University of Colorado Boulder

ECEN 5623: REAL TIME EMBEDDED SYSTEMS

TOUCH ME NOT

Under the guidance of Professor Tim Scherr

Deepesh Sonigra

Madhumitha Tolakanahalli Pradeep

Contents

INTRODUCTION

"TOUCH ME NOT" is a real time gesture detection tool which is capable of detecting the user's hand and counting the raised fingers. This tool can be used to interact with any computer system executing options based on the number of fingers raised. The tool can be integrated in multiple applications incorporating touchless interface for the users. It is cost-effective as it requires any type of display and a basic camera setup.

PROBLEMS SOLVED BY THE TOOL

- With keeping in mind, the present situation of COVID-19, the tool can fulfill the requirement of touchless user interface.
- The tool can be integrated with devices where proximity to the device is not possible.
- The tool can be further extended to detect and understand the sign language and convert it to an audio.

PRODUCT DESCRIPTION

We are designing a restaurant feedback system which asks users for feedback of the experience of the restaurant. The product runs on Jetson Nano which supports OpenCV and has an camera interface to capture the hand gestures. The product has a screen interface connected through and HDMI interface. Ideally the screen interface would be a user interaction screen executing some kind of a task. This prototype will have a user display providing a user interface, with ability to select multiple options on the screen

FUNCTIONAL CAPABILITIES

Hand Detection Capability

The hand detection would be done by detecting the contours in the binary image containing the hand. This is done by performing hand segmentation and background elimination of the image. Once the image is in a binary form, contours will return the hand in the image. Using the “convex hull” algorithm we will be finding the center of the hand of the user. This will give us a reference point to do scale invariant computations.

Fingers Detect Capability

The fingertips are required to be detected as we need to know the number of fingers lifted. This is done by finding the intersection points of the contour and the convex hull. We need to find the defects in the contour and using some filtration and calculations we can get the points near the fingertips with reference to the center of the hand. On averaging and filtering the points we can get 1 point per finger. The distance from the center to the fingertip will also be verified, leading to reduction in false positives.

Display UI

The Display UI is a user interface which interacts with the user based on the inputs it receives. The display UI is initialized, and a thread continuously monitors the inputs to the display and runs commands and transactions chosen by the user.

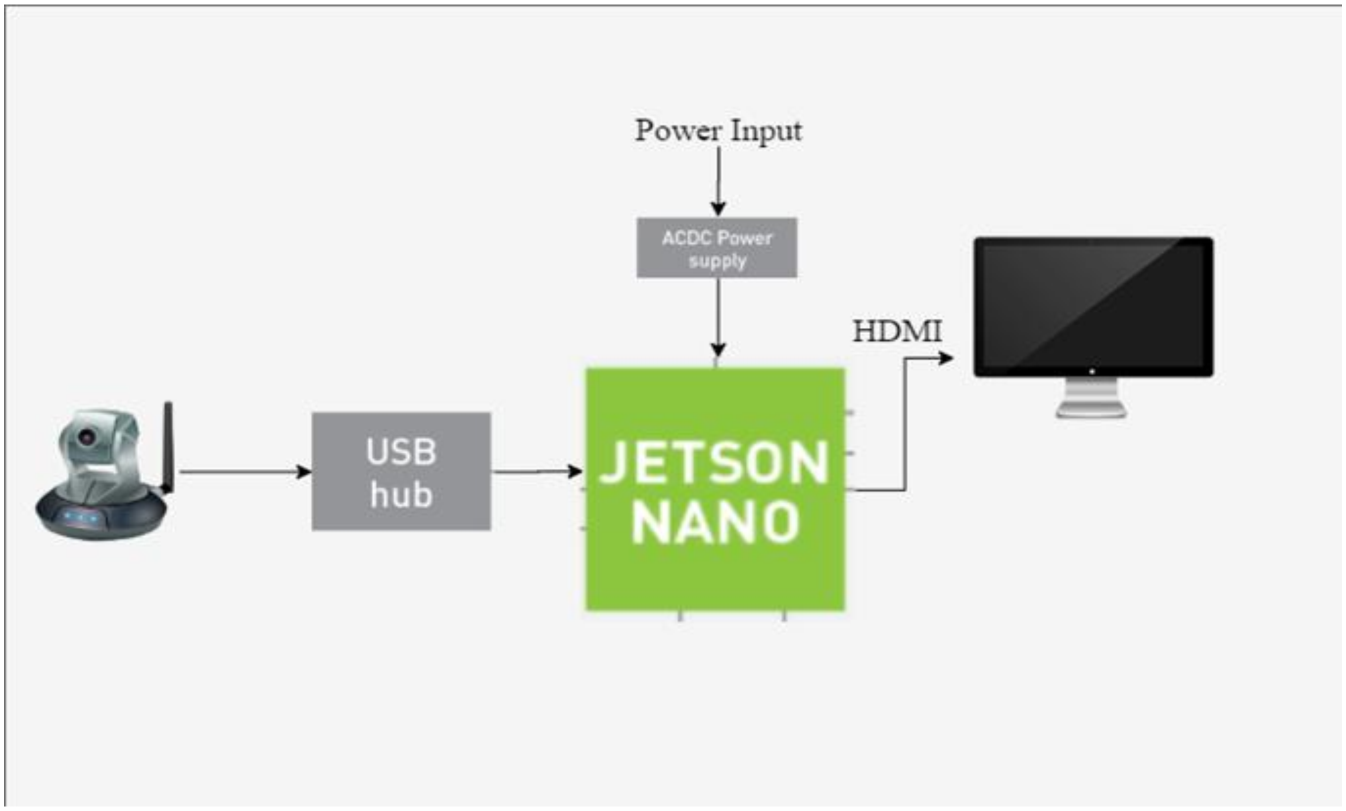
Changing Display Capability

Using non-blocking message queues provides the user to extend the application for continuously changing UI, without blocking on the UI for user input. This capability extends the application of this tool into multiple disciplines.

Real-Time Response

The tool is a replacement for touch displays, which have a real time response of the touch. This tool will be required to have similar response time and minimum deadline misses to make the user interactions real-time. The response time of the system is required to be within 1 second of the user gesturing for the feedback.

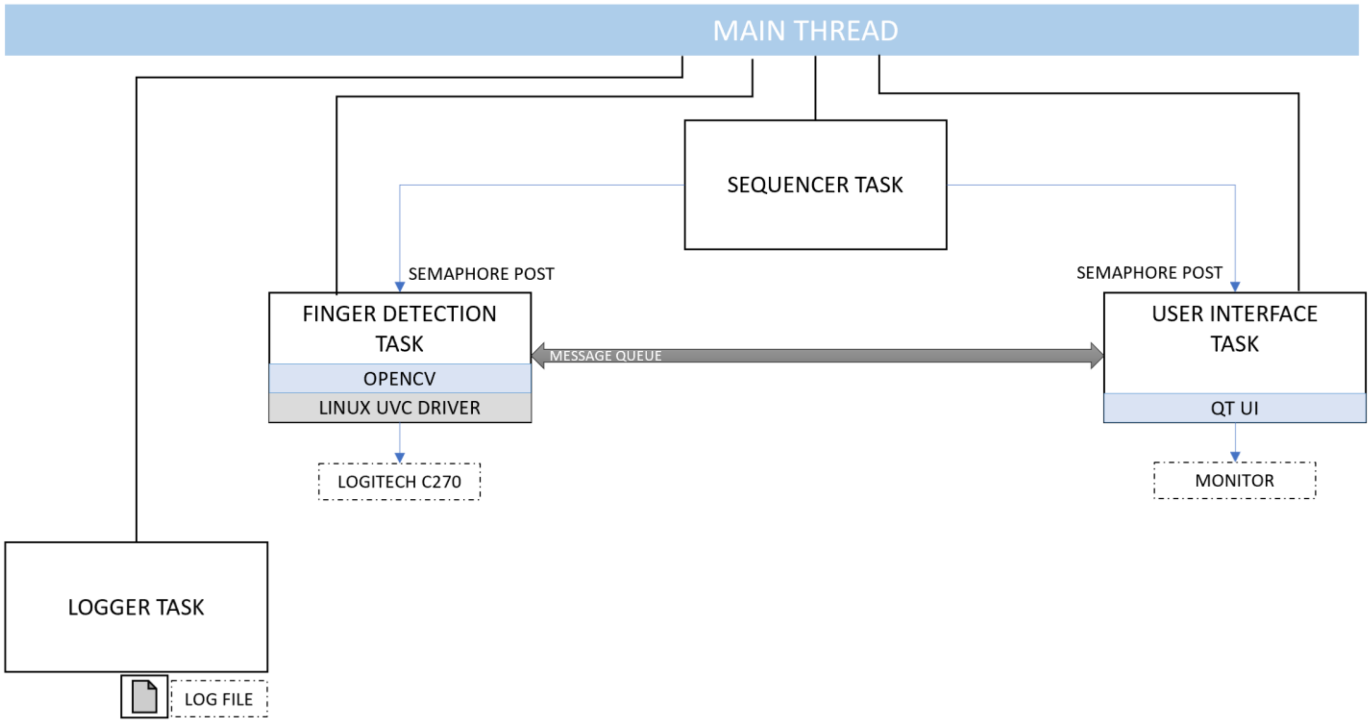
HARDWARE BLOCK DIAGRAM



The hardware components are:
- DC Power Supply (5V , 2A)
- Logitech 270 Camera 12MP
- Monitor/Screen (HDMI interface)

The software components are:
- Linux4Tegra (OS)
- OpenCV

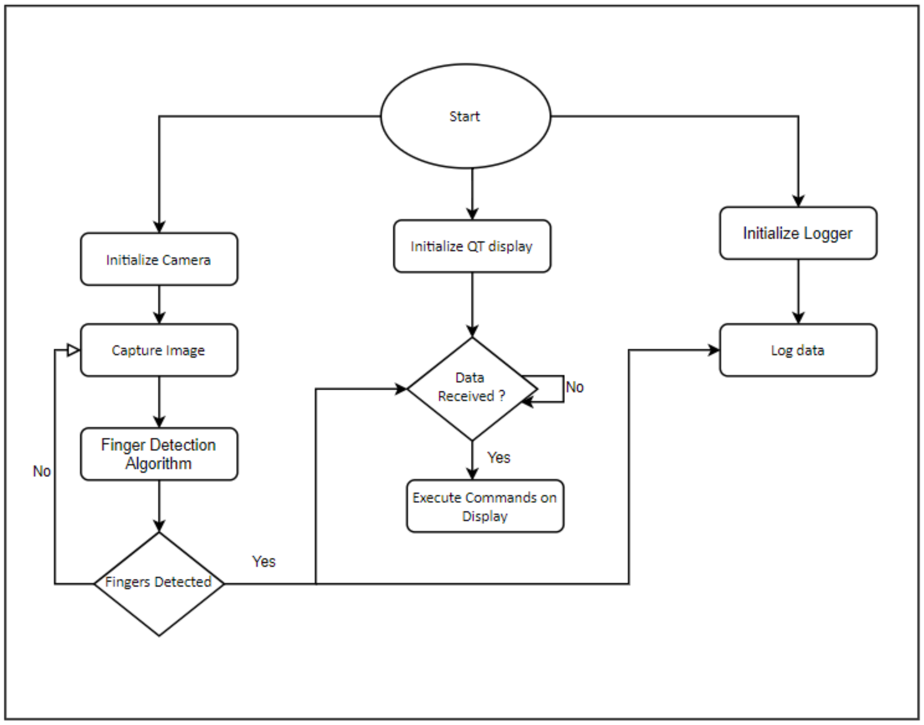
SOFTWARE BLOCK DIAGRAM



The Software Architecture is depicted in the above diagram. It involves multithreading where the main thread spawns four threads as described below

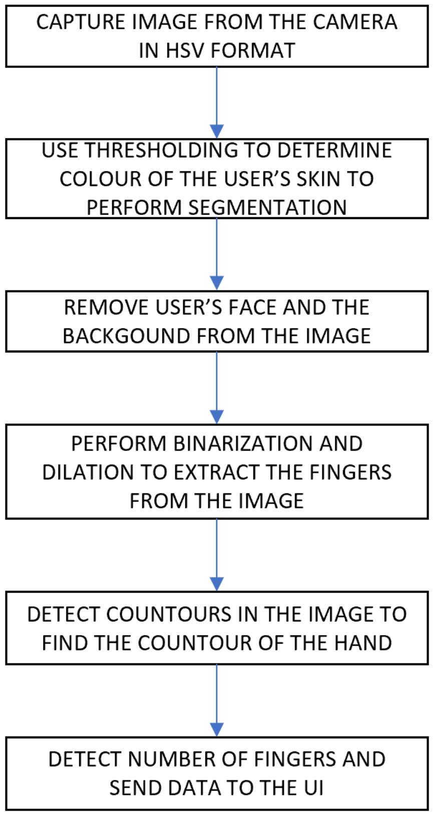
1. **SEQUENCER TASK** - This task schedules periodic tasks using semaphores to establish synchronization. The tasks are scheduled based on their periods.
2. **FINGER DETECTION TASK** - This task performs image processing on the image captured by the camera to detect the number of fingers raised. It uses OpenCV to perform various image processing functions like Thresholding, Binarization etc. Linux UVC Driver is used as an abstraction layer to connect to the camera.
3. **UI TASK** - This task consists of the sample menu that can be controlled using gestures. The data regarding the user gesture is communicated to the UI task from the Finger Detection Task using POSIX Message Queues.
4. **LOGGER TASK** - This task logs all the events that take place in the system for post-failure debugging purposes. The other tasks send their logging data to the Logger Task via POSIX Message Queues. This task can only be enabled for debugging purposes.

FLOWCHART



- The above diagram depicts the flowchart that we are designing our project on.
- The scheduler schedules 2 tasks based on their periods.
- The camera task captures the image, performs detection algorithms, and sends data to the Display UI
- Once the data is received by the display task, updates the display accordingly
- The Logger task logs all these events and information captured and sent
- The camera task goes back to capturing the image and processing the image.
- The Display task moves on to another display.

DATA FLOW DIAGRAM

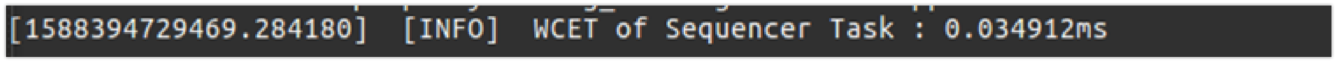

CAPTURE IMAGE FROM THE CAMERA IN HSV FORMAT
USE THRESHOLDING TO DETERMINE COLOUR OF THE USER'S SKIN TO PERFORM SEGMENTATION
REMOVE USER'S FACE AND THE BACKGOUND FROM THE IMAGE
PERFORM BINARIZATION AND DILATION TO EXTRACT THE FINGERS FROM THE IMAGE
DETECT COUNTOURS IN THE IMAGE TO FIND THE COUNTOUR OF THE HAND
DETECT NUMBER OF FINGERS AND SEND DATA TO THE UI

- The above diagram depicts the flow of control in the project.
- First, the image is captured from the camera
- Then, the image goes through a series of image processing steps to obtain the data regarding the number of fingers detected.
- The image processing steps include Thresholding, Binarization, Background Detection, Hand Contouring and Finger Identification
- Data approximation methods are applied on the dataset.
- Then, the data is passed on the GUI to display the data accordingly.

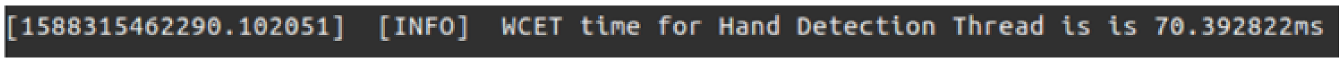
REAL-TIME SERVICE REQUIREMENTS

TASK	PERIOD(ms)	PRIORITY	WCET	DEADLINE	FREQUENCY
Sequencer	100ms	MAX - 1	0.034ms	100ms	10Hz
Finger Detection	100ms	MAX - 2	70.4ms	100ms	10Hz
User Interface	500ms	MAX - 3	38.86ms	500ms	1Hz
Logger	x	x	x	x	x

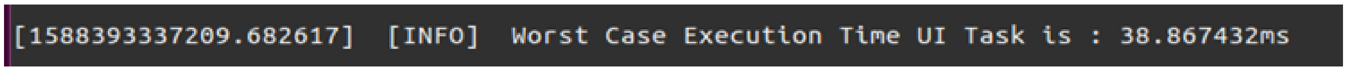
- The expected response time of this system is to react within 1 second of the user's hand gesture. So within one second, the system detects the number of fingers lifted, sends to the UI and UI updates the display accordingly.
- The sequencer runs at 10Hz scheduling both the Finger Detection Task (10Hz) and UI task (2Hz)
- The Finger detection task performs image processing and decision making on the number of fingers lifted.
- The worst computation time of finger detection task is 70.4ms. The Finger detection task runs for 5 times before it sends the option selected to the Display task.
- As the response of the Finger Detection task should be 0.5 seconds where it could send the option processed, the deadline was chosen to be 100ms
- This provides stability with gesture detection, eliminating random hand movements and increasing efficiency of the system.
- The UI task receives information from the Finger Detection Task and updates the display accordingly.
- The worst computation time of the Display UI task is 38.86ms. The deadline of this task is set to 500ms. The deciding factor was the minimum time the user would require processing the information.
- The rate monotonic scheduling algorithm will be used to schedule tasks, that is based on their time period, the priority of the task will be assigned.
- The period and the deadline of the tasks would be the same as the rate monotonic scheduling algorithm is selected.
- In this case the Sequencer is assigned the highest priority as it will be scheduling tasks.
- The Finger Detection Task has a higher priority than the UI task as it has a lower deadline.
- The logger task is not a periodic task and would only perform when an event or an exception occurs. These log snippets confirm the worst execution time of these tasks.



```
[1588394729469.284180]  [INFO]  WCET of Sequencer Task : 0.034912ms
```

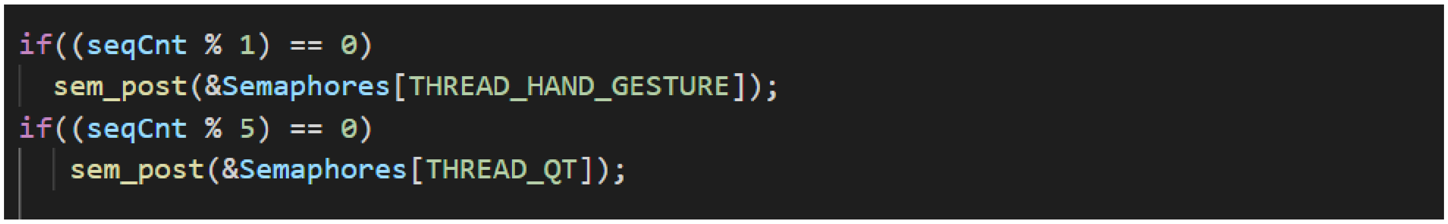


```
[1588315462290.102051]  [INFO]  WCET time for Hand Detection Thread is is 70.392822ms
```



```
[1588393337209.682617]  [INFO]  Worst Case Execution Time UI Task is : 38.867432ms
```

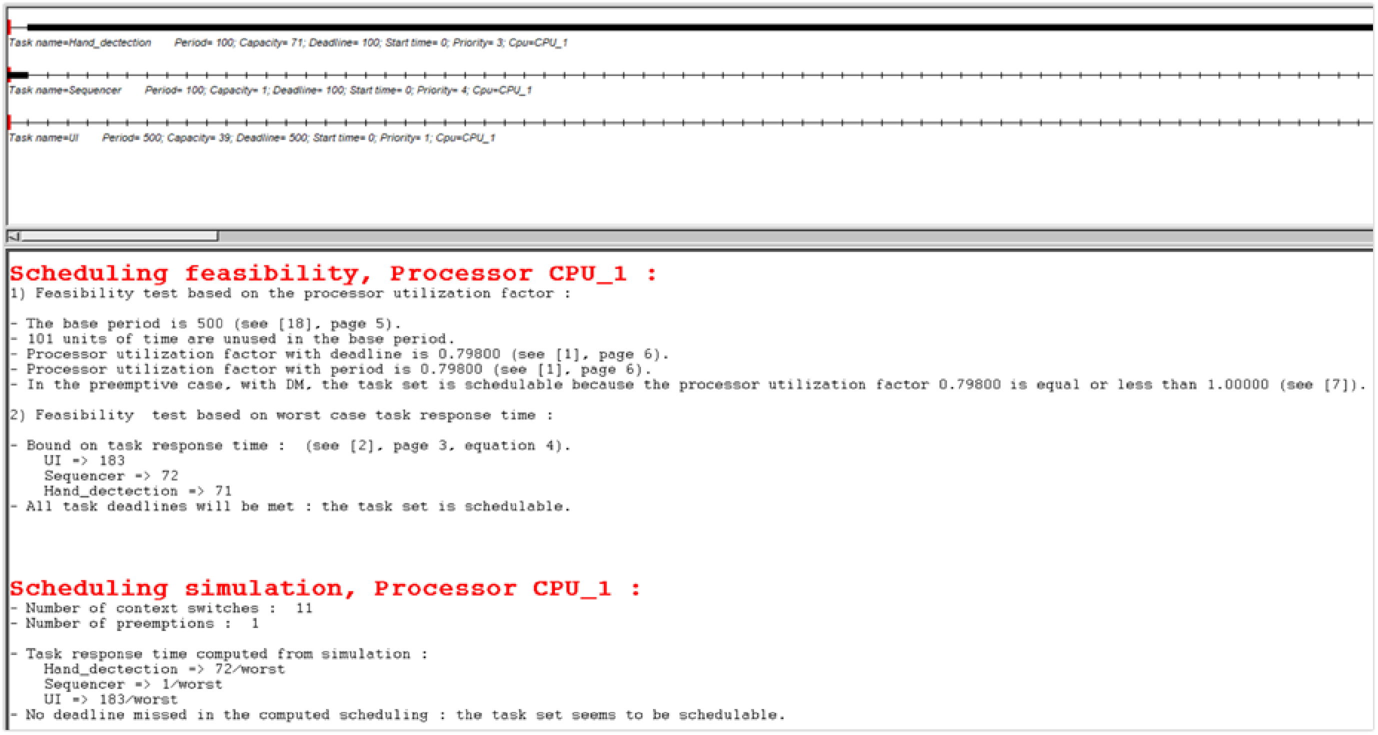
SEQUENCER CODE SNIPPET



```
if((seqCnt % 1) == 0)
  sem_post(&Semaphores[THREAD_HAND_GESTURE]);
if((seqCnt % 5) == 0)
    sem_post(&Semaphores[THREAD_QT]);
```

This code snippet confirms the scheduling of the tasks using a sequencer at desired frequency.

CHEDDAR DIAGRAM



```
Task name=Hand_dectection      Period= 100; Capacity= 71; Deadline= 100; Start time= 0; Priority= 3; Cpu=CPU_1
Task name=Sequencer      Period= 100; Capacity= 1; Deadline= 100; Start time= 0; Priority= 4; Cpu=CPU_1
Task name=UI      Period= 500; Capacity= 39; Deadline= 500; Start time= 0; Priority= 1; Cpu=CPU_1
```

```
Scheduling feasibility, Processor CPU_1 :
1) Feasibility test based on the processor utilization factor :

- The base period is 500 (see [18], page 5).
- 101 units of time are unused in the base period.
- Processor utilization factor with deadline is 0.79800 (see [1], page 6).
- Processor utilization factor with period is 0.79800 (see [1], page 6).
- In the preemptive case, with DM, the task set is schedulable because the processor utilization factor 0.79800 is equal or less than 1.00000 (see [7]).

2) Feasibility  test based on worst case task response time :

- Bound on task response time :  (see [2], page 3, equation 4).
    UI => 183
    Sequencer => 72
    Hand_dectection => 71
- All task deadlines will be met : the task set is schedulable.



Scheduling simulation, Processor CPU_1 :
- Number of context switches :  11
- Number of preemptions :  1

- Task response time computed from simulation :
    Hand_dectection => 72/worst
    Sequencer => 1/worst
    UI => 183/worst
- No deadline missed in the computed scheduling : the task set seems to be schedulable.
```

- The Cheddar Analysis shows that not task would miss a deadline
- All tasks seem to be schedulable and feasible.
- The processor utilization factor is 79.8%

PERFORMANCE REQUIREMENTS

- The tasks are scheduled using Rate Monotonic Scheduling Algorithm assigning highest priority to task with least period.
- The period and deadline of the tasks has been calculated with an additional safety margin considering the worst-case execution of the tasks.
- The safety margin of the system is calculated considering the worst-case execution time of the tasks and is calculated for the LCM of the periods of the tasks which is 500ms

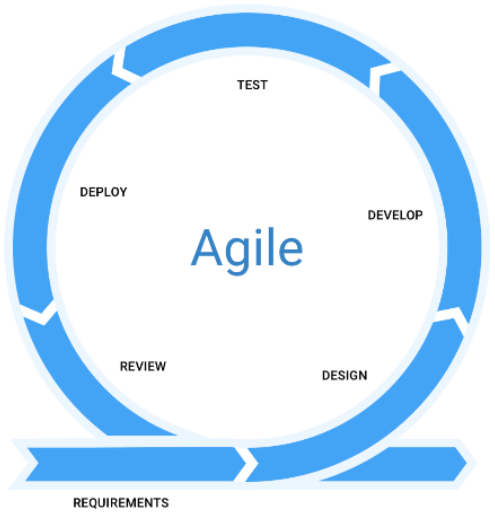
$$Safety\ Margin\ = 500\ -\ Execution\ Time\ of\ all\ Tasks$$
$$Execution\ Time\ for\ a\ Task = N\ *\ WCET\ of\ Task$$

TASKS	FREQUENCY	EXECUTION TIME	TOTAL EXECUTION TIME 500ms
Sequencer	5	0.034ms	0.17ms
Finger Detection	5	70.4ms	352ms
Display UI	1	38.86ms	38.86ms

- The total Execution time calculated is 391.03ms.
- The safety margin for the system for 500ms is around 109ms.This safety margin accounts for logging and interference time for the tasks.
- Using RM LUB where U < m($2^{1/m}$ - 1) = 0.78. The Percentage Utilization for our design is slightly over the RM LUB Limit. In future iterations, we could try to optimize the code so that it can satisfy these requirements.

VERIFICATION AND VALIDATION

- We used the Agile Methodology for the development of this project.
- With each incremental requirement, we followed the steps of Design, Development, Testing and Review to help detect bugs as early as possible,
- This model helped extensively with collaboration during the times of social distancing.
- With the integration of each requirement, we performed System Testing to ensure it did not break the existing system.



TEST PLAN

	TEST CASE	CRITERIA FOR PASSING	OBSERVED RESULT	PASS/FAIL
1.	FIFO scheduling of task	Scheduling priority should return SCHED_FIFO	Print scheduler function prints SHCED_FIFO	Passed
2.	Priority based scheduling	Highest priority task should be scheduled first	Log file indicates that the task with highest priority is scheduled first	Passed
3.	Cheddar Analysis	Task needs to be schedulable	No task misses deadline	Passed
4.	Feasibility Test	The task should be feasible	After running feasibility test, the task seems to be feasible	Passed
5.	Test the contours for detecting the hand	Binarization of image should show the hand in the image	Binarization of image shows the hand	Passed
6.	Capable of Detecting Lifted fingers	Ability to detect defects and end points	Can detect defects and end points to calculate lifted fingers	Passed
7.	Response of system with no hands	It should detect 0 fingers lifted	The response is 0 fingers	Passed
8.	Message Queues	Task should be able to communicate using message queues	Both tasks exchange information using message queues	Passed
9.	UI updating the display	UI updates the option selected within 1 second	UI updates the response by marking a circle	Passed
10.	Logging capability	Logging task logs the data accurately and synchronously	Logging task logs synchronously in the log file	Passed

PROOF OF CONCEPT

PLATFORM RESOURCES

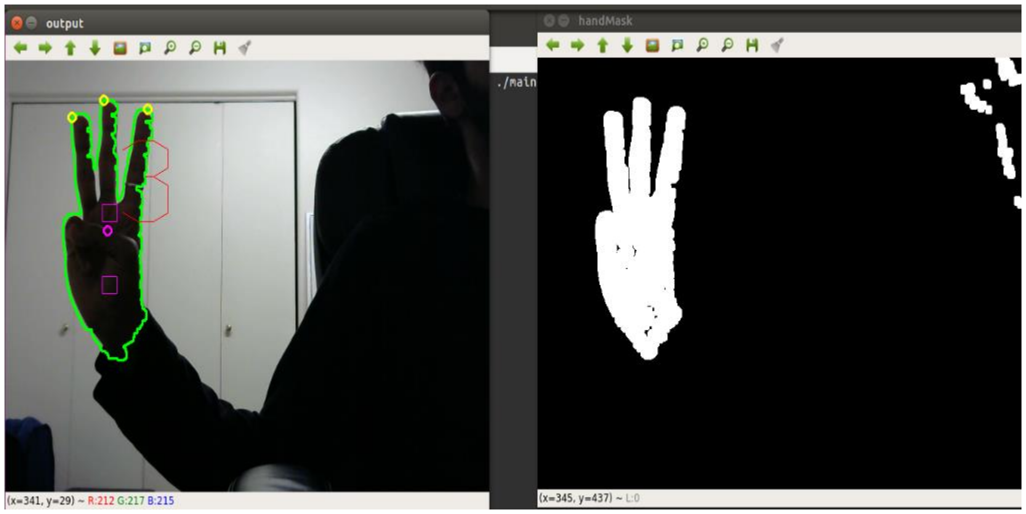
Jetson NANO

CPU	Quad Core ARM A57 @ 1.43GHz1.43GHz
GPU	128-core Maxwell
Memory	4GB 64-bit LPDDR4 25.6GB/s
Storage	microSD Card 64GB
Video Encode	4K @ 30 | 4x 1080p @ 30 | 9x 720p @ 30 (H.264/H.265)
Video Decode	4K @ 60 | 2x 4K @ 30 | 8x 1080p @ 30 | 18x 720p @ 30
Camera	2x MIPI CSI-2 DPHY lanes
Display	HDMI and DisplayPort

IMPLEMENTATION OF KEY SERVICES

FINGER DETECTION SERVICE

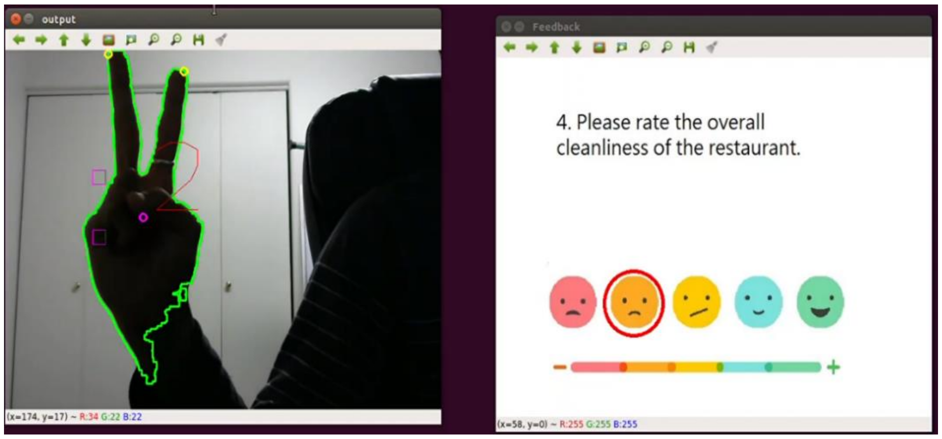
- The finger detection task is scheduled by the Sequencer by posting its corresponding semaphore every 10Hz i.e. every cycle of the sequencer and is run at a priority level, one less than the highest priority.
- First, it calibrates the algorithm to remove the background objects and retain the hand of the user to track fingers raised using – Thresholding, Skin Detection, Segmentation, Background Removal, Binarization.
- The number of fingers raised is calculated using Contour Detection which detects the fingertips by finding the intersection points of the contour and the convex hull
- The input data is sampled 5 times to ensure accuracy of number of fingers raised. This is accounted for when calculating the task's deadline.
- The detected option is sent to the UI Task using non-blocking message queues.



UI DISPLAY SERVICE

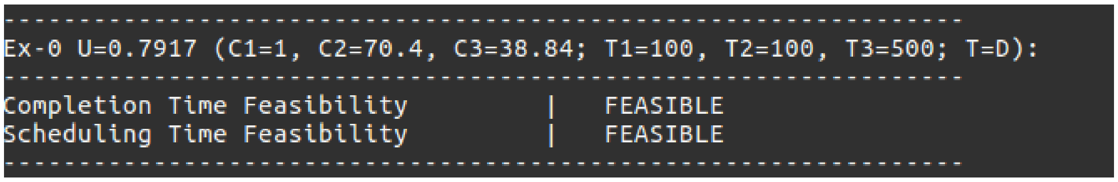
- The UI Display task is scheduled by the Sequencer by posting its corresponding semaphore every 2Hz i.e. every fifth cycle of the sequencer and is run at a priority level, one less than that of the Finger Detection Task.
- A screen with the appropriate menu is displayed to the user – in our application, a short feedback for a restaurant with five options.
- This Task receives the option chosen by the user via Message Queue. The design decision to use non-blocking queues was the corner case of an empty queue – this would cause the thread to be put into blocked state, possibly missing its deadline.

- The received option gets circled on the menu (as shown) and the thread moves on to the next page. Once all the feedback pages are exhausted, the questions get reset.


output
Feedback
4. Please rate the overall cleanliness of the restaurant.
(x=174, y=17) ~ R:34 G:22 B:22
(x=58, y=0) ~ R:255 G:255 B:255

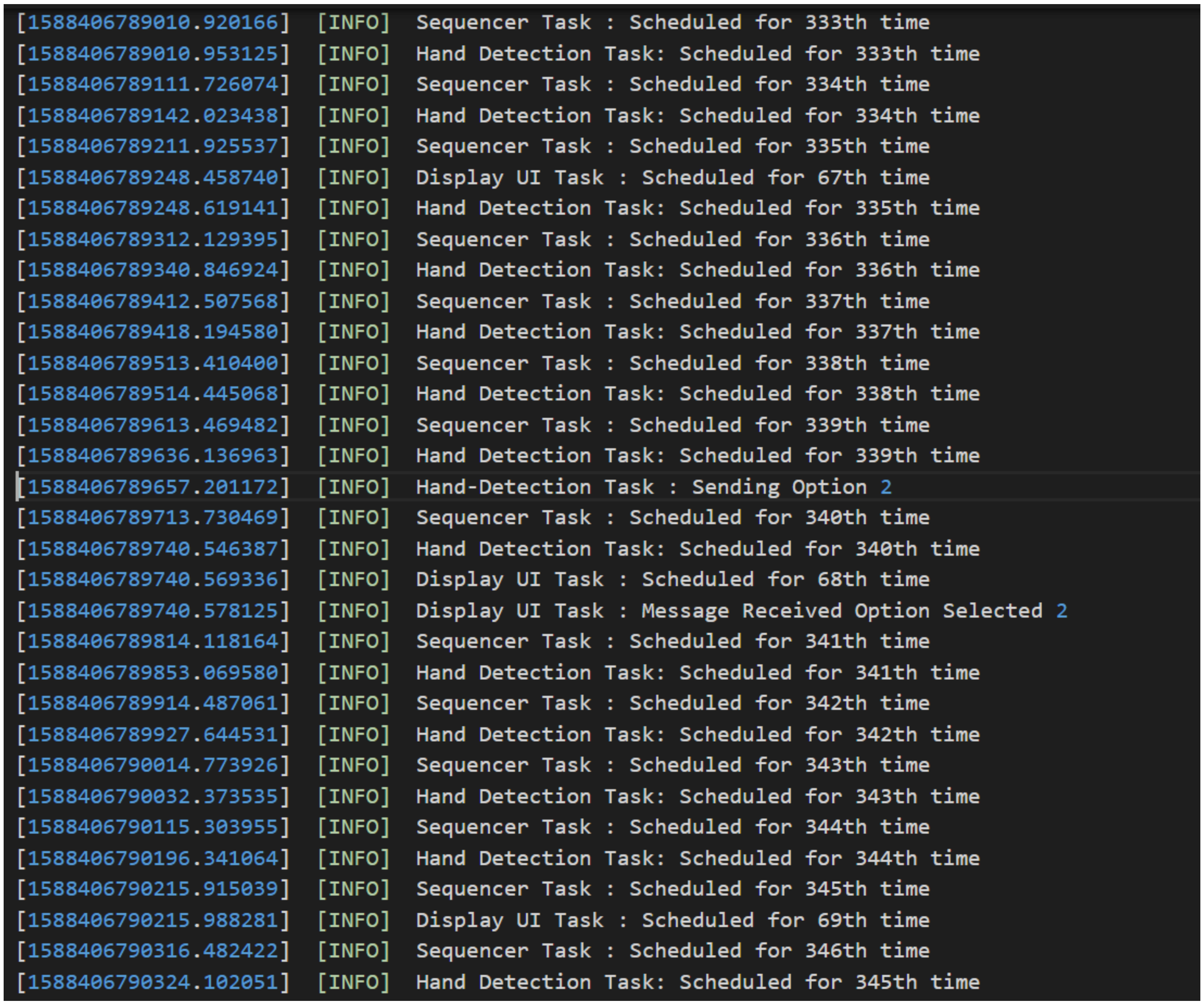
ANALYSIS

- One of the key areas of the project was to ensure the key services do not block each other and hog the CPU time. Based on this point, the deadlines were selected. Enough margins were provided to ensure the key services do not miss their deadlines.
- The choice of using Rate Monotonic Scheduling Algorithm was to ensure priority was given to tasks with higher frequency in case the system fails. To explain further, even if the UI task misses its deadline, the rest of the system continues to meet its real time requirement.
- Apart from performing the Cheddar Analysis, we plugged in our Period, Deadline and Computation Time values into the Feasibility Analysis Test from Exercise 2 to verify the same. The results from the test as shown below.
- The results show that the Utilization factor is 0.7917 and the tasks seems to be feasible.



```
----------------------------------------------------------------
Ex-0 U=0.7917 (C1=1, C2=70.4, C3=38.84; T1=100, T2=100, T3=500; T=D):
----------------------------------------------------------------
Completion Time Feasibility        |    FEASIBLE
Scheduling Time Feasibility        |    FEASIBLE
----------------------------------------------------------------
```

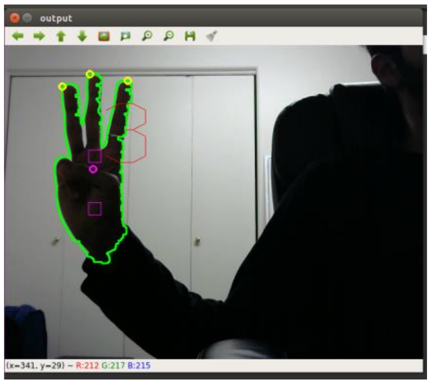
TIME STAMP TRACING



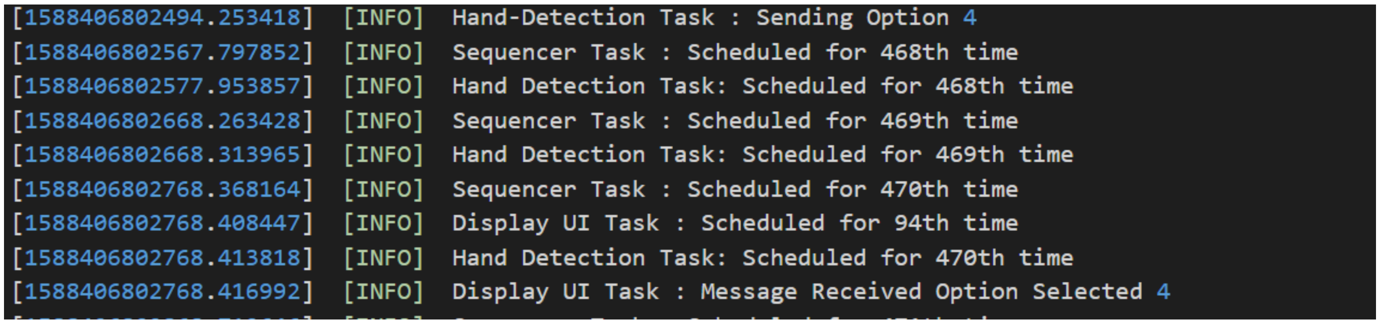
```
[1588406789010.920166]  [INFO]  Sequencer Task : Scheduled for 333th time
[1588406789010.953125]  [INFO]  Hand Detection Task: Scheduled for 333th time
[1588406789111.726074]  [INFO]  Sequencer Task : Scheduled for 334th time
[1588406789142.023438]  [INFO]  Hand Detection Task: Scheduled for 334th time
[1588406789211.925537]  [INFO]  Sequencer Task : Scheduled for 335th time
[1588406789248.458740]  [INFO]  Display UI Task : Scheduled for 67th time
[1588406789248.619141]  [INFO]  Hand Detection Task: Scheduled for 335th time
[1588406789312.129395]  [INFO]  Sequencer Task : Scheduled for 336th time
[1588406789340.846924]  [INFO]  Hand Detection Task: Scheduled for 336th time
[1588406789412.507568]  [INFO]  Sequencer Task : Scheduled for 337th time
[1588406789418.194580]  [INFO]  Hand Detection Task: Scheduled for 337th time
[1588406789513.410400]  [INFO]  Sequencer Task : Scheduled for 338th time
[1588406789514.445068]  [INFO]  Hand Detection Task: Scheduled for 338th time
[1588406789613.469482]  [INFO]  Sequencer Task : Scheduled for 339th time
[1588406789636.136963]  [INFO]  Hand Detection Task: Scheduled for 339th time
[1588406789657.201172]  [INFO]  Hand-Detection Task : Sending Option 2
[1588406789713.730469]  [INFO]  Sequencer Task : Scheduled for 340th time
[1588406789740.546387]  [INFO]  Hand Detection Task: Scheduled for 340th time
[1588406789740.569336]  [INFO]  Display UI Task : Scheduled for 68th time
[1588406789740.578125]  [INFO]  Display UI Task : Message Received Option Selected 2
[1588406789814.118164]  [INFO]  Sequencer Task : Scheduled for 341th time
[1588406789853.069580]  [INFO]  Hand Detection Task: Scheduled for 341th time
[1588406789914.487061]  [INFO]  Sequencer Task : Scheduled for 342th time
[1588406789927.644531]  [INFO]  Hand Detection Task: Scheduled for 342th time
[1588406790014.773926]  [INFO]  Sequencer Task : Scheduled for 343th time
[1588406790032.373535]  [INFO]  Hand Detection Task: Scheduled for 343th time
[1588406790115.303955]  [INFO]  Sequencer Task : Scheduled for 344th time
[1588406790196.341064]  [INFO]  Hand Detection Task: Scheduled for 344th time
[1588406790215.915039]  [INFO]  Sequencer Task : Scheduled for 345th time
[1588406790215.988281]  [INFO]  Display UI Task : Scheduled for 69th time
[1588406790316.482422]  [INFO]  Sequencer Task : Scheduled for 346th time
[1588406790324.102051]  [INFO]  Hand Detection Task: Scheduled for 345th time
```

- The Log file demonstrates the scheduling of each task, timestamps, and number of times it is scheduled.
- This verifies that the finger detection task gets scheduled every time the sequencer is scheduled.
- The Display task get scheduled once in five times of the sequencer scheduled.
- The log file also verifies the transfer of the data from the finger detection task to the Display UI task – in this case, Option 2 is chosen by the user and a message is sent from the former to latter.

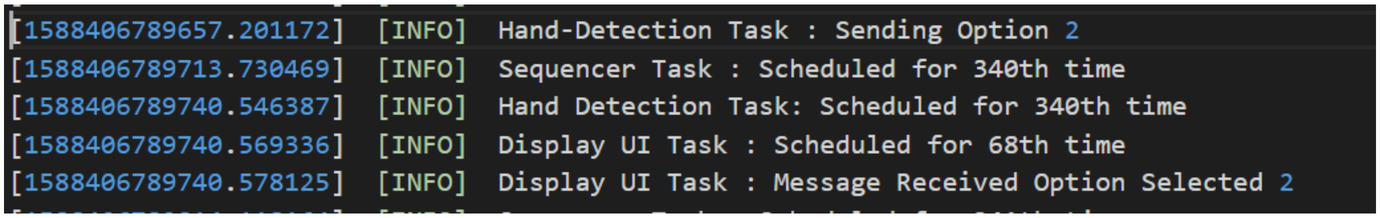
TEST CASES



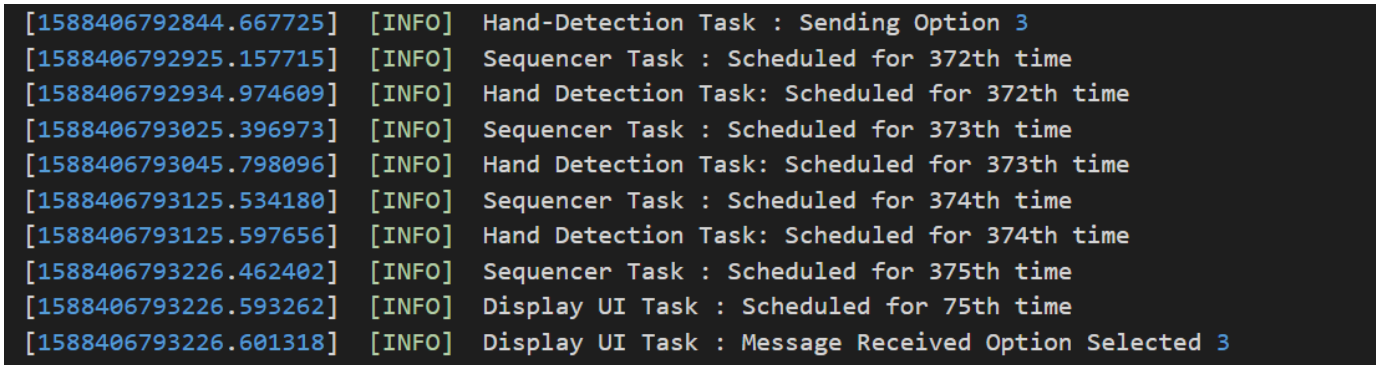
- As can be seen in the above image, the program was able to detect the number of fingers raised accurately.



```
[1588406802494.253418]  [INFO]  Hand-Detection Task : Sending Option 4
[1588406802567.797852]  [INFO]  Sequencer Task : Scheduled for 468th time
[1588406802577.953857]  [INFO]  Hand Detection Task: Scheduled for 468th time
[1588406802668.263428]  [INFO]  Sequencer Task : Scheduled for 469th time
[1588406802668.313965]  [INFO]  Hand Detection Task: Scheduled for 469th time
[1588406802768.368164]  [INFO]  Sequencer Task : Scheduled for 470th time
[1588406802768.408447]  [INFO]  Display UI Task : Scheduled for 94th time
[1588406802768.413818]  [INFO]  Hand Detection Task: Scheduled for 470th time
[1588406802768.416992]  [INFO]  Display UI Task : Message Received Option Selected 4
```



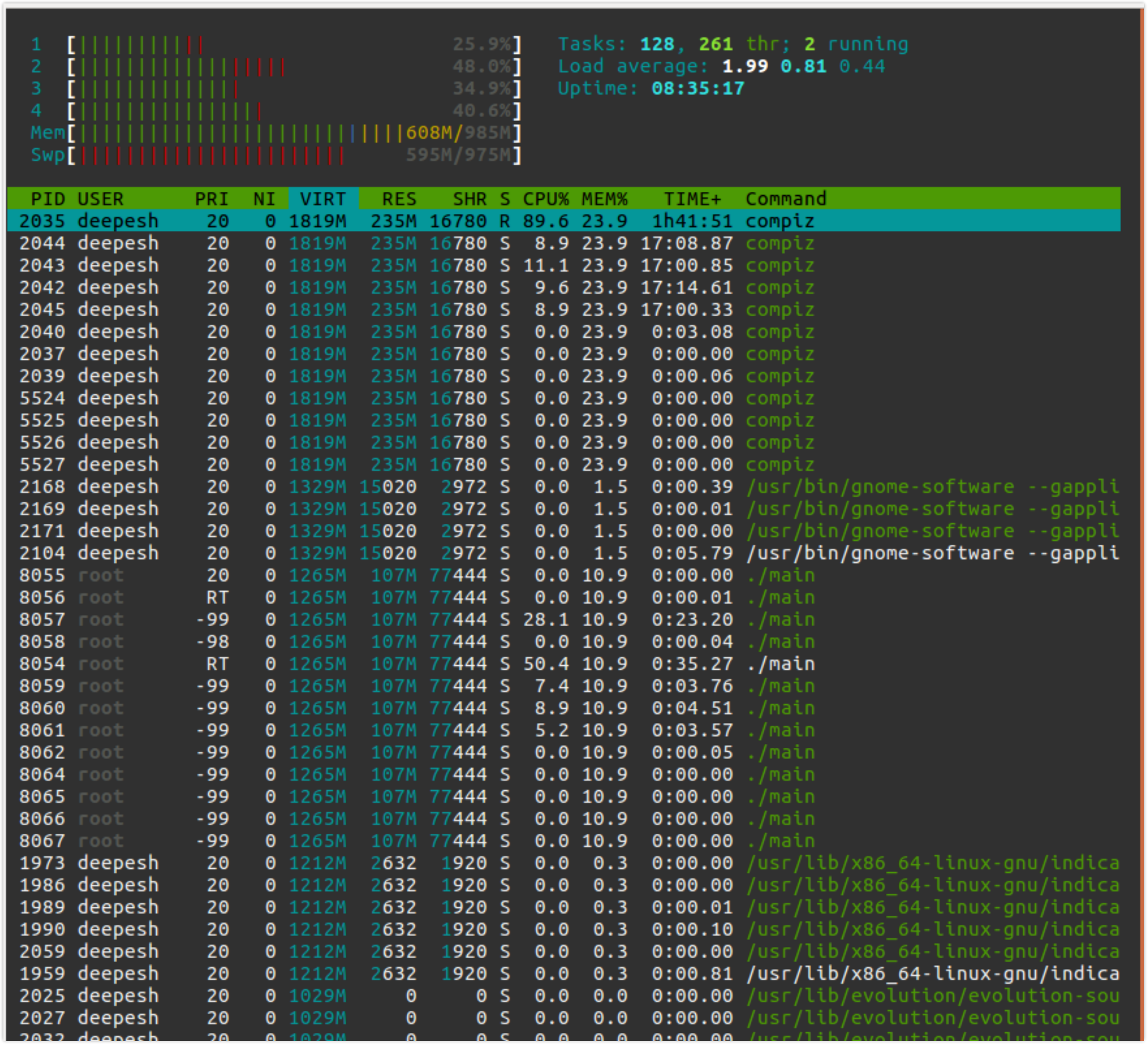
```
[1588406789657.201172]  [INFO]  Hand-Detection Task : Sending Option 2
[1588406789713.730469]  [INFO]  Sequencer Task : Scheduled for 340th time
[1588406789740.546387]  [INFO]  Hand Detection Task: Scheduled for 340th time
[1588406789740.569336]  [INFO]  Display UI Task : Scheduled for 68th time
[1588406789740.578125]  [INFO]  Display UI Task : Message Received Option Selected 2
```



```
[1588406792844.667725]  [INFO]  Hand-Detection Task : Sending Option 3
[1588406792925.157715]  [INFO]  Sequencer Task : Scheduled for 372th time
[1588406792934.974609]  [INFO]  Hand Detection Task: Scheduled for 372th time
[1588406793025.396973]  [INFO]  Sequencer Task : Scheduled for 373th time
[1588406793045.798096]  [INFO]  Hand Detection Task: Scheduled for 373th time
[1588406793125.534180]  [INFO]  Sequencer Task : Scheduled for 374th time
[1588406793125.597656]  [INFO]  Hand Detection Task: Scheduled for 374th time
[1588406793226.462402]  [INFO]  Sequencer Task : Scheduled for 375th time
[1588406793226.593262]  [INFO]  Display UI Task : Scheduled for 75th time
[1588406793226.601318]  [INFO]  Display UI Task : Message Received Option Selected 3
```

- We verified if each option was detected accurately as demonstrated in the logs shown above.

PROFILING



```
  1  [||||||||||||                      25.9%]   Tasks: 128, 261 thr; 2 running
  2  [||||||||||||||||||                48.0%]   Load average: 1.99 0.81 0.44
  3  [||||||||||||||                    34.9%]   Uptime: 08:35:17
  4  [||||||||||||||||                  40.6%]
  Mem[||||||||||||||||||||||||||||608M/985M]
  Swp[|||||||||||||||||||||||     595M/975M]

  PID USER      PRI  NI  VIRT   RES   SHR S CPU% MEM%   TIME+  Command
 2035 deepesh    20   0 1819M  235M 16780 R 89.6 23.9  1h41:51 compiz
 2044 deepesh    20   0 1819M  235M 16780 S  8.9 23.9 17:08.87 compiz
 2043 deepesh    20   0 1819M  235M 16780 S 11.1 23.9 17:00.85 compiz
 2042 deepesh    20   0 1819M  235M 16780 S  9.6 23.9 17:14.61 compiz
 2045 deepesh    20   0 1819M  235M 16780 S  8.9 23.9 17:00.33 compiz
 2040 deepesh    20   0 1819M  235M 16780 S  0.0 23.9  0:03.08 compiz
 2037 deepesh    20   0 1819M  235M 16780 S  0.0 23.9  0:00.00 compiz
 2039 deepesh    20   0 1819M  235M 16780 S  0.0 23.9  0:00.06 compiz
 5524 deepesh    20   0 1819M  235M 16780 S  0.0 23.9  0:00.00 compiz
 5525 deepesh    20   0 1819M  235M 16780 S  0.0 23.9  0:00.00 compiz
 5526 deepesh    20   0 1819M  235M 16780 S  0.0 23.9  0:00.00 compiz
 5527 deepesh    20   0 1819M  235M 16780 S  0.0 23.9  0:00.00 compiz
 2168 deepesh    20   0 1329M 15020  2972 S  0.0  1.5  0:00.39 /usr/bin/gnome-software --gappli
 2169 deepesh    20   0 1329M 15020  2972 S  0.0  1.5  0:00.01 /usr/bin/gnome-software --gappli
 2171 deepesh    20   0 1329M 15020  2972 S  0.0  1.5  0:00.00 /usr/bin/gnome-software --gappli
 2104 deepesh    20   0 1329M 15020  2972 S  0.0  1.5  0:05.79 /usr/bin/gnome-software --gappli
 8055 root       20   0 1265M  107M 77444 S  0.0 10.9  0:00.00 ./main
 8056 root       RT   0 1265M  107M 77444 S  0.0 10.9  0:00.01 ./main
 8057 root      -99   0 1265M  107M 77444 S 28.1 10.9  0:23.20 ./main
 8058 root      -98   0 1265M  107M 77444 S  0.0 10.9  0:00.04 ./main
 8054 root       RT   0 1265M  107M 77444 S 50.4 10.9  0:35.27 ./main
 8059 root      -99   0 1265M  107M 77444 S  7.4 10.9  0:03.76 ./main
 8060 root      -99   0 1265M  107M 77444 S  8.9 10.9  0:04.51 ./main
 8061 root      -99   0 1265M  107M 77444 S  5.2 10.9  0:03.57 ./main
 8062 root      -99   0 1265M  107M 77444 S  0.0 10.9  0:00.05 ./main
 8064 root      -99   0 1265M  107M 77444 S  0.0 10.9  0:00.00 ./main
 8065 root      -99   0 1265M  107M 77444 S  0.0 10.9  0:00.00 ./main
 8066 root      -99   0 1265M  107M 77444 S  0.0 10.9  0:00.00 ./main
 8067 root      -99   0 1265M  107M 77444 S  0.0 10.9  0:00.00 ./main
 1973 deepesh    20   0 1212M  2632  1920 S  0.0  0.3  0:00.00 /usr/lib/x86_64-linux-gnu/indica
 1986 deepesh    20   0 1212M  2632  1920 S  0.0  0.3  0:00.00 /usr/lib/x86_64-linux-gnu/indica
 1989 deepesh    20   0 1212M  2632  1920 S  0.0  0.3  0:00.01 /usr/lib/x86_64-linux-gnu/indica
 1990 deepesh    20   0 1212M  2632  1920 S  0.0  0.3  0:00.10 /usr/lib/x86_64-linux-gnu/indica
 2059 deepesh    20   0 1212M  2632  1920 S  0.0  0.3  0:00.00 /usr/lib/x86_64-linux-gnu/indica
 1959 deepesh    20   0 1212M  2632  1920 S  0.0  0.3  0:00.81 /usr/lib/x86_64-linux-gnu/indica
 2025 deepesh    20   0 1029M     0     0 S  0.0  0.0  0:00.00 /usr/lib/evolution/evolution-sou
 2027 deepesh    20   0 1029M     0     0 S  0.0  0.0  0:00.00 /usr/lib/evolution/evolution-sou
 2032 deepesh    20   0 1029M     0     0 S  0.0  0.0  0:00.00 /usr/lib/evolution/evolution-sou
```

The process is almost evenly distributed on all cores

CPU CORE	UTILIZATION
CPU - 1	26%
CPU – 2	48%
CPU – 3	34%
CPU – 4	40%

PERSONAL CONTRIBUTION

DEEPESH SONIGRA

I was responsible for the OpenCV setup on Jetson Nano, testing hand segmentation, background removal algorithm and counting fingers algorithm. I was responsible for the binarization of the image and detecting the hand and fingers in the image. The worst-case execution time of the task was calculated by running each task, one at a time. I implemented a logger task to log the system failures along with timestamps, providing debugging capability to the system.

MADHUMITHA TOLAKANHALLI PRADEEP

I will be writing an application peripheral interface for Display UI, response, and error handling of the display task. I implemented the least execution time algorithms to make the response of the system as real time as possible. I executed these tasks on Cheddar conducting the feasibility test and schedulability test. The code integration of all the processes was conducted and I calculated the deadline, period, and worst-case execution time of the finger counting service. The inter thread communication and synchronization was done using message queues and semaphores, avoiding race conditions.

LESSONS LEARNT

- Gained significant knowledge about real-time scheduling and analysis.
- Learnt to use the sequencer to schedule periodic tasks of varying priorities.
- Gained a better understanding of OpenCV and its APIs.
- Gained considerable experience in performing more complex image processing functions like contour detection.
- Gained better experience in synchronizing the tasks and communicating between the tasks using POSIX Message Queues
- Learnt that the response time of the system could be affected due to the physical limitations of your hardware, like the camera, leading to an IO bound system.

CHALLENGES FACED

- One of the major challenges we faced was to use OpenCV in a multi-threaded environment. It took a while to figure out how to display two windows (Finger Detection and UI) that were updated from two different threads.
- Extraction of ROI from the camera input required a lot of iterations.
- We had to improve the efficiency of the finger detection algorithm to reduce the computation time of the Finger Detection Task to meet the deadlines.

CONCLUSION

We were successfully able to implement the 'Touch Me Not' application, satisfying all the functional requirements. The system was schedulable and feasible, thereby satisfying its real-time requirements of providing response time of 1s.The system could be made more robust by using a faster camera, thus reducing computation of finger detection task. This will lead to capturing more data sets and resulting in much more accurate data. The project can be extended to provide an interactive UI as opposed to static images.

REFERENCES

[1] https://ideum.com/news/gesture-interaction-public-spaces-part1 [Touchless Gesture based Exhibits]
[2]https://www.concurrent-rt.com/wp-content/uploads/2016/09/The-RedHawk-Approach.pdf [Red Hawk Systems]
[3]https://becominghuman.ai/real-time-finger-detection-1e18fea0d1d4 [Real Time Finger detection]
[4] https://medium.com/@soffritti.pierfrancesco/handy-hands-detection-with-opencv-ac6e9fb3cec1 [Handy hand detection with OpenCV]s