ECEN-5623
PROJECT REPORT

VISUAL
SYNCHROME

ISHA
SHARMA

Date: 08/13/2023

Introduction

This project is a system that syncs with an external analog clock. The system acquires time-lapse images using a VGA resolution camera at 1Hz and partially at 10Hz and further verification is done. The system is implemented in real-time using services via Rate Monotonic Theory. The system captures real-time images, filters them, transforms them, and saves them in PPM format. All the services run on the same core (Core 0) .

Functional Requirements:

A. Minimum Requirements

1. 2 or more real-time services running on one AMP core
2. Capturing of images at 1hz (180+1 frames) without any blur /skip or repeat
3. The resolution of the images will be VGA (640x480)
4. Individual images will be stored in PPM/PGM format with the timestamp mentioned in the image header

B. Target goals

1. Capture 1800+1 stable frames at 1/10Hz and save them (some glitches acceptable)
2. Transformation feature enable/disable

C. Stretch Goals

1. Run at 10 Hz, non-blurry, unique, monotonic to 1/10th of second, 2x or more for 1800+1

Verification of Functional Requirements:

The images captured were compared with an external wall clock over a duration of 3mins with 181 frames for 1Hz mode of capture.

The execution times for each service were measured using syslog and framer jitter was calculated. Feasibility tests as well as RM analysis was done using Cheddar.

The frames were verified by stepping through each one of them to find glitches and if found, analysing why they were present.

Real-Time requirements

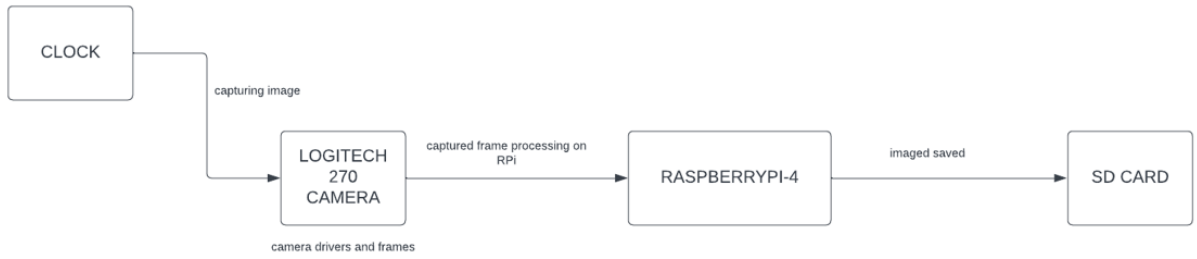
1. **Scheduler**
   This task runs on CPU-0 and is responsible for releasing semaphores and triggering other services. It ran at 33.33ms, ie, 30hz.

2. **Read Frame**
   This task runs on CPU-0 and is responsible for acquires camera images at fixed intervals.
   It runs at 1 Hz,
   a. C = 0.213 us
   b. T = D = 1ms
   The WCET has been calculated by comparing all the recorded execution times and finding the largest number. These frames are captured and stored in a buffer.

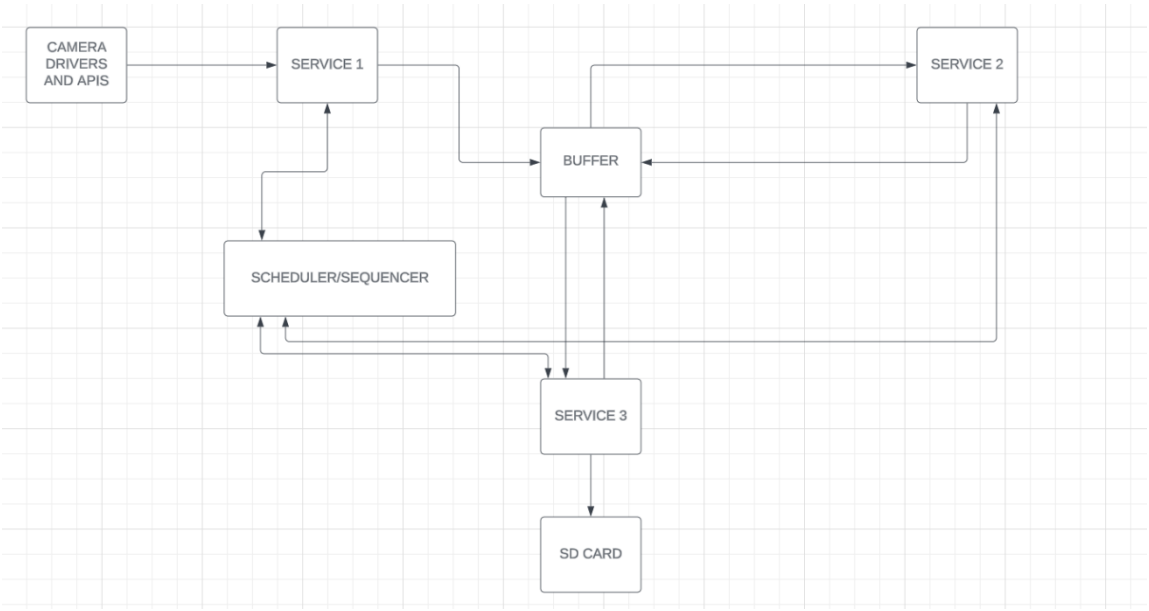
3. **Transforming and processing the frames.**
   This task runs on CPU-0. I am using the brightening transformation on the images. The image is read from the buffer at a fixed interval. It is then transformed into a new image and saved in a same buffer
   a. It runs at 3 Hz,
   b. C = 4.683us
   c. T = D = 0.333ms.

4. **Dump PGM**
   This task runs on CPU-0 and stores the images from the buffer in ppm format along with the header timestamps/frame numbers properly. It saves the frames in a folder called 'frames'.
   a. It runs at 1 Hz,
   b. C = 4.9us
   c. T = D =1ms

Block Diagrams



Shown above is the hardware block diagram. An external analog clock is monitored using a Logitech @70 camera. This camera is connected to the raspberry pi which performs the necessary functions and saves the images captured onto SD card.

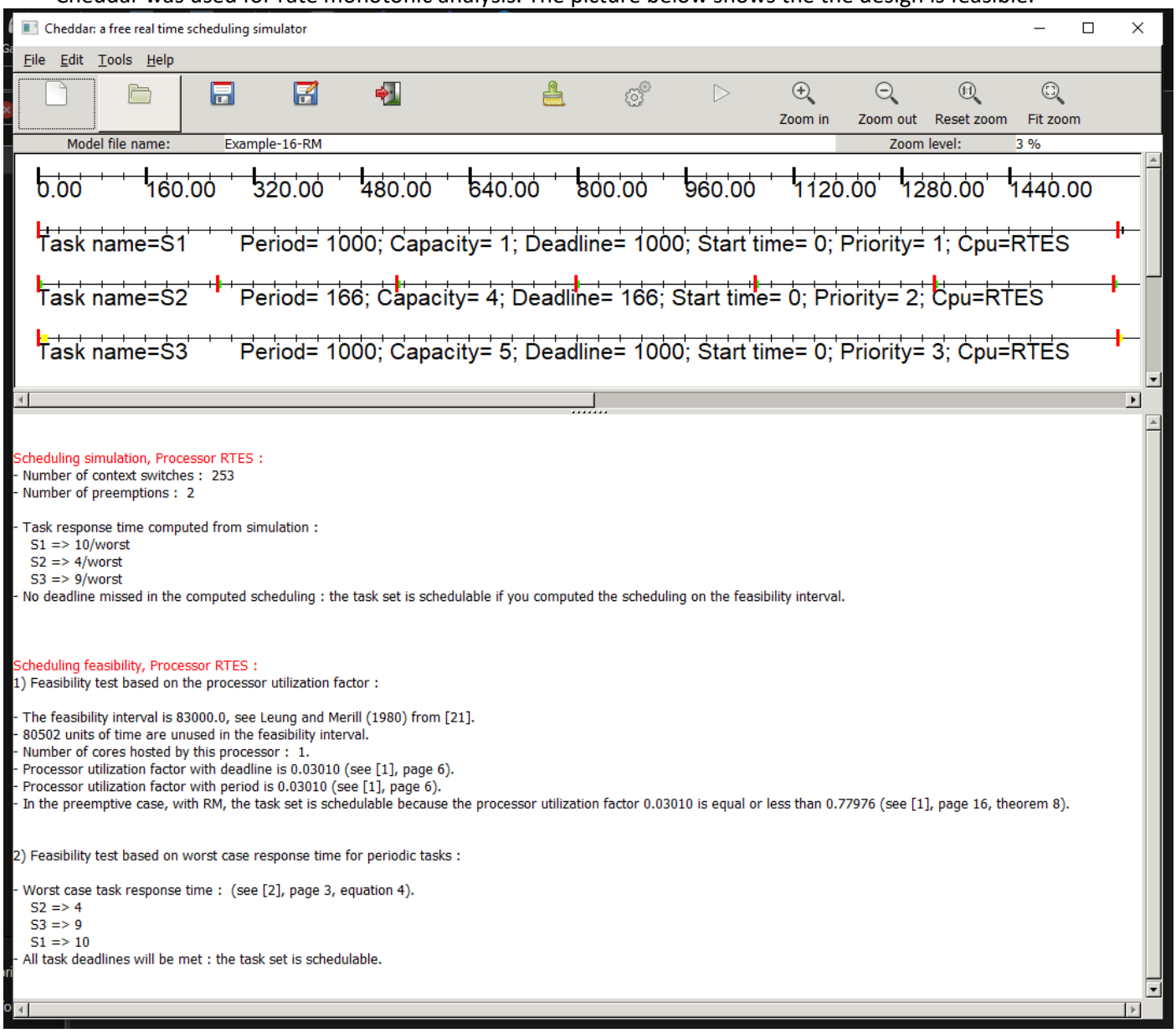


Shown above is the software block diagram. The service 1, read frame, captures the frames of the camera and stores them onto a buffer. This frame is read by the service 2, which processes the image and brightens it based on enable/disable. It does so by applying a simple linear transformation to each pixel's intensity value. The image is assumed to have dimensions of 480 rows, 640 columns, and 3 color channels (red, green, and blue). The function iterates through each pixel in the image and for each pixel, it applies the transformation using scaling factor and offset value. It then saves the frame onto the buffer. This is read by the service 3, which saves the frame onto sd card according to the naming convention and with the header information. All of the 3 services are triggered by the sequencer.

Analysis

1) Rate monotonic:

Cheddar was used for rate monotonic analysis. The picture below shows the the design is feasible.



2) Feasibility tests:

The scheduling point test and completion point test were run and the design successfully passed them.

```
Ex-1 U=3.33% (C1=0.213, C2=4.683, C3=4.9; T1=1000, T2=166, T3=1000; T=D): FEASIBLE
for 3, utility_sum = 0.000000
for 0, wcet=1.000000, period=1000.000000, utility_sum = 0.001000
for 1, wcet=4.000000, period=166.000000, utility_sum = 0.025096
for 2, wcet=4.000000, period=1000.000000, utility_sum = 0.029096
utility_sum = 0.029096
LUB = 0.779763
RM LUB FEASIBLE
```

The above image is for scheduling point test.

```
Ex-1 U=3.33% (C1=0.213, C2=4.683, C3=4.9; T1=1000, T2=166, T3=1000; T=D): FEASIBLE
for 3, utility_sum = 0.000000
for 0, wcet=1.000000, period=1000.000000, utility_sum = 0.001000
for 1, wcet=4.000000, period=166.000000, utility_sum = 0.025096
for 2, wcet=4.000000, period=1000.000000, utility_sum = 0.029096
utility_sum = 0.029096
LUB = 0.779763
RM LUB FEASIBLE
```

The above image is for completion point test.
Hence, it is clarified that the assumed values run successfully and safely.

3) Safety Margin

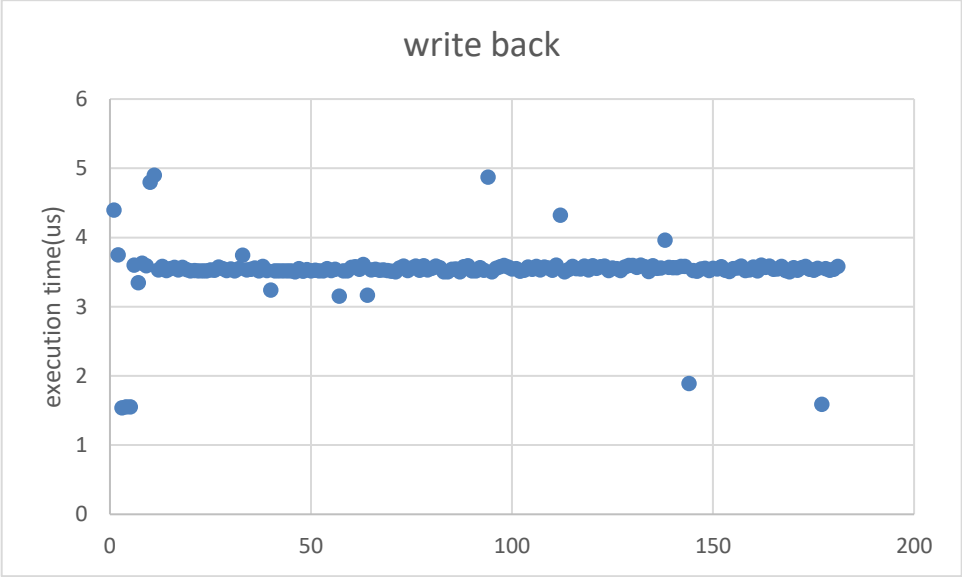
The time taken for Read Frame, Transformation and writeback for worst case is 10 us. The deadlinegiven in 100ms for 10Hz and 1s for 1Hz. This shows that the design has enough safety margin.

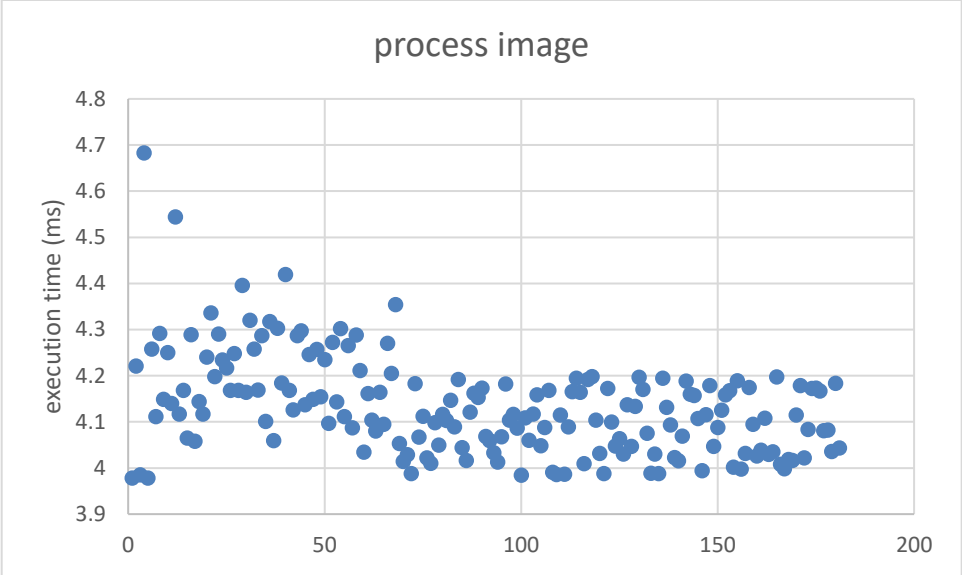
Feasibility Margin

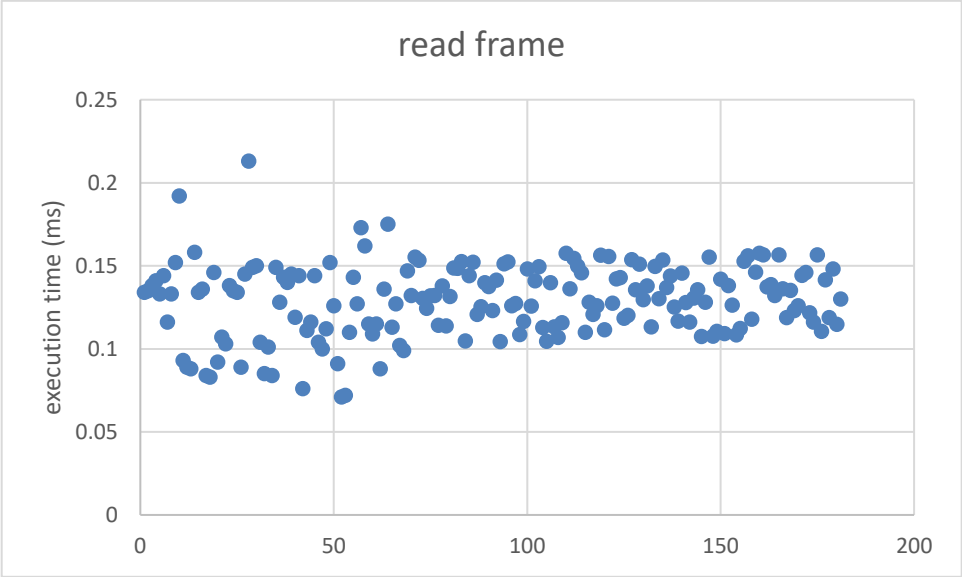
The best case execution time – worst case execution times are given below for each service:
1) Read frame: 0.213 – 0.017 = 0.196
2) Transform: 4.683 – 3.978 = 0.705
3) Write back: 4.9 – 1.54 = 2.96

Therefore, there is enough error margin for feasibility.

4) Frame Jitter







Above images are the jitter analysis for each service. It was seen that writeback had the least amount of jitters.

Conclusion

This project was a complete bundle or learning. From time management to technical real time skills, I leant quite a lot. Talking about the project, some goals were met and others were partially met. The main reasons that I could identify were:

1) There were sometimes glitches seen in the frames. This was deduced to be present because I was running all the services on core 0. Given that Kernel processes run on core 0, it affects the processes and their timings.
2) The transformation process whenever turned on used to brighten some frames completely but 40% of the frames were glitchy. Glitch in the sense, there would be some portions of the frames left un-brightened. This, I believe, happens due to inversion.
3) When the 10hz code were run, the first 5 frames out of the 1801 were blank. But comparing the 6^{th} frame to the last frame, the time difference was 3 mins. This is due to oversampling. Since no specific frame selection method was added to the code, there were multiple frames for the same time.

Other than these mentioned causes for errors, the other requirements were met.

References

https://www.sciencedirect.com/science/article/abs/pii/027510629390048T
Rate-Monotonic-Theory-Liu-and-Layland.pdf
Priority-Inheritance-Protocols.pdf
ECEN 5623 Textbook
https://neptune.ai/blog/image-processing-techniques-you-can-use-in-machine-learning
https://github.com/siewertsmooc/RTES-ECEE-5623/tree/main/Std-Project-Starter-Code

Professor Sam and all the TA's