

# ECEN 5623: REAL TIME EMBEDDED SYSTEMS

# TOUCH ME NOT

**Under the guidance of Professor Tim Scherr**

# 

Deepesh Sonigra

Madhumitha Tolakanahalli Pradeep

## CONTENTS

1. [INTRODUCTION](#_ve5fb16dne7o) 4
2. [PROBLEMS SOLVED BY THE TOOL](#_8ssduh1tt6gt) 4
3. [PRODUCT DESCRIPTION](#_nq6ydmlsq284) 4
4. [FUNCTIONAL CAPABILITIES](#_t1nhciwff922) 5
   1. [Hand Detection Capability](#_q3u2qhcee85m) 5
   2. [Fingers Detect Capability](#_l4wjhoulupi2) 5
   3. [Inter-thread Communication](#_gwhy1b4l2gnj) 5
   4. [QT Display](#_gwhy1b4l2gnj) 5
   5. [Tasks Synchronization](#_e8lanxta5stx) 5
   6. [Real-Time Response](#_gwhy1b4l2gnj) 5
5. [HARDWARE BLOCK DIAGRAM](#_gy4fxrjcd7j1) 6
6. [SOFTWARE BLOCK DIAGRAM](#_jsftu9w2lxiy) 7
7. [FLOWCHART](#_z56ousjozmzi) 8
8. [DATA FLOW DIAGRAM](#_17srxx4m7917) 9
9. [REAL-TIME SERVICE REQUIREMENTS](#_afh3ymasl1be) 10
10. [PERSONAL CONTRIBUTION](#_ratjksekq3bv) 11
    1. [DEEPESH SONIGRA](#_sp8l8f2ws9ja) 11
    2. [MADHUMITHA TOLAKANHALLI PRADEEP](#_1zw2h9ro9xqt) 11
11. [REFERENCES](#_ocpsqeenwuzc) 12

## 

## INTRODUCTION

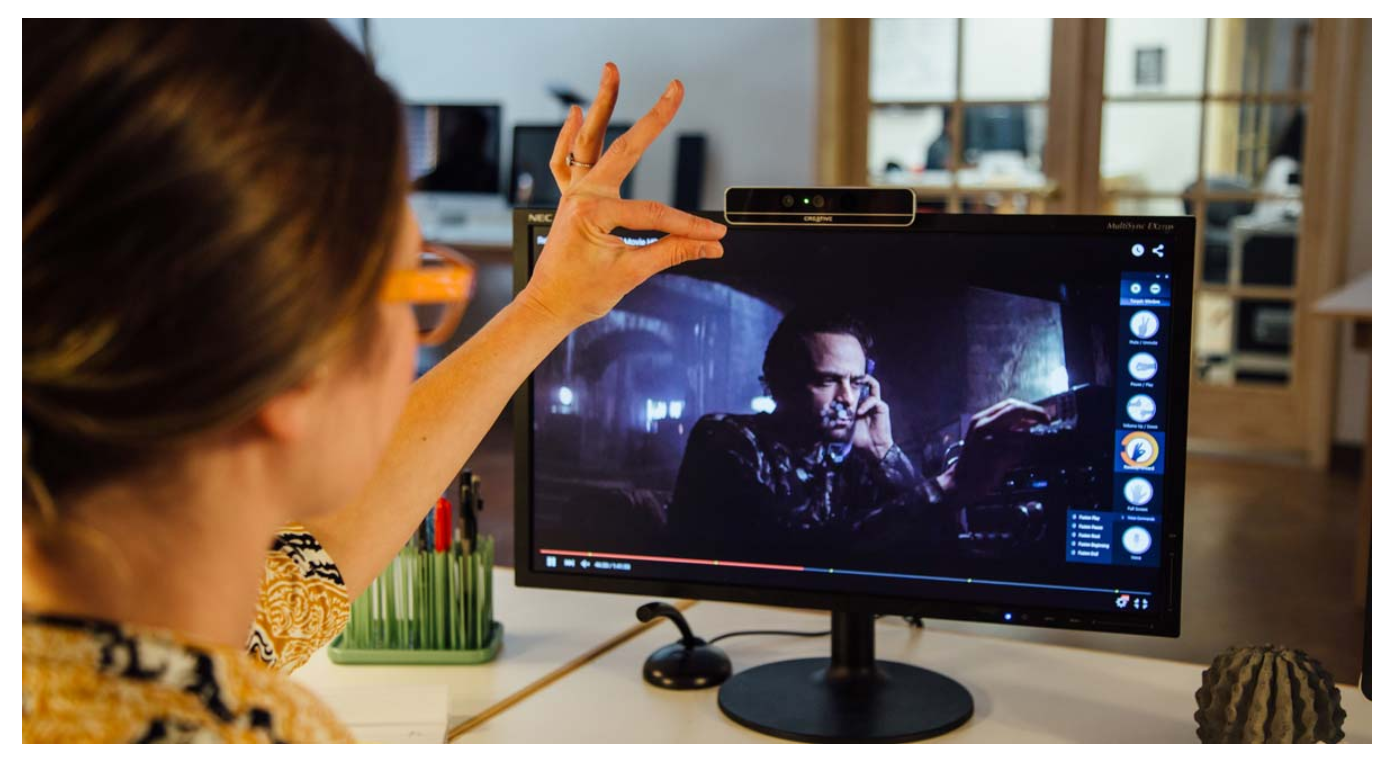
**“TOUCH ME NOT”** is a real time gesture detection tool which is capable of detecting and counting the raised fingers. This tool can be used to interact with any computer system executing options based on the number of fingers raised. The tool can be integrated in multiple applications incorporating 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 close 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 an 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 centre 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

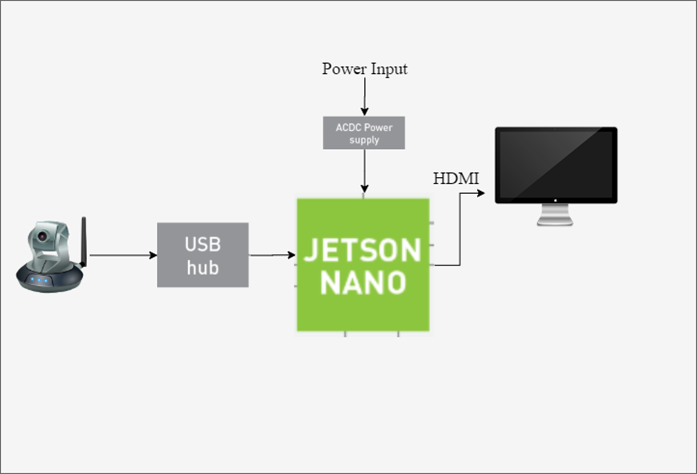
Using non-blocking message queues provides the user to extend the application for continuously changing UI, without blocking on the UI for user input. This capability extends the application of this tool into multiple disciplines.

### 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

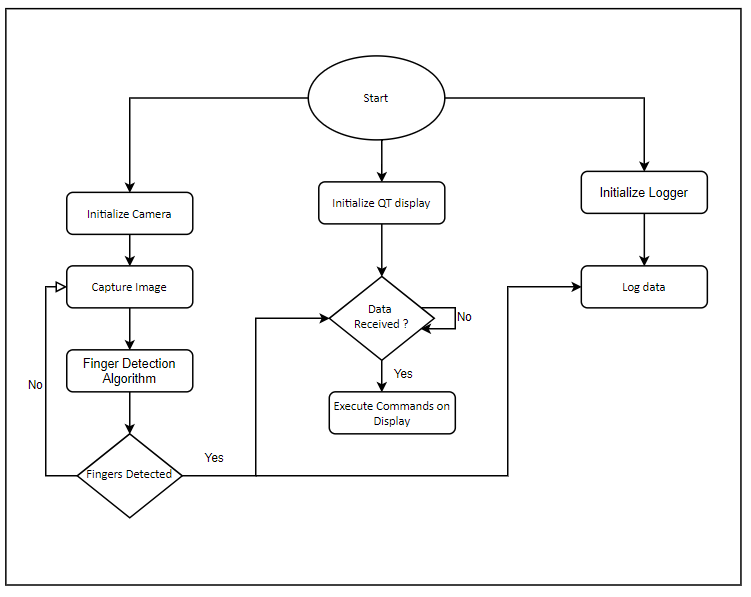
## SOFTWARE BLOCK DIAGRAM

### 

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.

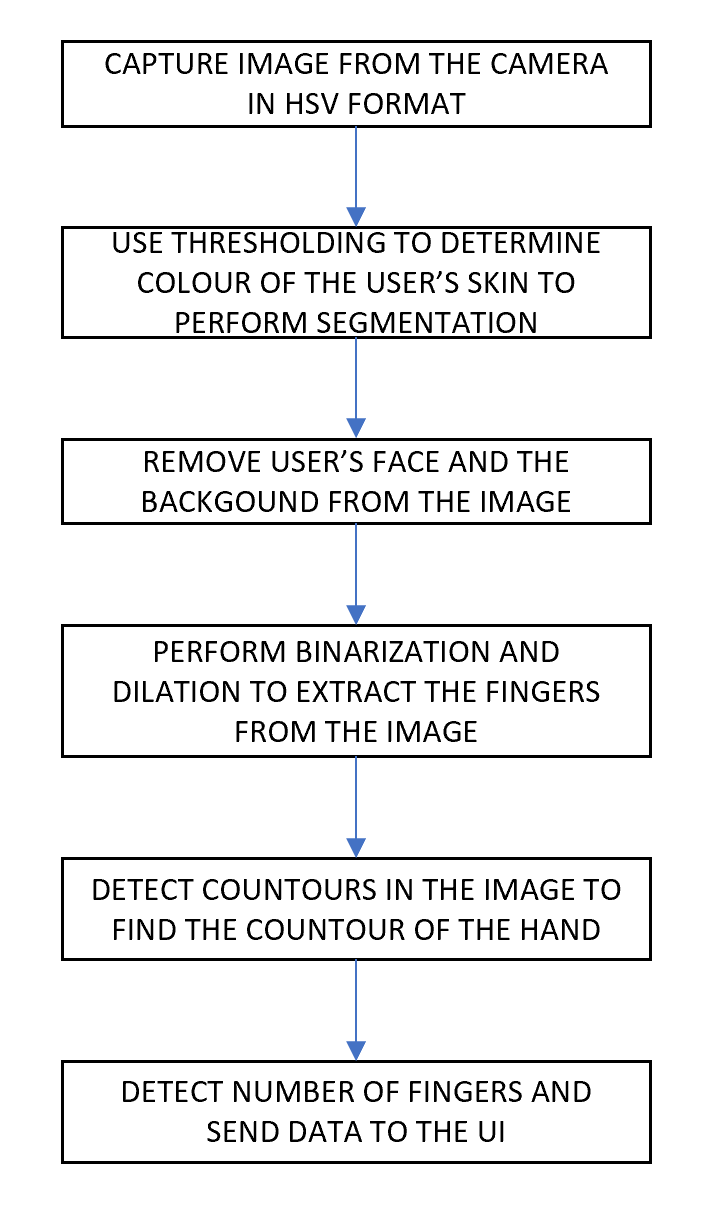
## FLOWCHART



* 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



* 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 in order 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

**TA**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 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 up.
* 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.
* 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 to process 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 a particular event or an exception occurs.

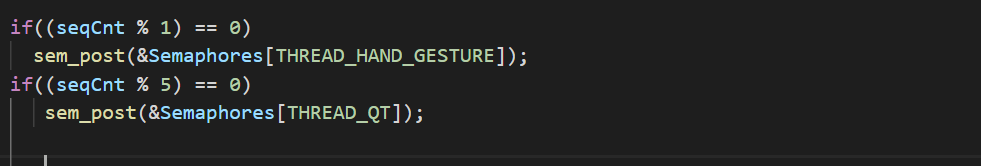
These log snippets confirm the worst execution time of these tasks.







### SEQUENCER CODE SNIPPET



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

### CHEDDAR DIAGRAM

## 

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

|  |  |  |  |
| --- | --- | --- | --- |
| 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 |
|  | | | 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.

## VERIFICATION AND VALIDATION

## 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

Connectivity Gigabit Ethernet, M.2 Key E

Display HDMI and DisplayPort

### IMPLEMENTATION OF KEY SERVICES

#### FINGER DETECTION TASK

* The finger detection task is scheduled by the Sequencer every 10Hz i.e every cycle of the sequencer and is run at a priority level, one less than the highest priority.

## PERSONAL CONTRIBUTION

#### DEEPESH SONIGRA

I will be responsible for the OpenCV setup on Jetson Nano, testing hand segmentation, background removal algorithm and counting fingers algorithm. I will be responsible for the binarization of the image and detecting the hand and fingers in the image. The execution time of these processes will be calculated and decisions would be made accordingly. The worst case execution time, period and deadline of the task will be calculated by running each task, one at a time. I will be implementing a logger task to log the system failures along with timestamps, providing debugging capability to the system. The LEDs integration will be done by me which works as an indication for various events

#### MADHUMITHA TOLAKANHALLI PRADEEP

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

## 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]

## 

## 

## 