 1. It proves that while running our system [multiple sensor data, computing data@30fps and audio data to process] both core 0 and core 1 uses same distributed load work.

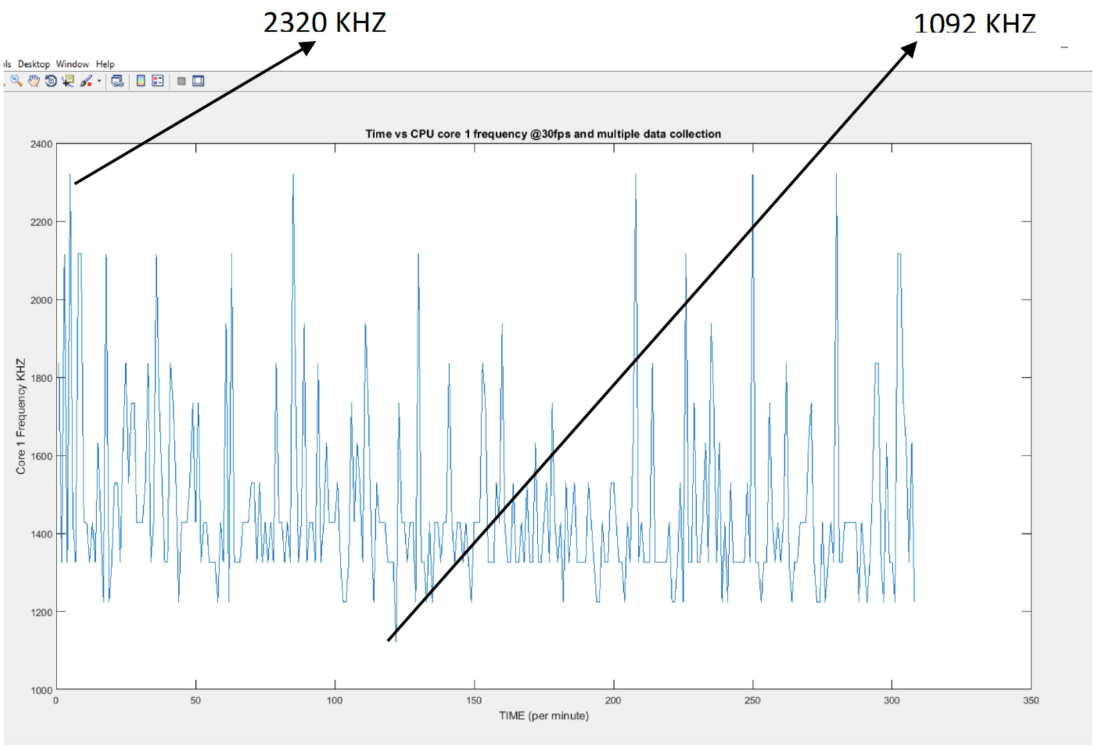


Figure 40: CPU core 1 frequency operation with respect to time @30fps

**While running all sensors simultaneously and computing data @30fps and calculated core2 frequency**

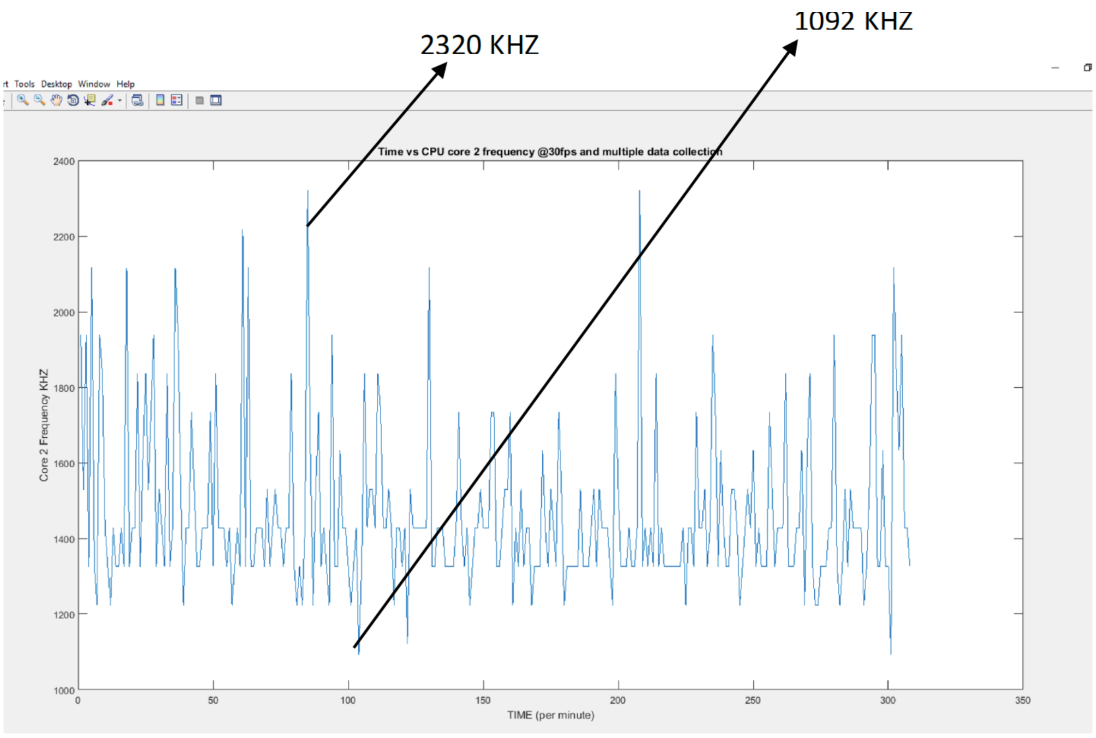


Figure 41: CPU core 2 frequency operation with respect to time @30fps

1. The above plot shows that core 2 distributed frequencies from this plot it concludes that minimum frequency of core 2 goes to 1092 KHz [1092000 Hz] and maximum frequency of core 2 goes to 2320 KHz [2320000 Hz].

2. The above plot shows same minimum and maximum frequency in both Core 0 and Core 1 and core2. It proves that while running our system [multiple sensor data, computing data@30fps and audio data to process] both core 0 and core 1 and core2 uses same distributed load work.

3. But in core 2 plot there are only 2 peaks of maximum frequency while in core1 has 5 peaks of maximum frequency.

**While running all sensors simultaneously and computing data @30fps and calculated core3 frequency**

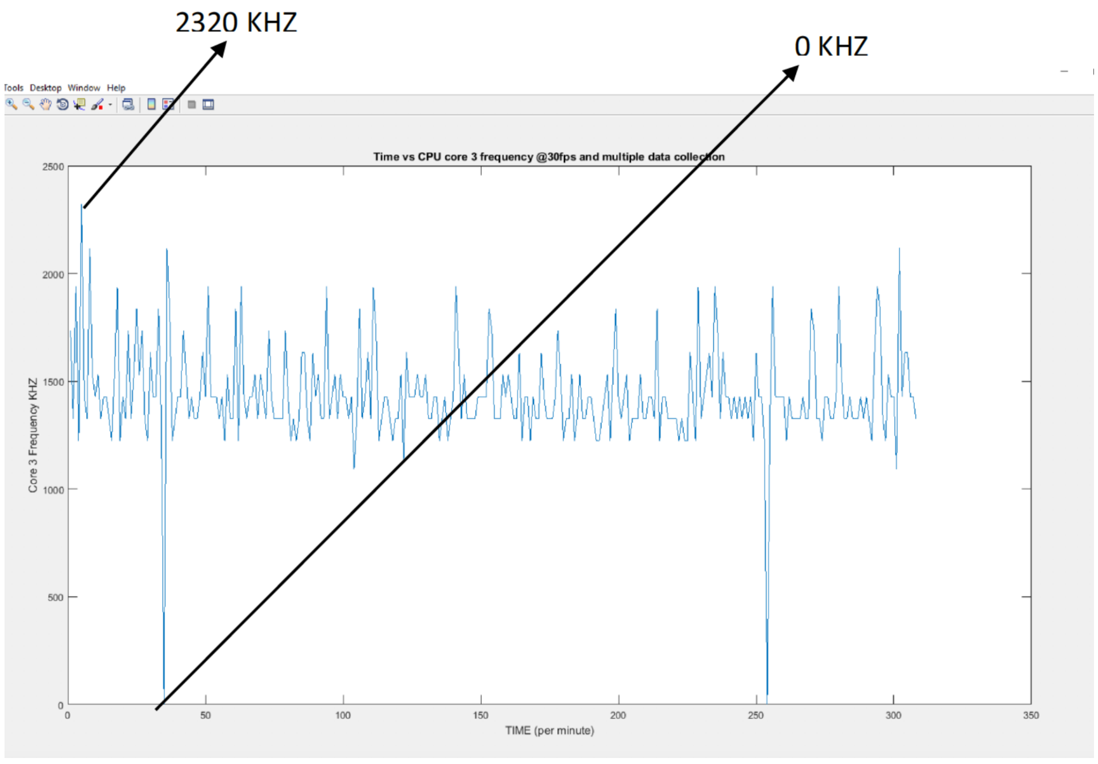


Figure 42: CPU core 3 frequency operation with respect to time @30fps

1. The above plot shows that core 3 distributed frequencies from this plot it concludes that minimum frequency of core 3 goes to 0 KHz and maximum frequency of core 3 goes to 2320 KHz [2320000 Hz].

2. From this graph it concludes that at some instant of time core 3 goes to offline.

3. By analyzing core frequency load work according to running all multiple sensor data and computing data @30fps and audio data process, which conclude that core0, core 1, core 2 uses all the time while running background application but core 3 goes offline at two times.

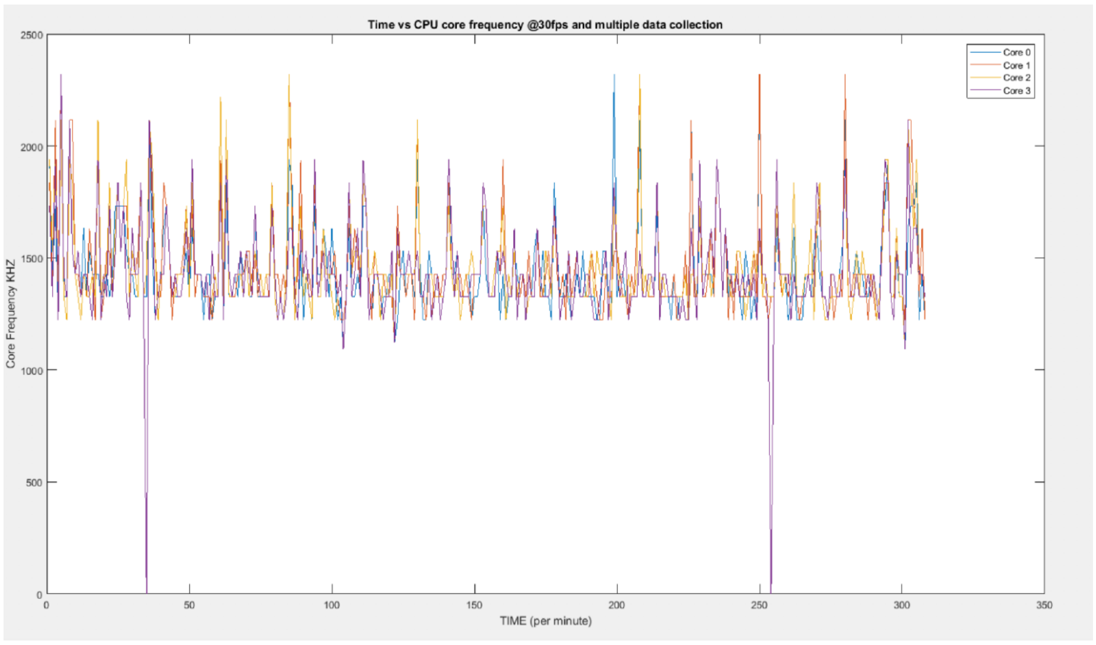
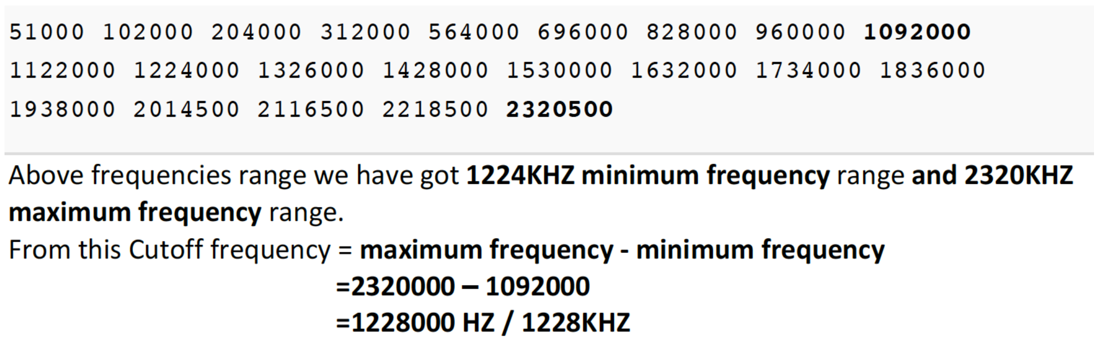


Figure 43: CPU core0,1,2 3 frequency operation with respect to time@30fps

**While running all sensors simultaneously and computing data @30fps and calculated combine all core [0,1,2,3] frequency:**

1. The above graph shows frequency distribution of all cores by running all multiple sensors and computing data @30fps and audio data.
2. This are all above CPU distributed work load frequency, while running all the process we analyzed that Core3 is not used two times, which makes that all load work distributed by only core0,1,2 and very less use of core3.
3. The range of frequency we analyzed from each cores all are matching with Tegra board data sheet frequencies which has 22 configurable CPU frequencies range from [51MHz to 2.32GHz].



```
51000 102000 204000 312000 564000 696000 828000 960000 **1092000**
1122000 1224000 1326000 1428000 1530000 1632000 1734000 1836000
1938000 2014500 2116500 2218500 **2320500**
```

Above frequencies range we have got **1224KHZ minimum frequency** range **and 2320KHZ maximum frequency** range.

From this Cutoff frequency = **maximum frequency - minimum frequency**

**=2320000 – 1092000**

**=1228000 HZ / 1228KHZ**

Figure 44: Tegra board data sheet frequencies which has 22 configurable CPU frequencies @30fps

**While running all sensors simultaneously and computing data @10fps and audio data process we calculated and combine all core0 frequency**

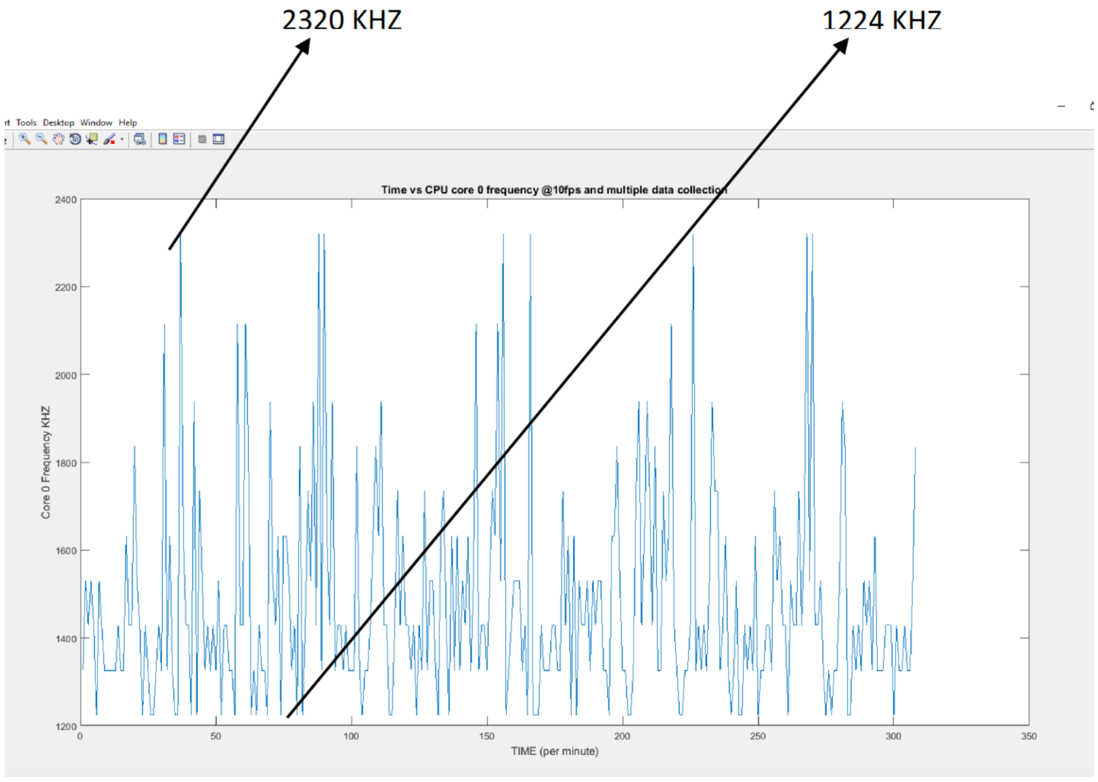


Figure 45: CPU core0 frequency operation with respect to time@10fps

1. The above plot shows the core 0 distributed frequencies from this plot it concludes that minimum frequency of core 0 goes to 1224 KHz [1224000 Hz] and maximum frequency of core 0 goes to 2320 KHZ [2320000 Hz]
2. In our system core 0 frequency ranges from [1224KHz to 2320KHz], which is totally depends on running all sensors and computing data @10 fps and audio data process.

**While running all sensors simultaneously and computing data @10fps and audio data process we calculated and combine all core1 frequency**

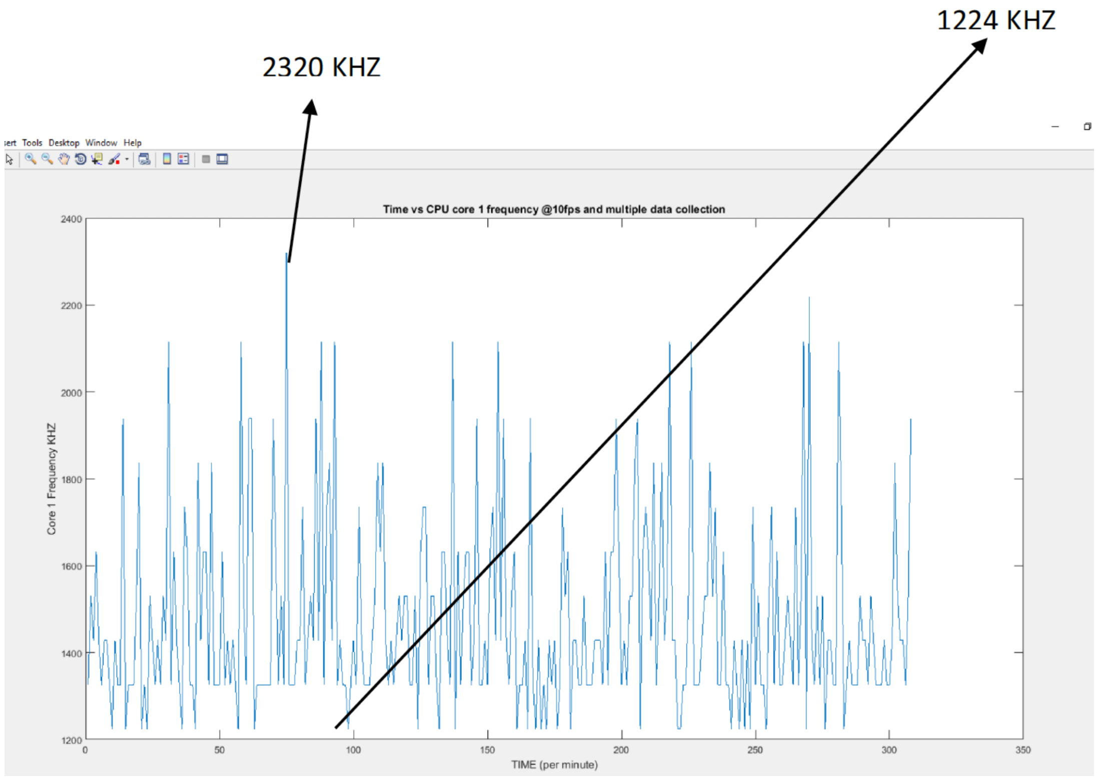


Figure 46: CPU core1 frequency operation with respect to time@10fps

1. The above plot shows that core 1 distributed frequencies from this plot it concludes that minimum frequency of core 1 goes to 1224 KHz [1224000 Hz] and maximum frequency of core 1 goes to 2320 KHz [2320000 Hz].
2. The above plot shows same minimum and maximum frequency in both Core 0 and Core 1. It proves that while running our system [multiple sensor data, computing data@10fps and audio data to process] both core 0 and core 1 uses same distributed load work.

**While running all sensors simultaneously and computing data @10fps and calculated combine all core2 frequency**

1. The above plot shows that core 2 distributed frequencies from this plot it concludes that minimum frequency of core 2 goes to 1224 KHZ [1224000 Hz] and maximum frequency of core 2 goes to 2320 KHz [2320000 Hz].
2. The above plot shows same minimum and maximum frequency in both Core 0 and Core 1 and core2. It proves that while running our system [multiple sensor data, computing data@10fps and audio data to process] both core 0 and core 1 and core2 uses same distributed load work.

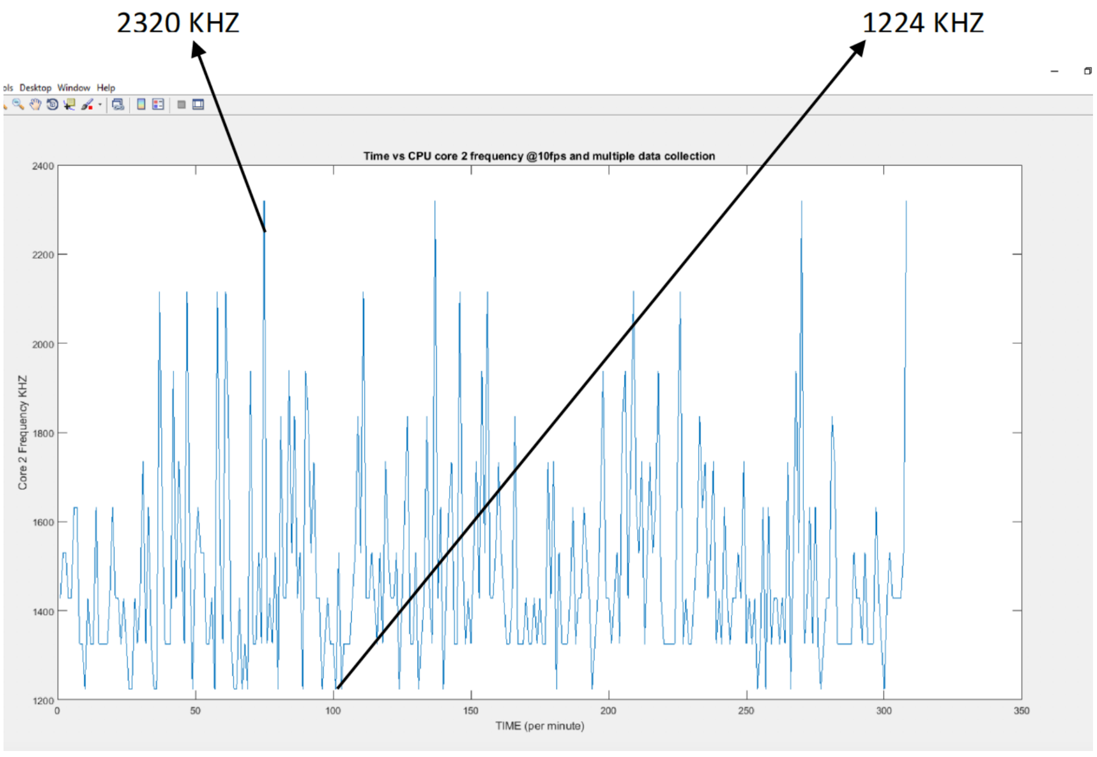


Figure 47: CPU core2 frequency operation with respect to time@10fps

**While running all sensors simultaneously and computing data @10fps and calculated combine all core3 frequency**

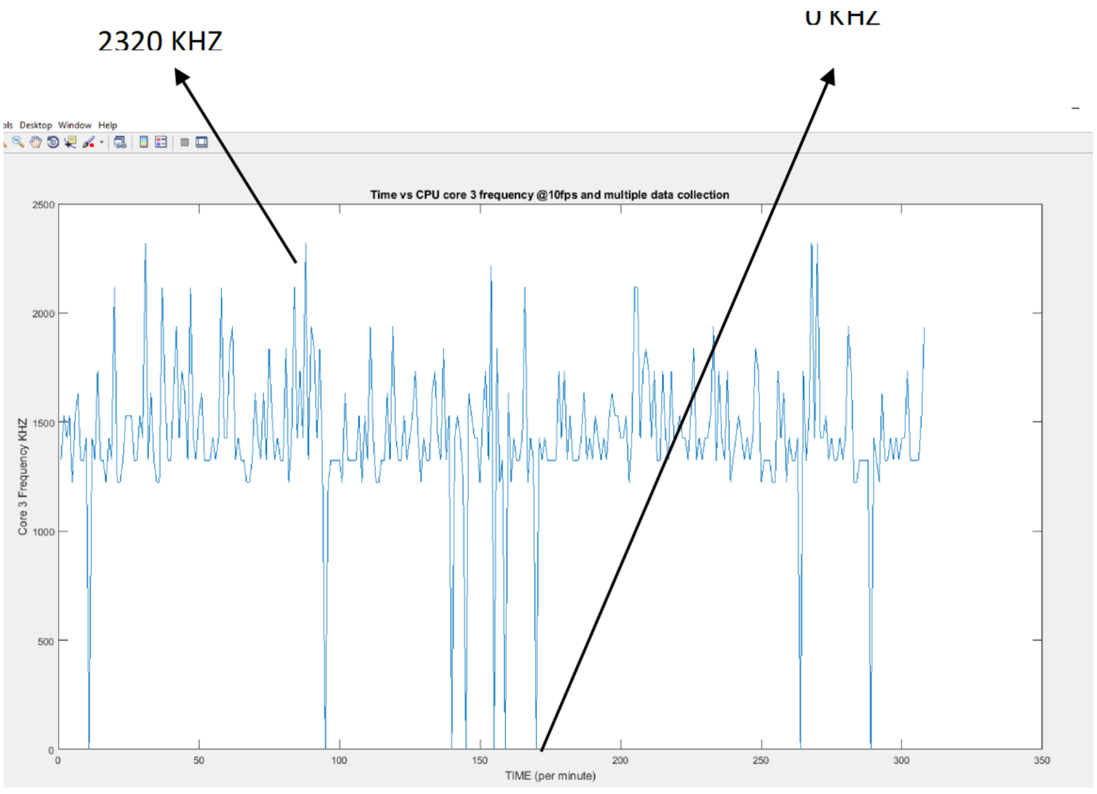


Figure 48: CPU core3 frequency operation with respect to time@10fps

1. The above plot shows that core 3 distributed frequencies from this plot it concludes that minimum frequency of core 3 goes to 0 KHz and maximum frequency of core 3 goes to 2320 KHz [2320000 Hz].
2. From this graph it concludes that at some instant of time core 3 goes to offline.
3. By analyzing core frequency load work according to running all multiple sensor data and computing data @10fps and audio data process, which conclude that core0, core 1, core 2 uses all the time while running background application but core 3 goes offline at nine times.

**While running all sensors simultaneously and computing data @10fps and calculated combine all core [0,1,2,3] frequency:**

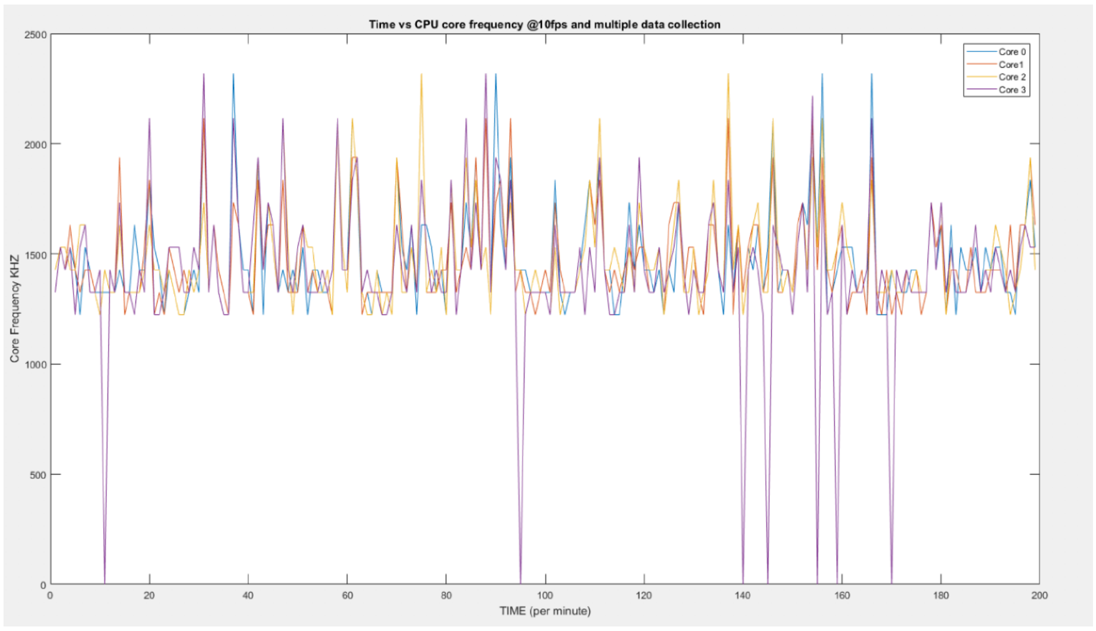
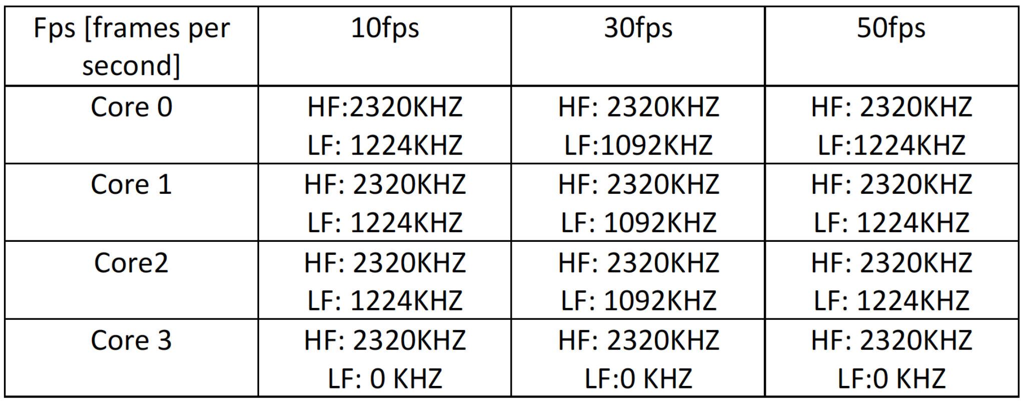


Figure 49: CPU core0,1,2 3 frequency operation with respect to time@10fps

1. According to each core individual frequency characteristics @ different computation data on @50fps,30fps,10fps. It shows below result which shows as Jetson TK1 CPU frequencies distributed in range from 1092KHZ, 1224KHZ and Maximum frequency 2320KHZ in our system.
2. In our system Minimum CPU core [core0,1,2,3] frequencies 1092KHz and Maximum CPU core frequencies 2320KHz. In our system we are not dealing with any GPU core frequency.
3. In this system our CPU core frequencies which shows moderate speed of clock frequency according to 22 configurable frequencies. Which shows good performance of a board. As CPU core clock frequencies increases [settled to moderate], performance of an operation also increases [Writing speed, reading speed and very less power consumption and low heat generation in TK1 board].



| Fps [frames per second] | 10fps | 30fps | 50fps |
|---|---|---|---|
| Core 0 | HF:2320KHZ LF: 1224KHZ | HF: 2320KHZ LF:1092KHZ | HF: 2320KHZ LF:1224KHZ |
| Core 1 | HF: 2320KHZ LF: 1224KHZ | HF: 2320KHZ LF: 1092KHZ | HF: 2320KHZ LF: 1224KHZ |
| Core2 | HF: 2320KHZ LF: 1224KHZ | HF: 2320KHZ LF: 1092KHZ | HF: 2320KHZ LF: 1224KHZ |
| Core 3 | HF: 2320KHZ LF: 0 KHZ | HF: 2320KHZ LF:0 KHZ | HF: 2320KHZ LF:0 KHZ |

Figure 50: High frequency operation and low frequency operation with respect to @10fps@20fps@30fps@40fps

1. If in case to lower power consumption in TK1 board which require to reduce each CPU core clock frequencies as low as possible.
2. Ex: Low CPU clock frequencies [core0,1,2,3] should be 5100HZ.
3. As in Jetson TK1 board has two CPU clusters [High performance and low performance cluster]. In Jetson TK1 board low performance core work on very low power in that case needs to off each high performance core [core0,1,2,3].

## 7.10 Performance of a TK1 board using Benchmark

In this process, we checked performance of a TK1 board by running prime number benchmark and find out load performance on TK1 board by changing internal parameter of a TK1 board such as temperature of a board, memory,PLL, diode temperature.

### 7.10.1 Computing all prime no. less than 100K

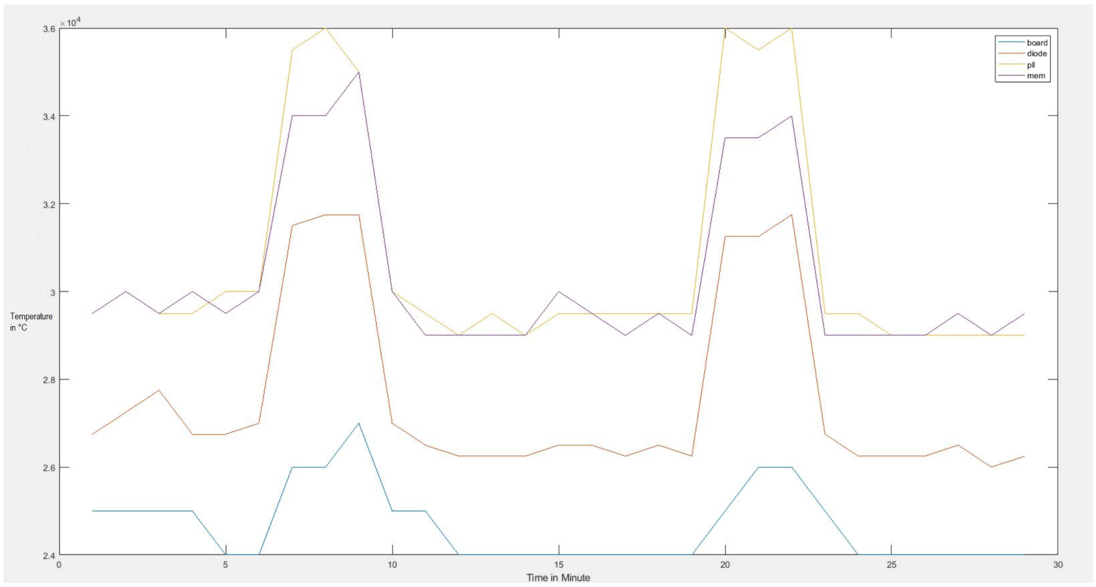


Figure 51: Time vs Temperature for calculating 100k benchmark

The above plot represents time for running benchmark for calculating number of prime number in 100k vs performance load in temperature with respect to board temperature, memory, PLL temperature. Plot shows first 30 minutes sleeping time before running benchmark when peak started benchmark starts to run which makes load on a board by variation in temperature. Once benchmark process done temperature slowly decreases and again maintained sleepier time for 30 minutes.

**Sensitivity Measurements**

Sensitivity calculation is a rate of change of temperature with respect to rate of change of time.

$$Sensitivity = \frac{\Delta ChangeinTemp}{\Delta ChangeinTime}$$

Here we calculated sensitivity while case 1 when benchmark started load increases so here first we measured increased computational load and calculated sensitivity. We measured sensitivity for different thermal zones inside Jetson TK1.

$$1.SensitivityofDiode[increase] = \frac{31.3^{\circ}\mathrm{C} - 27^{\circ}\mathrm{C}}{7min - 6min}$$

which is,

$$Sensitivity of Diode[increase] = 4.3\frac{℃}{min}$$

$$2.Sensitivity of Diode[decrease] = \frac{31.5℃ - 27℃}{9min - 11min}$$

which is,

$$Sensitivity of Diode[decrease] = -2.25\frac{℃}{min}$$

$$2.Sensitivity of Board[increase] = \frac{26℃ - 24℃}{7min - 6min}$$

which is,

$$Sensitivity of Board[increase] = 2\frac{℃}{min}$$

$$3.Sensitivity of Board[decrease] = \frac{27℃ - 24℃}{9min - 13min}$$

which is,

$$Sensitivity of Board[decrease] = -0.75\frac{℃}{min}$$

$$3.Sensitivity of PLL[increase] = \frac{34℃ - 31℃}{7min - 6min}$$

which is,

$$Sensitivity of PLL[increase] = 3\frac{℃}{min}$$

$$4.Sensitivity of PLL[decrease] = \frac{30℃ - 35℃}{11min - 8min}$$

which is,

$$Sensitivity of PLL[decrease] = -1.66 \frac{℃}{min}$$

$$5. Sensitivity of MEM[increase] = \frac{35℃ - 30℃}{7min - 6min}$$

which is,

$$Sensitivity of MEM[increase] = 5 \frac{℃}{min}$$

$$6. Sensitivity of MEM[decrease] = \frac{33℃ - 36℃}{11min - 8min}$$

which is,

$$Sensitivity of MEM[decrease] = -1 \frac{℃}{min}$$

### 7.10.2 Computing all prime no.less than 300K

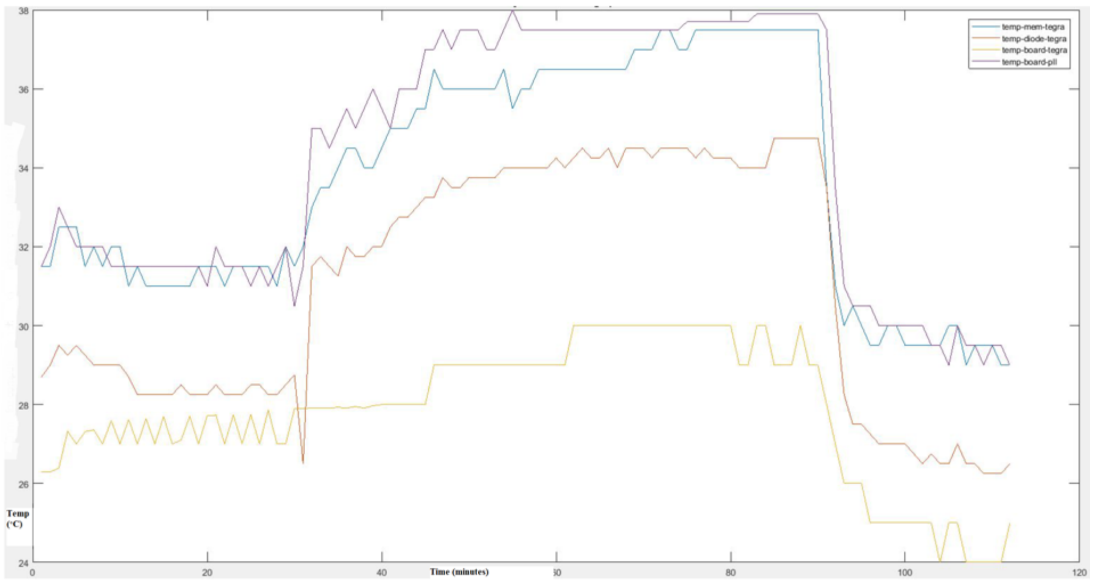


Figure 52: Time vs Temperature for calculating 300k benchmark

Calculating number of prime number in 300k vs performance load in temperature with respect to board temperature, memory, PLL temperature. Plot shows first 30 minutes sleeping time before running benchmark when peak started benchmark starts to run which makes load on a board by variation in temperature. Once benchmark process done temperature slowly decreases and again maintained sleepier time for 30 minutes.

$$Sensitivity = \frac{\Delta ChangeinTemp}{\Delta ChangeinTime}$$

Here we calculated sensitivity while case 1 when benchmark started load increases so here first we measured increased computational load and calculated sensitivity. We measured sensitivity for different thermal zones inside Jetson TK1.

$$1.Sensitivityof Diode[increase] = \frac{34℃ - 28℃}{84min - 30min}$$

which is,

$$Sensitivityof Diode[increase] = 0.11\frac{℃}{min}$$

$$2.Sensitivityof Diode[decrease] = \frac{27℃ - 34}{110min - 90min}$$

which is,

$$Sensitivityof Diode[decrease] = -0.35\frac{℃}{min}$$

$$2.Sensitivity of Board[increase] = \frac{28℃ - 26.5℃}{45min - 1min}$$

which is,

$$Sensitivity of Board[increase] = -0.034\frac{℃}{min}$$

$$3.Sensitivity of Board[decrease] = \frac{24℃ - 29℃}{110min - 90min}$$

which is,

$$Sensitivity of Board[decrease] = -0.25\frac{℃}{min}$$

$$3.Sensitivity of PLL[increase] = \frac{34℃ - 28℃}{90min - 30min}$$

which is,

$$Sensitivity of PLL[increase] = 0.1\frac{℃}{min}$$

$$4.Sensitivity of PLL[decrease] = \frac{29℃ - 38℃}{110min - 82min}$$

which is,

$$Sensitivity of PLL[decrease] = -0.321\frac{℃}{min}$$

$$5.Sensitivity of MEM[increase] = \frac{37℃ - 32℃}{70min - 30min}$$

which is,

$$Sensitivity of MEM[increase] = 0.125 \frac{℃}{min}$$

$$6. Sensitivity of MEM[decrease] = \frac{29℃ - 37.5℃}{110min - 70min}$$

which is,

$$Sensitivity of MEM[decrease] = -0.2125 \frac{℃}{min}$$

### 7.10.3 Computing all prime no. less than 500K

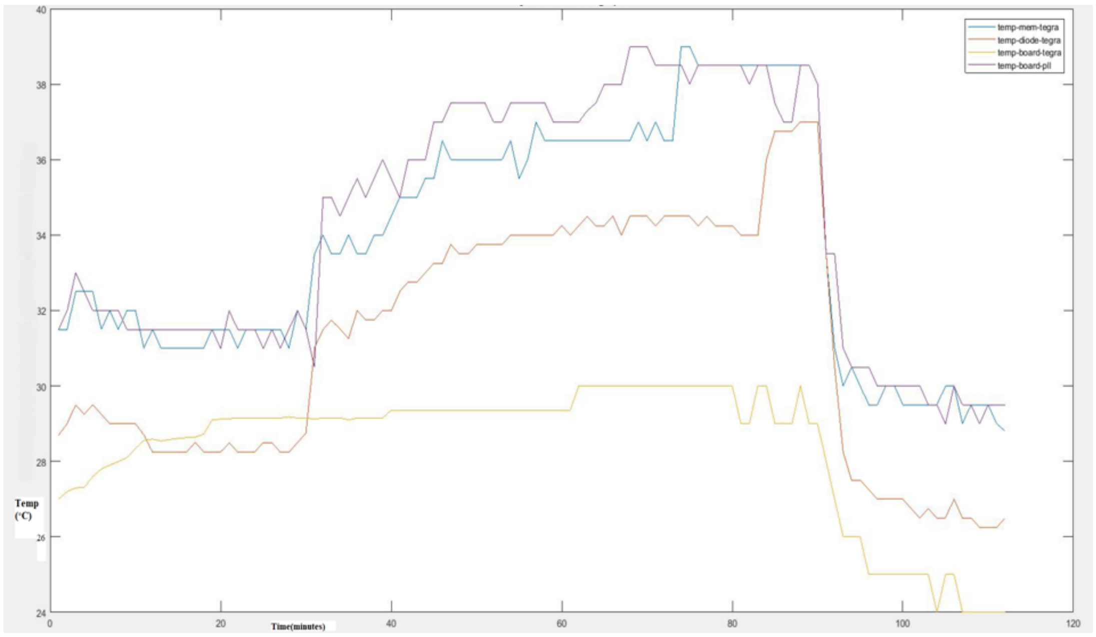


Figure 53: Time vs Temperature for calculating 500k benchmark

Calculating number of prime number in 500k vs performance load in temperature with respect to board temperature, memory, Phase Lock Loop (PLL) temperature. Plot shows first 30 minutes sleeping time before running benchmark when peak started benchmark starts to run which makes load on a board by variation in temperature. Once benchmark process done temperature slowly decreases and again maintained sleepier time for 30 minutes.

$$Sensitivity = \frac{\Delta ChangeinTemp}{\Delta ChangeinTime}$$

Here we calculated sensitivity while case 1 when benchmark started load increases so here first we measured increased computational load and calculated sensitivity. We measured sensitivity for different thermal zones inside Jetson TK1.

$$1.Sensitivityof Diode[increase] = \frac{36℃ - 30℃}{82min - 29min}$$

which is,

$$Sensitivityof Diode[increase] = 0.113\frac{℃}{min}$$

$$2.Sensitivityof Diode[decrease] = \frac{37℃ - 26℃}{90min - 110min}$$

which is,

$$Sensitivity of Diode[decrease] = -0.55\frac{℃}{min}$$

$$2.Sensitivity of Board[increase] = \frac{29℃ - 27℃}{61min - 1min}$$

which is,

$$Sensitivity of Board[increase] = -.033\frac{℃}{min}$$

$$3.Sensitivity of Board[decrease] = \frac{29℃ - 24℃}{90min - 110min}$$

which is,

$$Sensitivity of Board = -0.25\frac{℃}{min}$$

$$3.Sensitivity of PLL[increase] = \frac{39℃ - 31℃}{70min - 30min}$$

which is,

$$Sensitivity of PLL[increase] = 0.2\frac{℃}{min}$$

$$4.Sensitivity of PLL[decrease] = \frac{29℃ - 38℃}{110min - 82min}$$

which is,

$$Sensitivity of PLL[decrease] = -0.321\frac{℃}{min}$$

$$5. Sensitivity of MEM[increase] = \frac{39℃ - 31℃}{70min - 30min}$$

which is,

$$Sensitivity of MEM[increase] = 0.2 \frac{℃}{min}$$

$$6. Sensitivity of MEM[decrease] = \frac{29℃ - 39℃}{112min - 90min}$$

which is,

$$Sensitivity of MEM[decrease] = -0.455 \frac{℃}{min}$$

### 7.10.4 Computing VDE-video decoder engine benchmark

We have checked computational parameter load/ related to heat on TK1 board using change in CPU temp, GPU temp,RAM used,VDE,ADE.

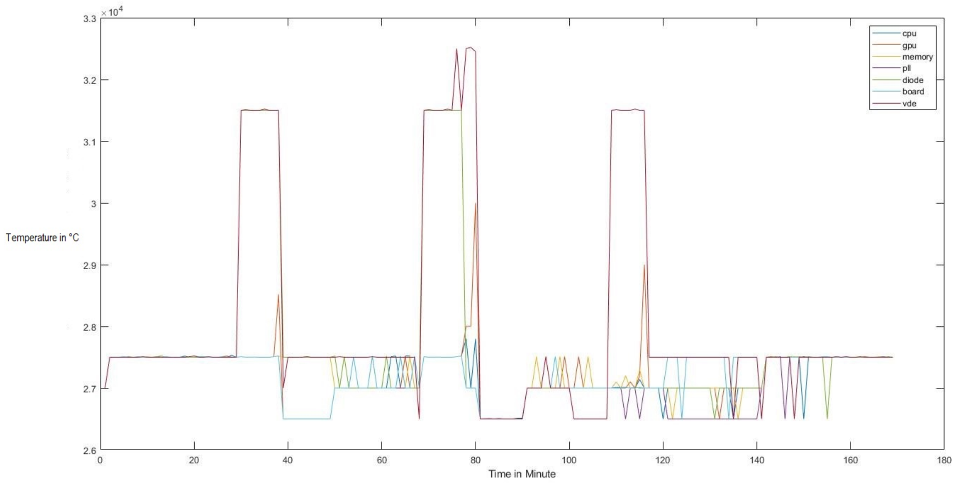


Figure 54: Time vs Video Benchmark

$$Sensitivity = \frac{\Delta ChangeinTemp}{\Delta ChangeinTime}$$

Here we calculated sensitivity while case 1 when benchmark started load increases so here first we measured increased computational load and calculated sensitivity. We measured sensitivity for different thermal zones inside Jetson TK1. **Board temperature doesn't effect much.**

$$1stcase.SensitivityofCPU, GPU, VDE, MEM, PLL, DIODE[increase] = \frac{31.5℃ - 26℃}{34min - 30min}$$

which is,

$$SensitivityofCPU, GPU, VDE, MEM, PLL, DIODE[increase] = 1.37\frac{℃}{min}$$

$$1.1SensitivityofCPU, GPU, VDE, MEM, PLL, DIODE[decrease] = \frac{32.5℃ - 26℃}{39min - 41min}$$

which is,

$$Sensitivity of CPU, GPU, VDE, MEM, PLL, DIODE[decrease] = -3.25 \frac{℃}{min}$$

$$2nd case. Sensitivity of CPU, GPU, VDE, MEM, PLL, DIODE[increase] = \frac{31.5℃ - 27℃}{104min - 100min}$$

which is,

$$Sensitivity of CPU, GPU, VDE, MEM, PLL, DIODE[increase] = 1.125 \frac{℃}{min}$$

$$2.1 Sensitivity of CPU, GPU, VDE, MEM, PLL, DIODE[decrease] = \frac{26℃ - 32.5℃}{78min - 68min}$$

which is,

$$Sensitivity of CPU, GPU, VDE, MEM, PLL, DIODE[decrease] = -0.65 \frac{℃}{min}$$

$$3rd case Sensitivity of CPU, GPU, VDE, MEM, PLL, DIODE[increase] = \frac{31.5℃ - 27℃}{104min - 100min}$$

which is,

$$Sensitivity of CPU, GPU, VDE, MEM, PLL, DIODE[increase] = 1.125 \frac{℃}{min}$$

$$3.1 Sensitivity of CPU, GPU, VDE, MEM, PLL, DIODE[decrease] = \frac{26℃ - 32.5℃}{110min - 104min}$$

which is,

$$Sensitivity of CPU, GPU, VDE, MEM, PLL, DIODE[decrease] = -1.08\frac{℃}{min}$$

**Mean in all three case when temperatures increases**= 1.37 +1.125+1.125/3

**Mean in all three case when temperatures increases**= 1.20

**Mean in all three case when temperatures decreases**= -3.25-0.65-1.08/3

**Mean in all three case when temperatures decreases**= -1.9

1. 30min Sleeping GPU MORE
2. 100x100 pixel@30fps 8min RUN
3. 30min Sleeping
4. 1000x1000 pixel@30fps 12min RUN
5. 30min Sleeping
6. 100x100 pixel@30fps. 8min RUN
7. 30min Sleeping
8. PLot has: CPU,GPU,MEM,PLL,DIODE,BOARD, VDE
9. video decoder engine has more temperature.

### 7.10.5 Computing ADE-audio decoder engine benchmark

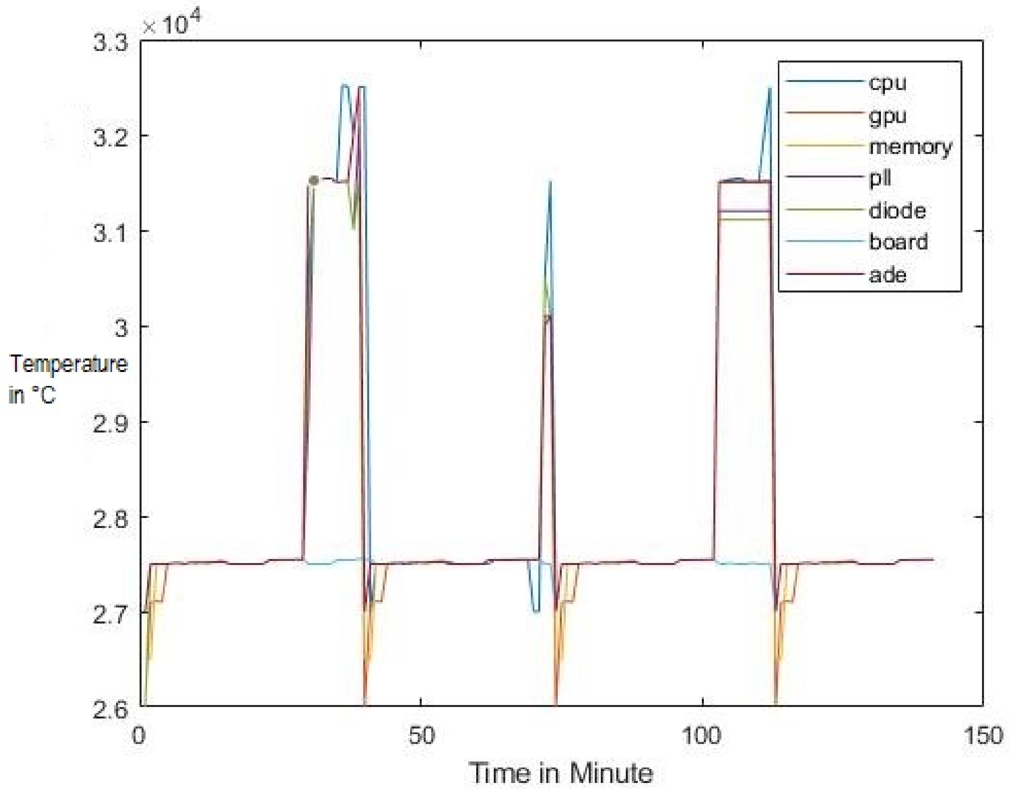


Figure 55: Time vs Audio benchmark

1. 30min Sleeping
2. 44.1KHZ audio= 10min run
3. 30min Sleeping=70
4. 1KHZ audio =2min run
5. 30min Sleeping
6. 22KHZ audio= 8min run
7. 30min Sleeping
8. PLot has: CPU,GPU,MEM,PLL,DIODE,BOARD,ADE
9. audio decoder engine has more temperature.

$$Sensitivity = \frac{\Delta ChangeinTemp}{\Delta ChangeinTime}$$

Here we calculated sensitivity while case 1 when benchmark started load increases so here first we measured increased computational load and calculated sensitivity. We measured sensitivity for different thermal zones inside Jetson TK1.**Board temperature doesn't effect much.**

$$1st case Sensitivity of CPU, GPU, ADE, MEM, PLL, DIODE[increase] = \frac{31.5\text{℃} - 26\text{℃}}{34 min - 30 min}$$

which is,

$$Sensitivity of CPU, GPU, ADE, MEM, PLL, DIODE[increase] = 1.37 \frac{\text{℃}}{min}$$

$$1.1 Sensitivity of CPU, GPU, ADE, MEM, PLL, DIODE[decrease] = \frac{32.5\text{℃} - 26\text{℃}}{39 min - 41 min}$$

which is,

$$Sensitivity of CPU, GPU, ADE, MEM, PLL, DIODE[decrease] = -3.25 \frac{\text{℃}}{min}$$

$$2nd case Sensitivity of CPU, GPU, ADE, MEM, PLL, DIODE[increase] = \frac{31.5\text{℃} - 27\text{℃}}{104 min - 100 min}$$

which is,

$$Sensitivity of CPU, GPU, ADE, MEM, PLL, DIODE[increase] = 1.125 \frac{\text{℃}}{min}$$

$$2.1 Sensitivity of CPU, GPU, ADE, MEM, PLL, DIODE[decrease] = \frac{26\text{℃} - 32.5\text{℃}}{78 min - 68 min}$$

which is,

$$Sensitivity of CPU, GPU, ADE, MEM, PLL, DIODE[decrease] = -0.65 \frac{℃}{min}$$

$$3rd case Sensitivity of CPU, GPU, ADE, MEM, PLL, DIODE[increase] = \frac{31.5℃ - 27℃}{104min - 100min}$$

which is,

$$Sensitivity of CPU, GPU, ADE, MEM, PLL, DIODE[increase] = 1.125 \frac{℃}{min}$$

$$3.1 Sensitivity of CPU, GPU, ADE, MEM, PLL, DIODE[decrease] = \frac{26℃ - 32.5℃}{110min - 104min}$$

which is,

$$Sensitivity of CPU, GPU, ADE, MEM, PLL, DIODE[decrease] = -1.08 \frac{℃}{min}$$

**Mean in all three case when temperatures increases**= 1.37 +1.125+1.125/3

**Mean in all three case when temperatures increases**= 1.20

**Mean in all three case when temperatures decreases**= -3.25-0.65-1.08/3

**Mean in all three case when temperatures decreases**= -1.9

### 7.10.6 Computing individual core benchmark

1. 30min Sleeping -No benchmark utilized
2. BM for 2 core TEMP MORE= 20MIN RUN: LOAD MORE
3. 30min Sleeping
4. BM for 4core DIVIDE LOAD= 10MIN RUN
5. 30min Sleeping
6. BM for 2core =20MIN
7. 30min Sleeping
8. Plot has: CPU,GPU,MEM,PLL,DIODE,BOARD.
9. CPU core has more temperature.

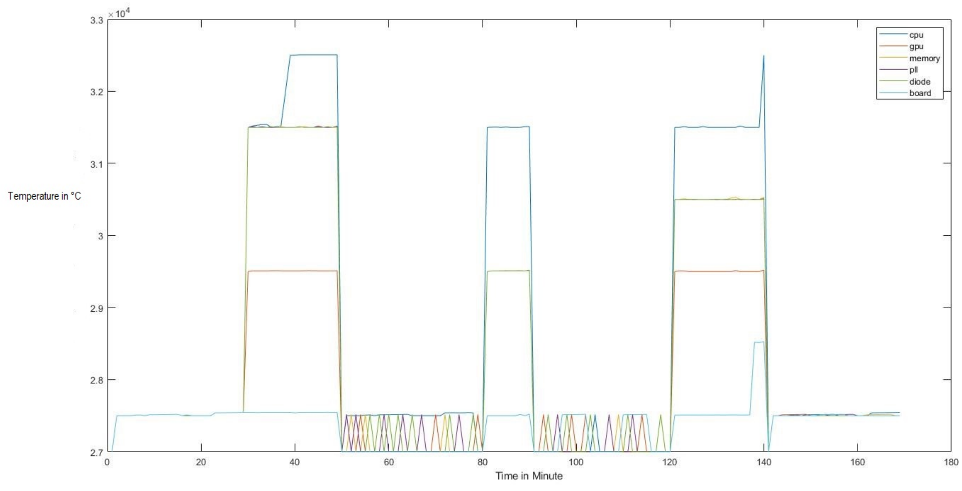


Figure 56: Time vs Individual core benchmark

$$Sensitivity = \frac{\Delta ChangeinTemp}{\Delta ChangeinTime}$$

Here we calculated sensitivity while case 1 when benchmark started load increases so here first we measured increased computational load and calculated sensitivity. We measured sensitivity for different thermal zones inside Jetson TK1.**Board temperature doesn't effect much.**

$$1stcaseSensitivityofCPU[increases] = \frac{31.5°\text{C} - 27.5°\text{C}}{27min - 25min}$$

which is,

$$Sensitivity of CPU[increases] = 2\frac{℃}{min}$$

$$1.1 Sensitivity of GPU[increases] = \frac{29.5℃ - 27.5℃}{31min - 25min}$$

which is,

$$Sensitivity of GPU[increases] = 0.33\frac{℃}{min}$$

$$2nd case Sensitivity of CPU[increase] = \frac{31.5℃ - 27℃}{81min - 79.5min}$$

which is,

$$Sensitivity of CPU[increase] = 1.5\frac{℃}{min}$$

$$2.1 Sensitivity of GPU[increases] = \frac{32℃ - 30℃}{84min - 81min}$$

which is,

$$Sensitivity of GPU[increases] = 0.66\frac{℃}{min}$$

$$3rd case Sensitivity of CPU[increase] = \frac{33℃ - 27℃}{122min - 120min}$$

which is,

$$Sensitivity of CPU[increase] = 2\frac{℃}{min}$$

$$3.1 Sensitivity of GPU[increases] = \frac{30℃ - 26℃}{120min - 118min}$$

which is,

$$Sensitivity of GPU[increases] = 0.5 \frac{℃}{min}$$

$$1st case Sensitivity of CPU[decreases] = \frac{27℃ - 32℃}{50min - 30min}$$

which is,

$$Sensitivity of CPU[decreases] = -0.25 \frac{℃}{min}$$

$$1.1 Sensitivity of GPU[decreases] = \frac{27℃ - 29℃}{50min - 40min}$$

which is,

$$Sensitivity of GPU[decreases] = -0.2 \frac{℃}{min}$$

$$2nd case Sensitivity of CPU[decrease] = \frac{27℃ - 32℃}{90min - 88min}$$

which is,

$$Sensitivity of CPU[decrease] = -2.5 \frac{℃}{min}$$

$$2.1 Sensitivity of GPU[decreases] = \frac{27℃ - 29℃}{90min - 88min}$$

which is,

$$SensitivityofGPU[decreases] = -1\frac{℃}{min}$$

$$3rdcaseSensitivityofCPU[decrease] = \frac{27℃ - 33℃}{140min - 138min}$$

which is,

$$SensitivityofCPU[decrease] = -3\frac{℃}{min}$$

$$3.1SensitivityofGPU[decreases] = \frac{27℃ - 29℃}{140min - 138min}$$

which is,

$$SensitivityofGPU[decreases] = -2\frac{℃}{min}$$

### 7.10.7 Computing DLBS-deep learning benchmark

1. Layers need to say 30,2,15 layers in Alexnet
   30min Sleeping MORE TEMP GPU AND TEMP CPU
2. Alexnet: 30Max layers =21min RUN
3. 30min Sleeping
4. Alexnet: 2 Min layers =5minRUN
5. 30min Sleeping
6. Alexnet: 15 Min layers =15minRUN
7. 30min Sleeping
8. PLot has: CPU,GPU,MEM,PLL,DIODE,BOARD.
9. GPU and CPU has more temperature

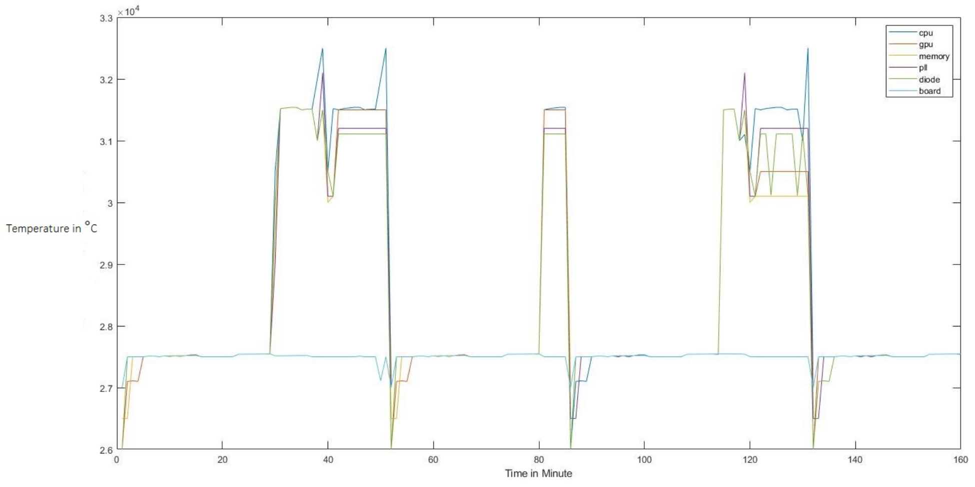


Figure 57: Time vs deep learning benchmark

$$Sensitivity = \frac{\Delta ChangeinTemp}{\Delta ChangeinTime}$$

Here we calculated sensitivity while case 1 when benchmark started load increases so here first we measured increased computational load and calculated sensitivity. We measured sensitivity for different thermal zones inside Jetson TK1.

$$1.SensitivityofCPU, GPU[increase] = \frac{31.5℃ - 26℃}{32min - 30min}$$

which is,

$$SensitivityofCPU, GPU[increase] = 2.75\frac{℃}{min}$$

$$1.1 SensitivityofCPU, GPU[decreases] = \frac{26℃ - 31.5℃}{59min - 55min}$$

which is,

$$SensitivityofCPU, GPU[decreases] = -1.375\frac{℃}{min}$$

$$2. SensitivityofCPU, GPU[increase] = \frac{31.5℃ - 26℃}{82min - 80min}$$

which is,

$$SensitivityofCPU, GPU[increase] = 2.75\frac{℃}{min}$$

$$2.1 SensitivityofCPU, GPU[decreases] = \frac{26℃ - 31.5℃}{82min - 81min}$$

which is,

$$SensitivityofCPU, GPU[decreases] = -5.5\frac{℃}{min}$$

$$3. SensitivityofCPU, GPU[increase] = \frac{27℃ - 32℃}{110min - 128min}$$

which is,

$$SensitivityofCPU, GPU[increase] = 2.75\frac{℃}{min}$$

$$3.1 Sensitivity of CPU, GPU[decreases] = \frac{26℃ - 33℃}{132min - 130min}$$

which is,

$$Sensitivity of CPU, GPU[decreases] = -3.5\frac{℃}{min}$$

**Mean in all three case when temperatures increases**= 2.75 +2.755+2.75/3

**Mean in all three case when temperatures increases**= 2.7

**Mean in all three case when temperatures decreases**= -1.37-5.5-3.5/3

**Mean in all three case when temperatures decreases**= -3.45

# 8 Speed and Accuracy

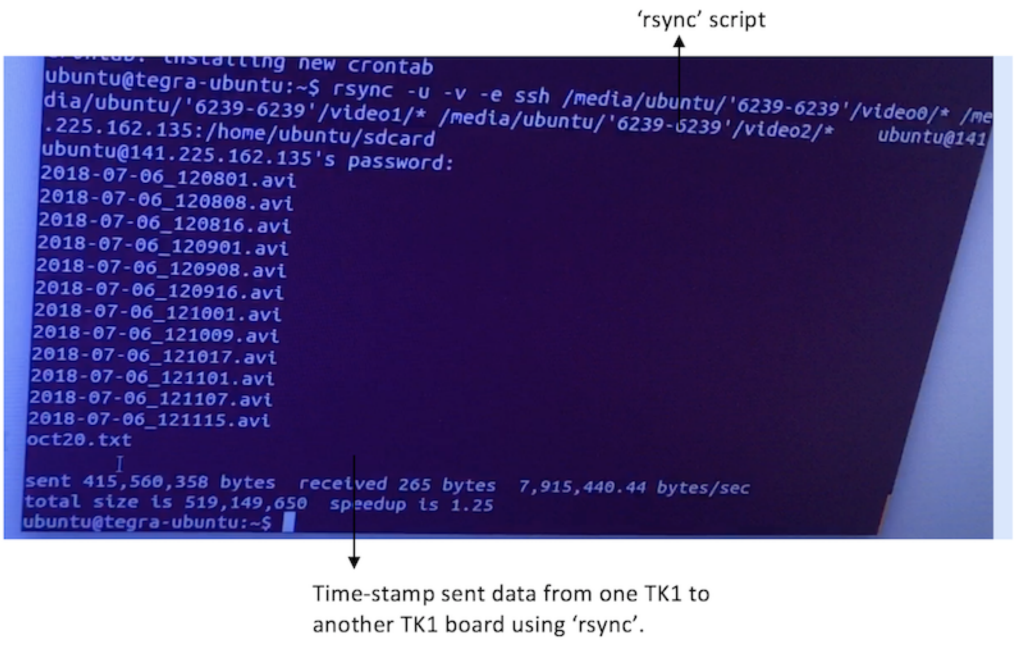
## 8.1 Transmitting/Receiving speed between two TK1

1. **rsync** is command use for transferring and synchronizing data between two Jetson TK1 board by checking timestamp and size of a file. **rsync** process operates by communicating with another rsync process, between a sender and a receiver.

2. While sending data from one TK1 transmitter board to another TK1 receiver board, need to require Internet Protocol (IP) address of a receiver board.

3. rsync-u-v-essh/media/ubuntu/'6239-6239'/video0/*/media/ubuntu/'6239-6239'/video1* /media/ubuntu/'6239-6239'/video2/*ubuntu@address of TK1 receiver:/home/ubuntu/sdcard

4. Above script represent to send data from one TK1 transmitter to another TK1 receiver with IP address and path location of transmitter files of a sender TK1.


'rsync' script
Time-stamp sent data from one TK1 to another TK1 board using 'rsync'.

```
crontab: installing new crontab
ubuntu@tegra-ubuntu:~$ rsync -u -v -e ssh /media/ubuntu/'6239-6239'/video0/* /me
dia/ubuntu/'6239-6239'/video1/* /media/ubuntu/'6239-6239'/video2/*    ubuntu@141
.225.162.135:/home/ubuntu/sdcard
ubuntu@141.225.162.135's password:
2018-07-06_120801.avi
2018-07-06_120808.avi
2018-07-06_120816.avi
2018-07-06_120901.avi
2018-07-06_120908.avi
2018-07-06_120916.avi
2018-07-06_121001.avi
2018-07-06_121009.avi
2018-07-06_121017.avi
2018-07-06_121101.avi
2018-07-06_121107.avi
2018-07-06_121115.avi
oct20.txt

sent 415,560,358 bytes  received 265 bytes  7,915,440.44 bytes/sec
total size is 519,149,650  speedup is 1.25
ubuntu@tegra-ubuntu:~$
```

Figure 58: Time stamping sent data from one TK1 to another TK1 board.

# 9 Challenges and Requirements

In this work results and analysis showed successfully that system can sense all environmental sensor data and physiological data but there were some challenges faced during analysis which are listed below:

1. Running all sensors [BMP280 sensor, Multichannel gas sensor, Microphone, Multiple lens camera]simultaneously to capture data inside Jetson TK1 board.
2. Unsuitable packages and drivers [wifi, sensors driver installation] for Jetson TK1 board to make all things work for sense inside TK1 board.

## 9.1 Future Research of a system

Future research work of a system can be on algorithm part, which states that by running different cases of an algorithm on Jetson TK1 board. In this our experiment result shows that, Irrespective of a different frames per second of camera and simultaneously running multiple sensors, we evaluated different parameter such as CPU temperature of a TK1 board, Reading-Writing speed of a TK1 board, individual core's frequency characteristics and computational load on a TK1 board by running benchmark.

# 10 Conclusion

## 10.1 Key Findings

Wearable devices are most common in modern world and health related functions like, step tracking, Heart rate detection, fall detection are being imbibed into wearables to enable consumer aware of some important aspects of health.

In this thesis we can contribute using Garmin Smart-watch instead of wearable sensor on clothes of mining industry people. Smartwatch has all inbuilt biological sensors and GPS navigation system to track mining industries people. Here all gadgets [smartwatch, Mobile] are connected to TK1 device via Bluetooth or Wi-Fi. As our system has Video cameras which gives live streaming information to another person. As TK1 board is powerful board we can use this system for multiple real-time application by running algorithms. [ex: Health issue, Avoiding Critical emergency situations etc.]

## 10.2 Future Implementation

In the future, we want to experiment with different kinds of processors and implement the same system using newer state-of-the-art processor and sensors.

## References

[1] Stokke, Kristoffer & Stensland, Håkon & Griwodz, Carsten & Halvorsen, Pål. (2015). "Energy Efficient Video Encoding Using the Tegra K1 Mobile Processor". 10.1145/2713168.2713186.

[2] A. Kaur and A. Jasuja, "Health monitoring based on IoT using Raspberry PI," 2017 International Conference on Computing, Communication and Automation (ICCCA), Greater Noida, 2017, pp. 1335-1340. doi: 10.1109/CCAA.2017.8230004.

[3] https://elinux.org/Jetson/I2C.

[4] https://www.garmin.com/en-US/software/express/.

[5] https://www.pugetsystems.com/labs/articles/Impact-of-Temperature-on-Intel-CPU-Performance-606/.

[6] https://en.wikipedia.org/wiki/Nvidia-Jetson.

[7] Kannan, R. Ali, S. S. A., Farah, A., Adil, S. H., & Khan, A. (2017). Smart Wearable EEG Sensor. Procedia Computer Science, 105(December 2016), 138–143. https://doi.org/10.1016/j.procs.2017.01.193.

[8] Praveen, K Nanda, R. M., Venkata, A. B., & Kadakol, H. (2014). Modular Weather and Environment Monitoring Systems using Raspberry Pi, 3(9), 686–689.

[9] S. G. (2015). Wireless sensor network system using Raspberry Pi and zigbee for environmental monitoring applications. 2015 International Conference on Smart Technologies and Management for Computing, Communication, Controls, Energy and Materials (ICSTM), (May), 376–381. https://doi.org/10.1109/ICSTM.2015.7225445.

[10] Reddington, J., & Tintarev, N. (2011). Automatically Generating Stories from Sensor Data. Proceedings of the 16th International Conference on Intelligent User Interfaces, (November 2010), 407–410 https://doi.org/10.1145/1943403.1943477.

[11] Tomasi, M., Pundlik, S., Bowers, A. R., Peli, E., & Luo, G. (2016). Mobile gaze tracking system for outdoor walking behavioral studies. Journal of Vision, 16(3), 27. https://doi.org/10.1167/16.3.27.

[12] Wang, J., Zhang, G., & Shi, J. (2015). Pupil and glint detection using wearable camera sensor and near-infrared LED array. Sensors (Switzerland), 15(12), 30126–30141. https://doi.org/10.3390/s151229792.

[13] Praveen, K., Nanda, R. M., Venkata, A. B., & Kadakol, H. (2014). Modular Weather and Environment Monitoring Systems using Raspberry Pi, 3(9), 686–689.

[14] Ferdoush, S., & Li, X. (2014). Wireless sensor network system design using Raspberry Pi and Arduino for environmental monitoring applications. Procedia Computer Science, 34, 103–110. https://doi.org/10.1016/j.procs.2014.07.059.

[15] Sunrom.com, 'Serial Bluetooth Module 5V [1179] Sunrom Technologies'. [Online]. Available: http://www.sunrom.com/p/serialbluetooth- module-5v. [Accessed: 17-Aug- 2014].

[16] Usman, Ahmad, and Sajjad Haider Shami. "Evolution of communication technologies for smart grid applications." Renewable and Sustainable Energy Reviews 19 (2013): 191-199.

[17] Suzuki, Y.; Toyozumi, N.; Takahashi, L.; Guillaume, L.; Hosaka, H.; Itao, K.Wearable Individual Adapting Cooling System Using Smartphone and Heart Beat Sensor. In Proceedings of the SICE Annual Conference, Tsukuba, Japan, 20–23 September 2016; pp. 531–536.

[18] Wang, J.M.; Yang, M.T.; Chen, P.L. Design and Implementation of an IntelligentWindowsill System Using Smart Handheld Device and Fuzzy Microcontroller. Sensors 2017, 17, 830.

[19] Kaewkannate, K.; Kim, S. A comparison of wearable fitness devices. BMC Public Health 2016, 16, 433.

[20] Gerrett, N.; Redortier, B.; Voelcker, T.; Havenith, G. A comparison of galvanic skin conductance and skin wittedness as indicators of thermal discomfort during moderate and high metabolic rates. J. Therm. Biol.

[21] Zeller, L.; Novack, V.; Barski, L.; Almog, Y. Exertional heatstroke: Clinical characteristics, diagnostic and therapeutic considerations. Eur. J. Intern. Med. 2013, 22, 296–299.

[22] Iervolino, R., Bonavolonta, F., & Cavallari, A. (2017). A wearable device for sport performance analysis and monitoring. 2017 IEEE International Workshop on Measurement and Networking (M&N), (September), 1–6. https://doi.org/10.1109/IWMN.2017.8078375

[23] Mardonova, M., & Choi, Y. (2018). Review of Wearable Device Technology and Its Applications to the Mining Industry. Energies, 11(3), 547. https://doi.org/10.3390/en11030547

# 11 Appendix

NVIDIA TEGRA LINUX DRIVER PACKAGE QUICK-START GUIDE

The information here is intended to help you quickly get started using NVIDIA Tegra Linux Driver package (L4T).

ASSUMPTIONS:

1. You have a Jetson-tk1 Tegra Developer System, equipped with the NVIDIA Tegra K1 32 bit family processor.
2. You have a host machine that is running Linux.
3. Your developer system is cabled as follows:
4. USB Micro-B cable connecting Jetson-tk1 (J1E1 USB0) to your Linux host for flashing.
5. (Not included in the developer kit) Connect USB peripherals such as keyboard, mouse, and USB hub.
6. An HDMI cable plugged into "J1C1 HDMI1" on the target which is connected to an external HDMI display.
7. An Ethernet cable plugged into the J1D1 on board Ethernet port.
8. If you would like to connect to the debug console, a female to female NULL modem cable is required to plug into the serial port J1A2 UART4 on the target connected to your Linux host directly or through a serial-to-USB converter.
9. The following directions will create a 14 GiB partition on the eMMC device (internal storage) and will flash the root file system to that location.

**INSTRUCTIONS:**

1. Download the latest L4T release package for your developer system and the sample file system from https://developer.nvidia.com/linux-tegra
2. If NVIDIA does not yet provide public release for the developer system you have, please contact your NVIDIA support representative to obtain the latest L4T release package for use with the developer board.
3. Untar the files and assemble the rootfs:
4. Flash the rootfs onto the system's internal eMMC.

   a) Put your system into "reset recovery mode" by holding down the RECOVERY button and press RESET button once on the main board. b) Ensure your Linux host system is connected to the target device through the USB cable for flashing.

5. The target will automatically reboot upon completion of the flash. The command prompt will show up over the display that you have attached to the target. Log in as user login:ubuntu and password:ubuntu. All actions are completed unless you wish to configure the graphical desktop on your setup. You now have Linux running on your developer system.

6. Installing the graphical desktop on your target board (if not already installed):

   a) Make sure the Ethernet cable is connected. b) Use eth0 for the built-in Ethernet port:

   sudo dhclient eth0

   c) Check to see if Ethernet is up and running. You should see an IP address associated with eth0.

   ifconfig sudo apt-get update sudo apt-get install ubuntu-desktop

   d) Reboot and the system will boot to the graphical desktop.

   NOTE: The above steps can be used to install other packages with "sudo apt-get install".

## 11.1 List of equipment

1. Nvidia Jetson TK1 board.
2. BMP280 Sensor [Temperature, Pressure, Humidity].
3. Gas Sensor.
4. Dual lens camera module and Logitech webcam.
5. Garmin Vivoactive smart watch.
6. Microphone.
7. Multiple USB hub.
8. HDMI cable.
9. Ethernet cable.
10. External SD card.