Final Project: Synchronome

Saurav Negi

ECEN 5623 Real Time Embedded System

CU Boulder
May 6, 2024

1. Introduction

1.1. Synchronome

The Shortt–Synchronome free pendulum clock, developed in 1921 by British engineer William Hamilton Shortt and horologist Frank Hope-Jones, was a highly sophisticated electromechanical timepiece manufactured by the Synchronome Co., Ltd. of London, UK. Renowned for its unparalleled precision, it set the standard for timekeeping from the 1920s to the 1940s before being surpassed by quartz clocks. Widely utilized in observatories, research facilities, and national time services, Shortt clocks were the first to surpass the Earth's own timekeeping accuracy, even detecting slight seasonal variations in the Earth's rotation rate in 1926. With approximately 100 units produced between 1922 and 1956, Shortt clocks boasted accuracy to within a second per year, with recent assessments suggesting even greater precision.

These clocks maintained accuracy through a dual-pendulum system: a primary pendulum swung within a vacuum tank, while a secondary pendulum in a separate unit was synchronized to the primary via an electric circuit and electromagnets. The secondary pendulum, linked to the clock's timekeeping mechanism, ensured the primary pendulum remained undisturbed by external factors. This precision made Shortt clocks invaluable for synchronizing events.

In a related project, the aim is to synchronize camera image capture with the ticking of the clock's second hand at 1 Hz, while also capturing images at 10 Hz intervals for stopwatch purposes.

2. System Overview

2.1. Block Diagram

The following diagram provides a top-level overview of the end-to-end communication process, encompassing both hardware and software components, for capturing and storing images.

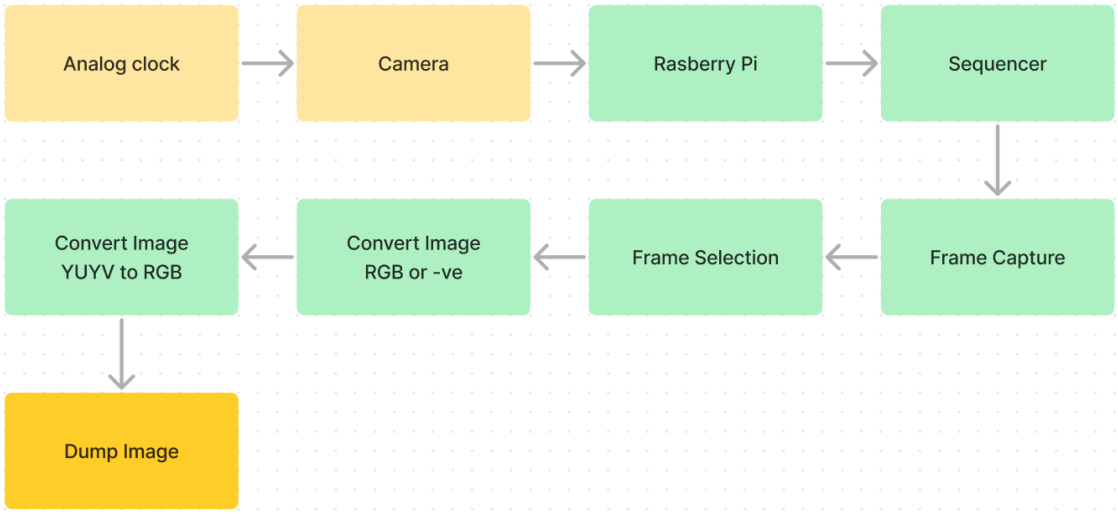
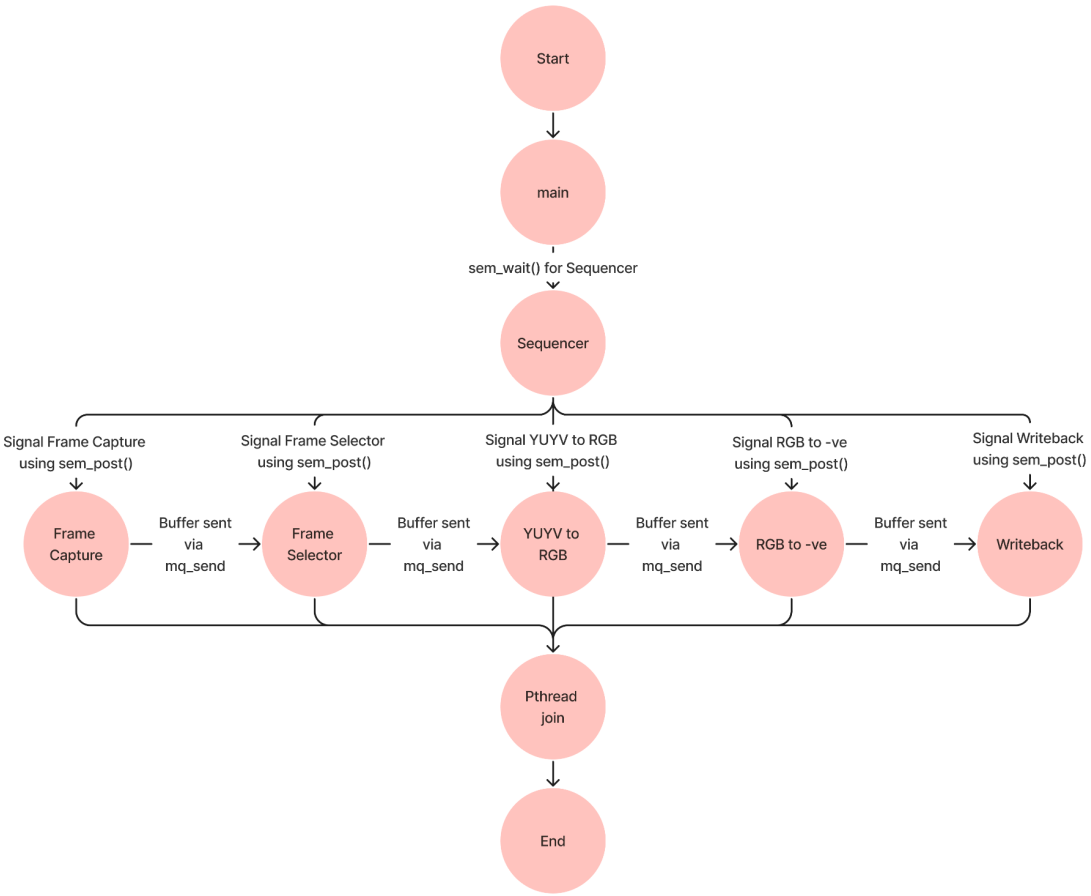


Fig: System Overview Hardware Diagram

The camera depicted in the aforementioned diagram captures data from the analog clock or stopwatch. This data, comprising frames, is transmitted to the Raspberry Pi via USB. Upon receiving the images, they undergo processing, including conversion from YUYV to RGB and subsequent application of a negative effect.

2.2. Software Architecture



The Sequencer is the main service running at a 100 Hz frequency and it dictates the frequency of all the other services. The Frame Capture service gathers frames from the camera and sends it to the next service via message queue.

Frame selector service selects every 10th frame out of the received frames in case of 1Hz & every 2nd frame out of the received frames in case of 10 Hz and then sends that to the next service via message queue.

YUYV to RGB conversion service performs the conversion at 1Hz frequency and 10 Hz frequency and then sends that to the next service via message queue.

RGB to -ve conversion service performs the conversion at 1Hz frequency and 10 Hz frequency and then sends that to the next service via message queue.

Writeback service saves frames at 1Hz frequency 10 Hz frequency respectively and then sends that to the next service via message queue.

2.3. Breakdown of CPU Cores

All real-time (RT) services are set to SCHED_FIFO priority.

2.3.1. Core 0

No threads are allocated to Core 0, allowing the operating system (OS) to schedule its services independently. This prevents interference with our designated services by avoiding task scheduling on our chosen cores.

2.3.2. Core 1

Both image capture and image selection services are assigned to Core 1 to run under Rate Monotonic Analysis (RMA).

Image capture operates at 10 Hz for 1Hz verification and 20 Hz for 10 Hz verification, while image selection runs at 10 Hz for 1 Hz verification to effectively extract message queue data for 1 Hz verification.

2.3.3. Core 2

Frame selection service operates at 20 Hz for 10 Hz verification.

YUYV to RGB conversion service operates at 1 Hz for 1Hz verification and 10 Hz for 10 Hz verification.

RGB to -ve conversion service operates at 1 Hz for 1Hz verification and 10 Hz for 10 Hz verification.

2.3.4. Core 3

To mitigate this impact on other RT services, writeback is designated as a best-effort service and executed separately on Core 3.

Frame saving service operates at 1 Hz for 1Hz verification and 10 Hz for 10 Hz verification.

Message queues facilitate communication of image data, timestamps, and frame numbers across services.

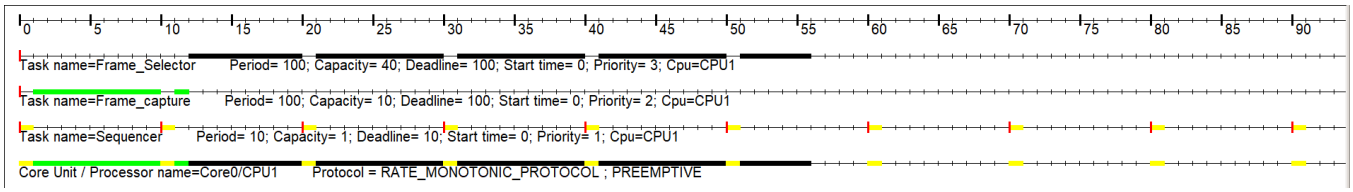
3. Analysis for 1 Hz

3.1. WCET & Capacity from hardware testing

Thread	Time Period/Deadline (T = D) (ms)	Execution Time (**WCET**)	Capacity (C) (ms)	Core	Service Priority
1 Sequencer	10	0.072851 ms	1	1	1
2 Frame Capture	100	7.218992 ms	10	1	2
3 Frame Selection	100	36.4788 ms	40	1	3
4 YUYV to RGB	1000	78.91108 ms	80	2	4
5 RGB to -ve	1000	17.966961 ms	20	2	5

3.2. Cheddar report for Core 1

Core 1 has 3 threads running on it namely Sequencer thread, Image capture thread & Frame selection thread.



Scheduling simulation, Processor CPU1 :
- Number of context switches : 5
- Number of preemptions : 1

- Task response time computed from simulation :
  Frame_capture => 17/worst
  Sequencer => 1/worst
  X-frame-diff => 6/worst
- No deadline missed in the computed scheduling : the task set is schedulable if you computed the scheduling on the feasibility interval.

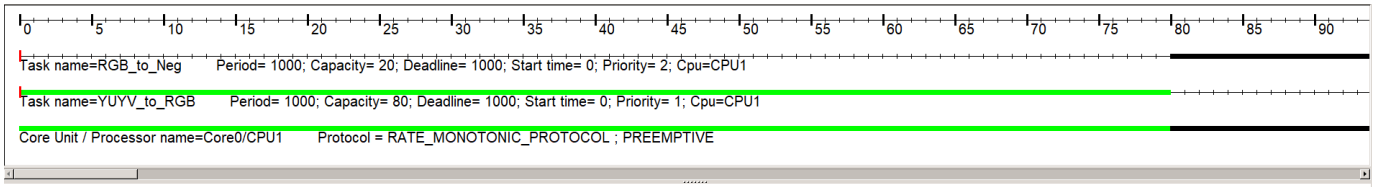
Scheduling simulation, Processor CPU1 :
- Number of context switches : 13
- Number of preemptions : 5

- Task response time computed from simulation :
  Frame_Selector => 56/worst
  Frame_capture => 12/worst
  Sequencer => 1/worst
- No deadline missed in the computed scheduling : the task set is schedulable if you computed the scheduling on the feasibility interval.

It clearly shows that there is no possibility that the deadlines will be missed.

3.3. Cheddar report for Core 2

Core 2 has 2 threads running on it namely yuyv to rgb conversion thread, rgb to -ve conversion thread.



Scheduling simulation, Processor CPU1 :
- Number of context switches : 13
- Number of preemptions : 5

- Task response time computed from simulation :
  Frame_Selector => 56/worst
  Frame_capture => 12/worst
  Sequencer => 1/worst
- No deadline missed in the computed scheduling : the task set is schedulable if you computed the scheduling on the feasibility interval.

Scheduling simulation, Processor CPU1 :
- Number of context switches : 1
- Number of preemptions : 0

- Task response time computed from simulation :
  RGB_to_Neg => 100/worst
  YUYV_to_RGB => 80/worst
- No deadline missed in the computed scheduling : the task set is schedulable if you computed the scheduling on the feasibility interval.

It clearly shows that there is no possibility that the deadlines will be missed.

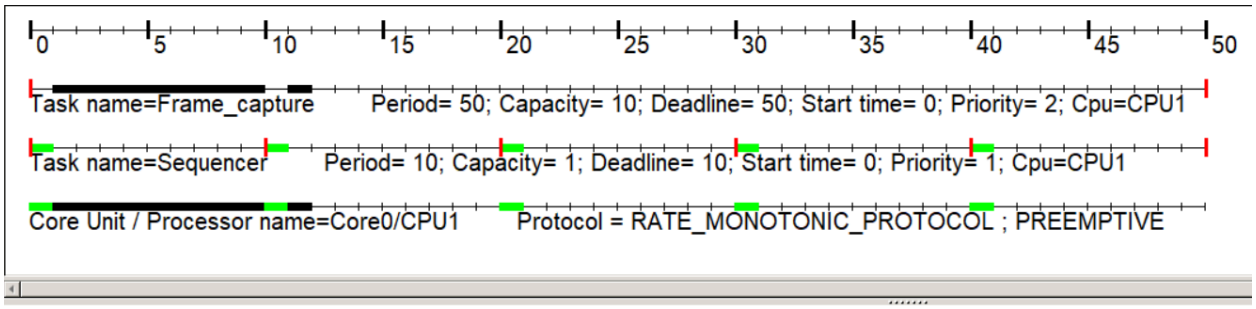
4. Analysis for 10 Hz

4.1. WCET & Capacity from hardware testing

Thread	Time Period/Deadline (T = D) (ms)	Execution Time (**WCET**)	Capacity (C) (ms)	Core	Service Priority
1 Sequencer	10	0.082629 ms	1	1	1
2 Frame Capture	50	7.69654 ms	10	1	2
3 Frame Selection	50	36.8678 ms	40	2	3
4 YUYV to RGB	100	70.84371 ms	80	2	4
5 RGB to -ve	100	15.245727 ms	20	2	5

4.2. Cheddar report for Core 1

Core 1 has 2 threads running on it namely Sequencer thread, Image capture thread.



Scheduling simulation, Processor CPU1 :
- Number of context switches : 13
- Number of preemptions : 5

- Task response time computed from simulation :
  Frame_Selector => 56/worst
  Frame_capture => 12/worst
  Sequencer => 1/worst
- No deadline missed in the computed scheduling : the task set is schedulable if you computed the scheduling on the feasibility interval.

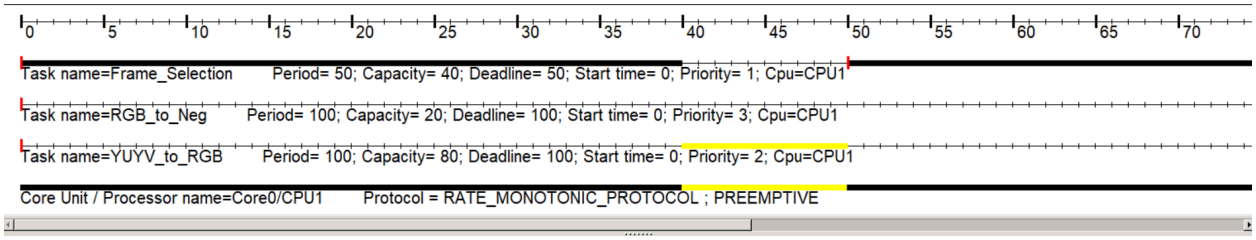
Scheduling simulation, Processor CPU1 :
- Number of context switches : 4
- Number of preemptions : 1

- Task response time computed from simulation :
  Frame_capture => 12/worst
  Sequencer => 1/worst
- No deadline missed in the computed scheduling : the task set is schedulable if you computed the scheduling on the feasibility interval.

It clearly shows that there is no possibility that the deadlines will be missed.

4.3. Cheddar report for Core 2

Core 2 has 3 threads running on it namely Frame selection, yuyv to rgb conversion thread & rgb to -ve conversion thread.



Scheduling simulation, Processor CPU1 :
· Number of context switches : 1
· Number of preemptions : 0

· Task response time computed from simulation :
RGB_to_Neg => 100/worst
YUYV_to_RGB => 80/worst
· No deadline missed in the computed scheduling : the task set is schedulable if you computed the scheduling on the feasibility interval.

Scheduling simulation, Processor CPU1 :
· Number of context switches : 3
· Number of preemptions : 1

· Task response time computed from simulation :
Frame_Selection => 40/worst
RGB_to_Neg => 0/worst , response time not computed since the task did not run all its capacity
YUYV_to_RGB => 0/worst , response time not computed since the task did not run all its capacity
· One or several tasks did not complete their execution.

It clearly shows that there is no possibility that the deadlines will be missed.

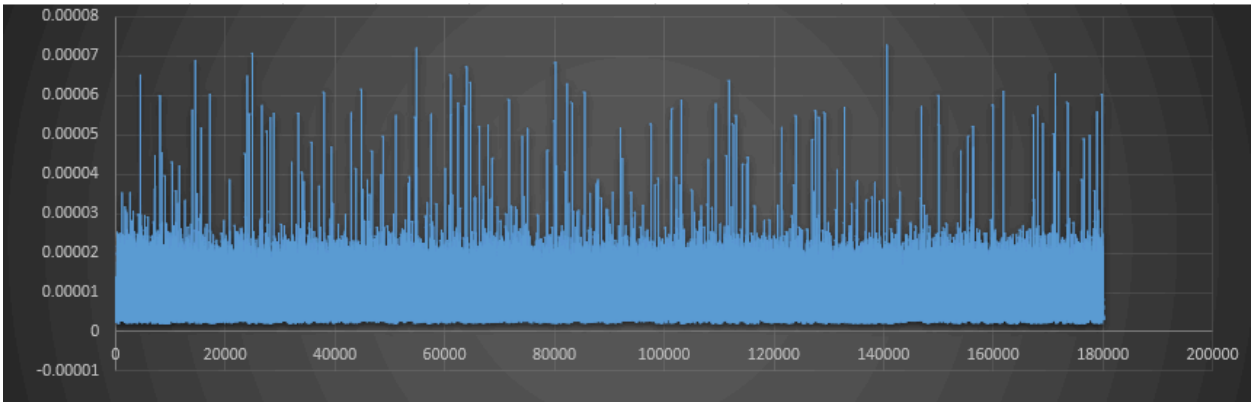
5. Results from practical testing on HW (1Hz)

5.1. Sequencer Service

This function is triggered at a 100 Hz frequency by the kernel timer using a Signal and timer_create combination. Here, we set the value of the trigger frequency by modifying it_interval and it_value parameters. This function is also used to disable all the RT services when an abort request is received.

This particular service oversees the operation of all real-time services distributed across various cores of the system.

The Frequency of this service is 100 Hz and it runs on Core 1.

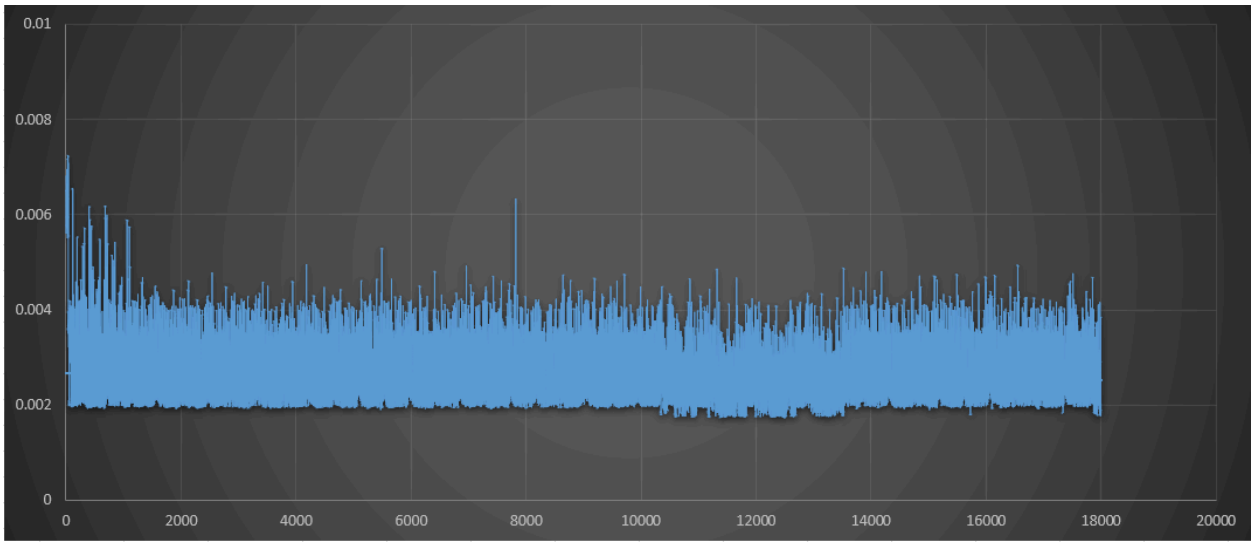


Thread-1 : Jitter Analysis

5.2. Frame Capture Service

This service is responsible for capturing frames from the camera. It acquires frames at 10 Hz for 1 Hz verification and at 20 Hz for 10 Hz. The service involves making driver calls over USB to the UVC camera to obtain the frames.

It runs on core 1. The service is called at frequencies of 10 Hz for 1 Hz verification.

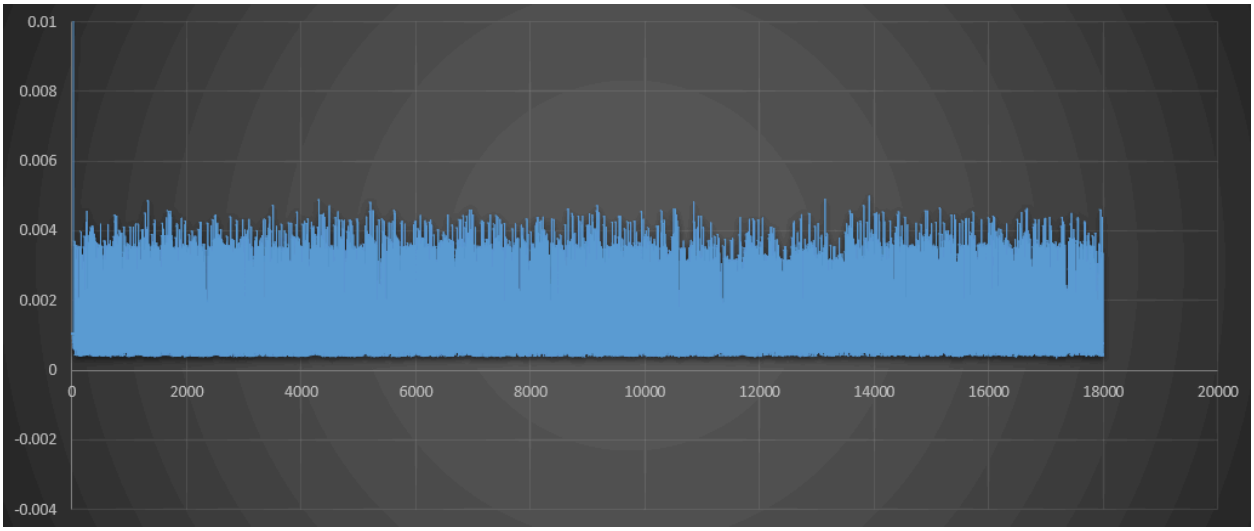


Thread-2 : Jitter Analysis

5.3. Frame Selection Service

This service aims to choose the appropriate images for forwarding to YUYV to RGB or YUYV to Negative processing. For 1 Hz operations, an oversampling mechanism is employed using a shotgun approach. For 10 Hz operations, a selection logic is applied.

It runs on core 1. The service is called at frequencies of 10 Hz for 1 Hz verification.

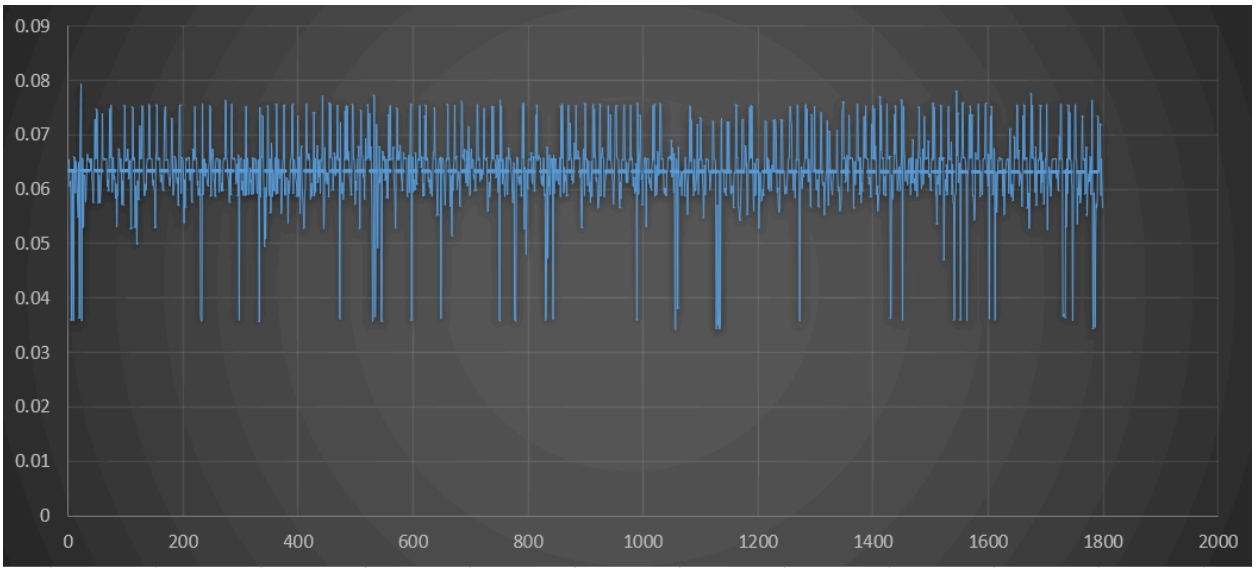


Thread-3 : Jitter Analysis

5.4. YUYV to RGB Conversion Service

This service is designed to handle the frames received from Service_X_frame_diff. It operates at a frequency of 1 Hz for 1 Hz verification and 10 Hz for 10 Hz verification. Its primary task is to convert the previously selected YUYV image to RGB format.

It runs on core 2. The service is called at frequencies of 1 Hz for 1 Hz verification.

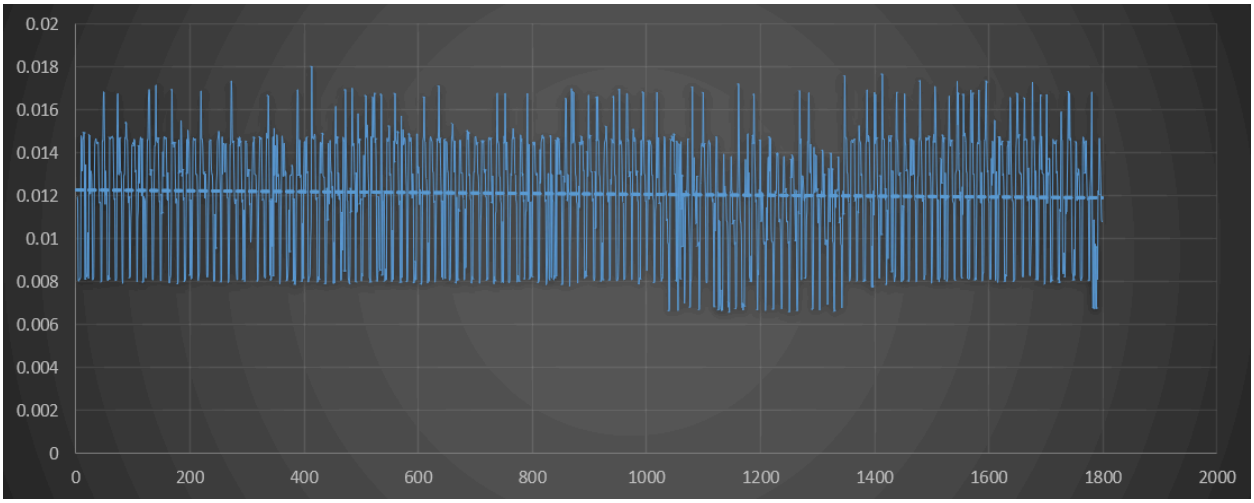


Thread-4 : Jitter Analysis

5.5. RGB to -ve Conversion Service

This service, operating with a priority just below the highest real-time priority, alters an RGB image to a negative version upon user request. If the user specifies a preference for a colored image, either permanently or during runtime, the output remains in RGB format.

It runs on core 2. The service is called at frequencies of 1 Hz for 1 Hz verification.



Thread-5 : Jitter Analysis

5.6. Writeback Dump

This service is responsible for saving the ppm images received from transform image service. It operates as a best-effort service since verification and analysis occur post-processing and do not require real-time priority. As dumping is the slowest operation among all services, it is isolated to a separate core to avoid potential interference with other real-time services.

This service runs on core 3.

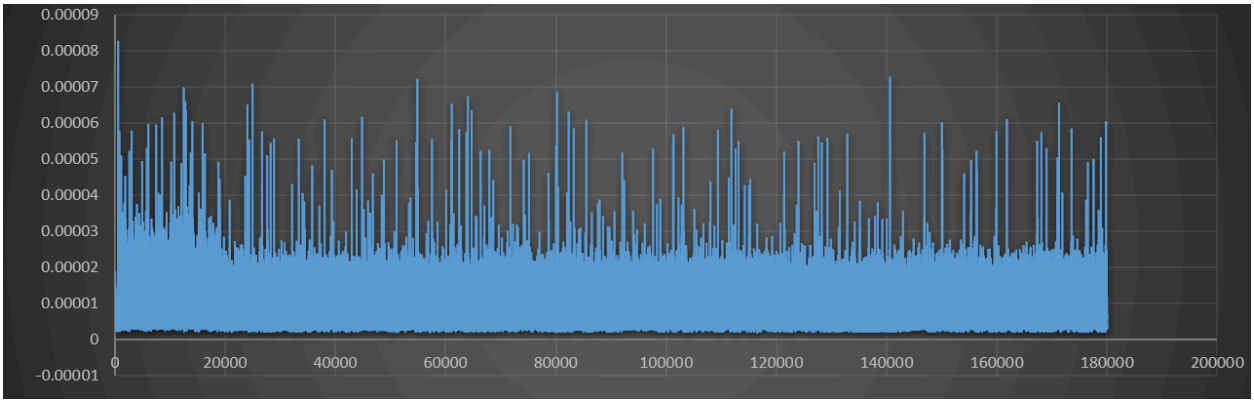
6. Results from practical testing on HW (10Hz)

6.1. Sequencer Service

This function is triggered at a 100 Hz frequency by the kernel timer using a Signal and timer_create combination. Here, we set the value of the trigger frequency by modifying it_interval and it_value parameters. This function is also used to disable all the RT services when an abort request is received.

This particular service oversees the operation of all real-time services distributed across various cores of the system.

The Frequency of this service is 100 Hz and it runs on Core 1.

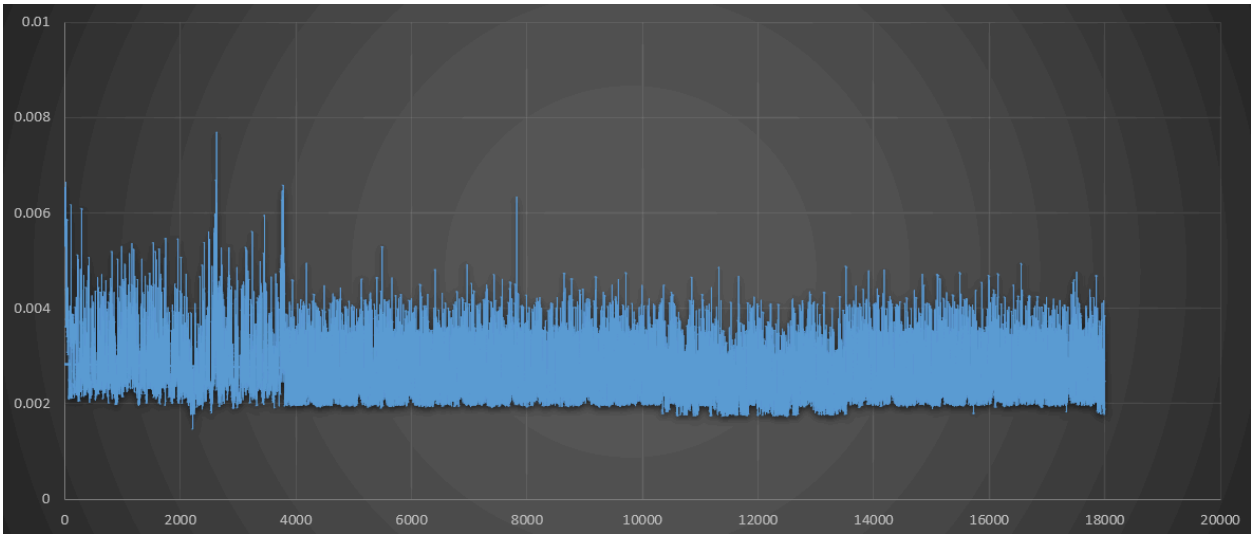


Thread-1 : Jitter Analysis

6.2. Frame Capture Service

This service is responsible for capturing frames from the camera. It acquires frames at 10 Hz for 1 Hz verification and at 20 Hz for 10 Hz. The service involves making driver calls over USB to the UVC camera to obtain the frames.

It runs on core 1. The service is called at frequencies of 20 Hz for 10 Hz verification.

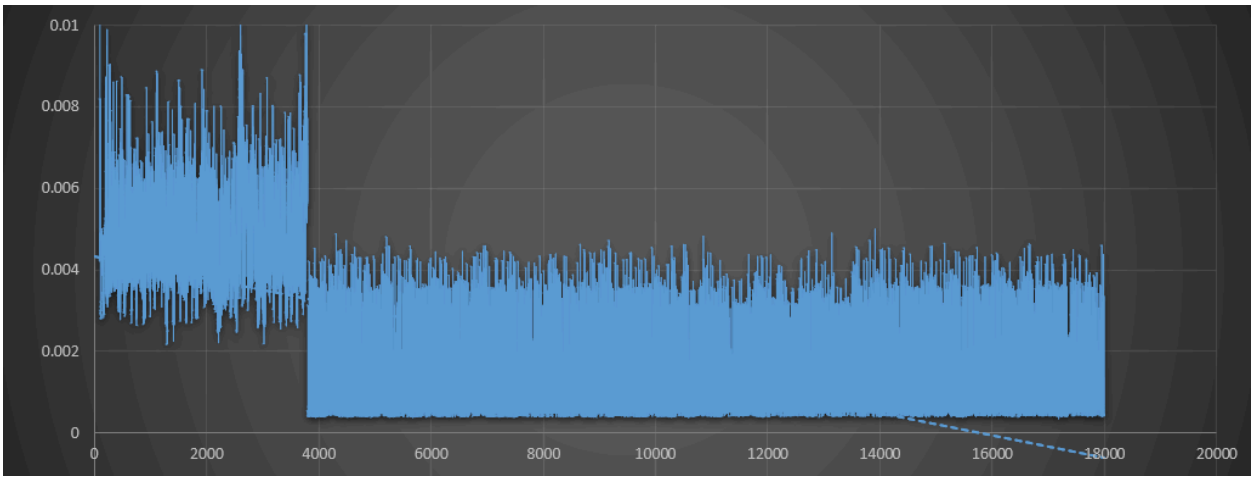


Thread-2 : Jitter Analysis

6.3. Frame Selection Service

This service aims to choose the appropriate images for forwarding to YUYV to RGB or YUYV to Negative processing. For 1 Hz operations, an oversampling mechanism is employed using a shotgun approach. For 10 Hz operations, a selection logic is applied.

It runs on core 2. The service is called at frequencies of 20 Hz for 10 Hz verification.

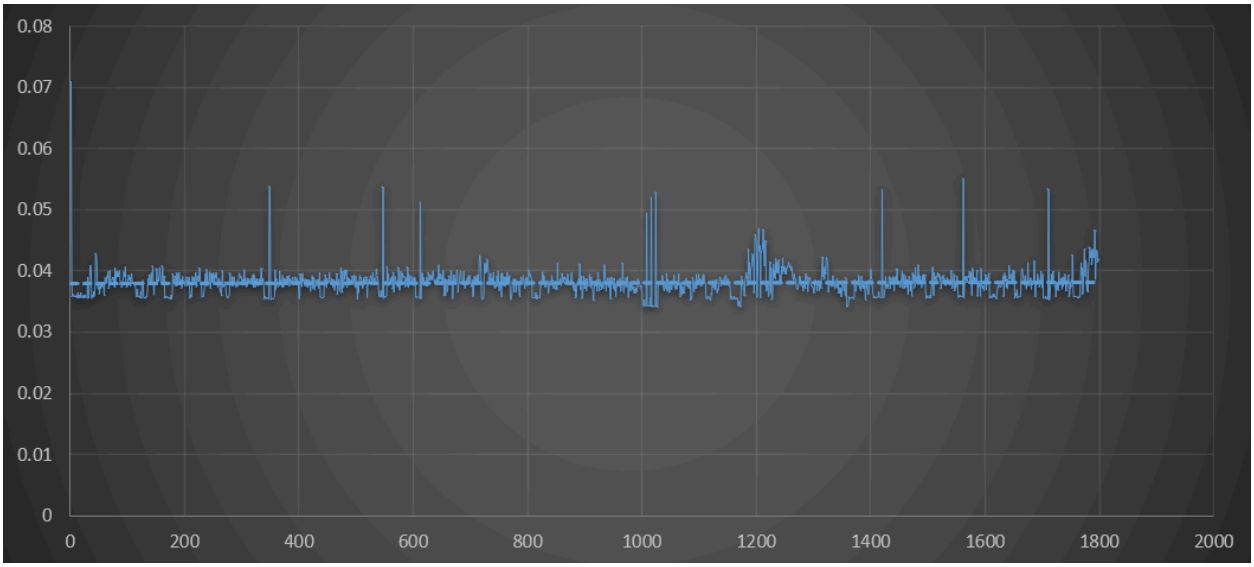


Thread-3 : Jitter Analysis

6.4. YUYV to RGB Conversion Service

This service is designed to handle the frames received from Service_X_frame_diff. It operates at a frequency of 1 Hz for 1 Hz verification and 10 Hz for 10 Hz verification. Its primary task is to convert the previously selected YUYV image to RGB format.

It runs on core 2. The service is called at frequencies of 10 Hz for 10 Hz verification.

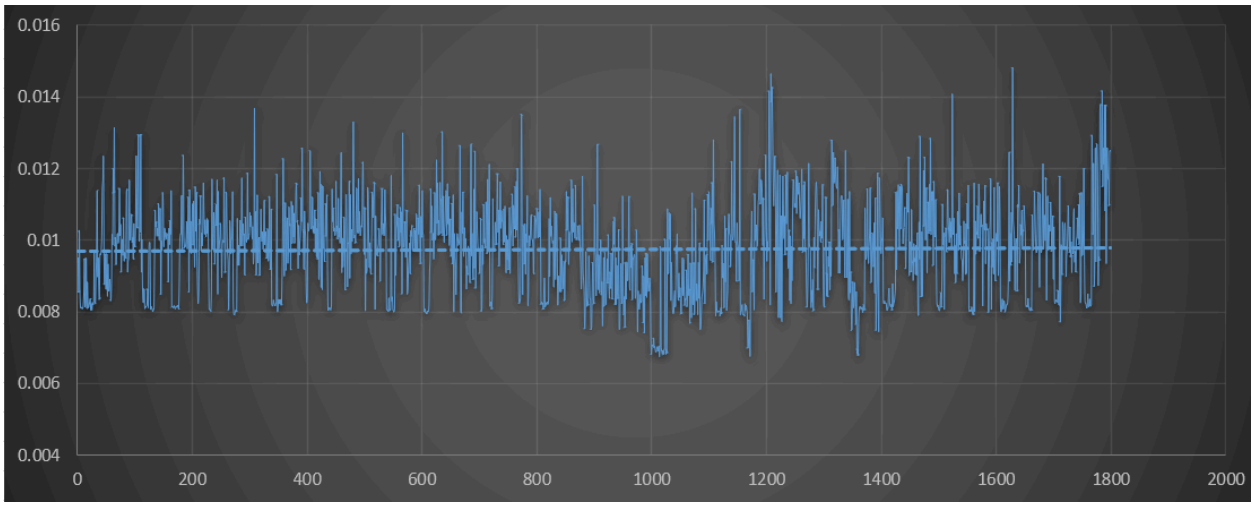


Thread-4 : Jitter Analysis

6.5. RGB to -ve Conversion Service

This service, operating with a priority just below the highest real-time priority, alters an RGB image to a negative version upon user request. If the user specifies a preference for a colored image, either permanently or during runtime, the output remains in RGB format.

It runs on core 2. The service is called at frequencies of 10 Hz for 10 Hz verification.



Thread-5 : Jitter Analysis

6.6. Writeback Dump

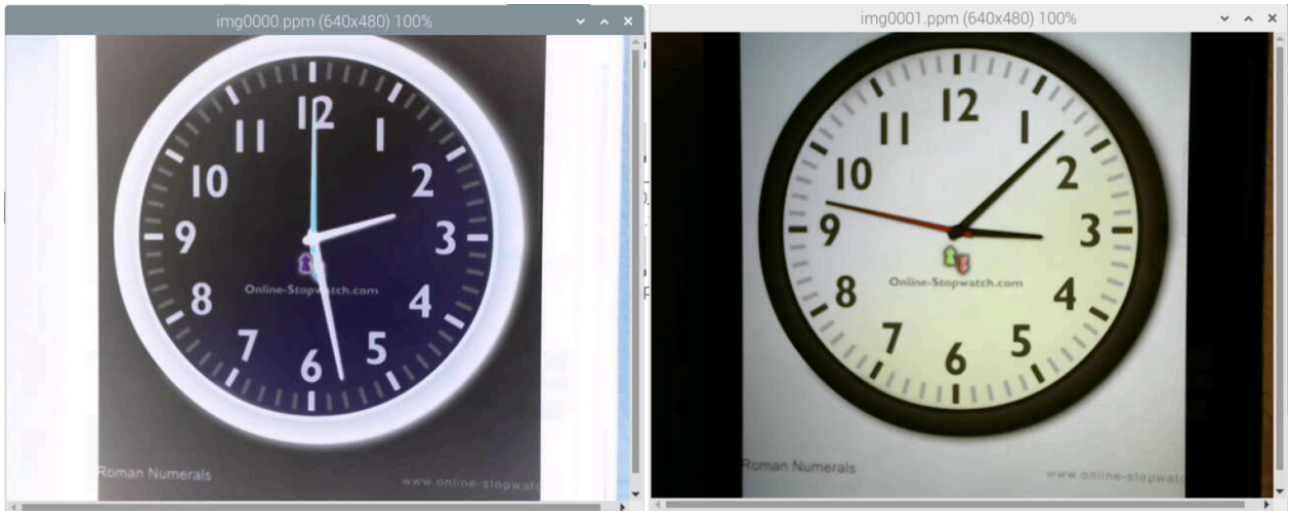
This service is responsible for saving the ppm images received from transform image service. It operates as a best-effort service since verification and analysis occur post-processing and do not require real-time priority. As dumping is the slowest operation among all services, it is isolated to a separate core to avoid potential interference with other real-time services.

This service runs on core 3.

7. Proof of concept

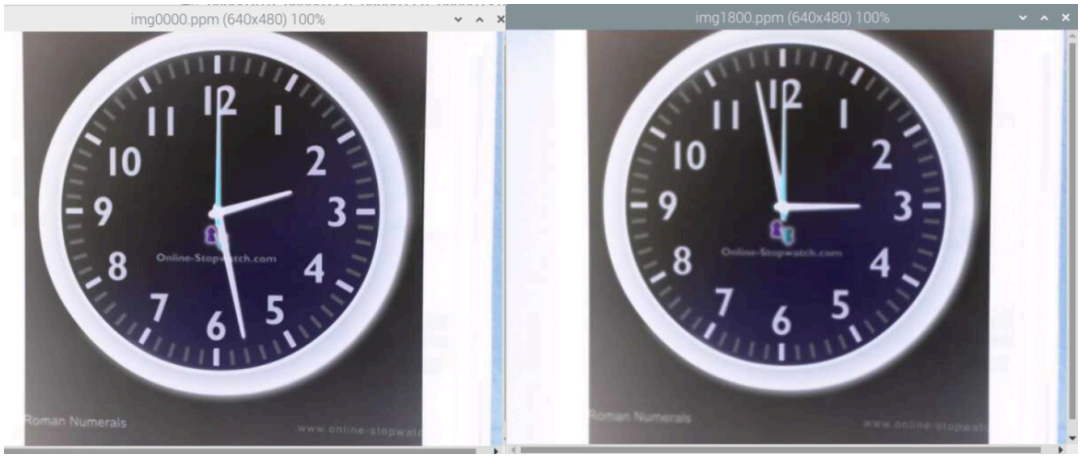
7.1. Additional feature enabled and disabled showcase

Below image shows that system allows additional feature enabled and disabled options to the user.



7.2. Test results for 1Hz, 1800 frames

Initial and last frames below for the case of 1800 frames:



Above difference in time shows that the two images are exactly 30 minutes apart which was intended from this test scenario. There were no blurred images, skipped frames observed while extensive testing.

7.3. Terminal behavior of the system

Below terminal image shows the working of the system.

```
================================================================
      Welcome to Synchronome Demo
================================================================


================================================
      Select Frequency
================================================
1. 1  Hz
2. 10 Hz
================================================
2
================================================


================================================
      Additional feature enable/disable ?
================================================
1. disable
2. enable
================================================
1
================================================


================================================
      Select Frames count
================================================
1. 180  frames
2. 1800 frames
================================================
1
================================================

================================================================


Synchronome Project code Start

Camera Opened
Camera Initialization Successful

Thread 1 - Sequencer Created
Thread 2 - Image Capture Created
Thread 3 - Image Capture Created
Thread 4 - YuYV to RGB Created
Thread 5 - RGB to -ve if selected Created
Thread 6 - Image dumping thread created

Initialize Sequencer timer

Total frames saved = 181

Camera Closed

Synchronome Project code End
```

8. Conclusions

Embarking on the Synchronome project has been an enlightening journey, filled with learning and personal growth. This experience has provided me with valuable insights into managing services in real-time environments. Through hands-on work, I've learned to appreciate the importance of small timing differences—whether measured in milliseconds or microseconds—that can lead to system failures or unexpected behaviors.

This project has allowed me to dive deep into fundamental concepts, particularly emphasizing the crucial role of precise scheduling. I've focused on understanding the Rate Monotonic policy, recognizing its significance in ensuring the precise and predictable execution of real-time tasks. Applying theoretical knowledge to practical scenarios has strengthened my grasp of real-time system dynamics, bridging the gap between theory and real-world outcomes.

9. References

1. Professor Sam Siewert's example codes.
2. Real time embedded components & systems with linux & rtos.
3. https://ieeexplore.ieee.org/document/9239814
4. https://ieeexplore.ieee.org/document/5376455