Project Journal

Machine Vision-Based Underwater Anti-Backscatter Lighting System.

Sidharth Shanmugam.

Table of Contents

[W/B 30-10-2023 3](#_Toc150450628)

[Initial Supervision Meeting & High-Level Objectives 3](#_Toc150450629)

[W/B 06-11-2023 4](#_Toc150450630)

[Intro to Real-Time Software [1] 4](#_Toc150450631)

[Supervision Meeting 4](#_Toc150450632)

[Bibliography 6](#_Toc150450633)

# W/B 30-10-2023

## Initial Supervision Meeting & High-Level Objectives

* Meeting with Paul Mitchell and Benjamin Henson on Tuesday.
* Discussed background for original project idea and overall goals to achieve.
* Discussed previous work carried out by Katie, limitations, and scope for future work.
  + Improving real-time computing.
    - Experimentation with more powerful processing hardware.
    - Research on hypervisors, threading, and multiprocessing.
    - Using IR beams to detect backscatter, potentially eliminating need for high/low beam flash loops.
    - Streaming video from Pi to a more powerful computer to compute backscatter locations, transmitting back to Pi to drive projector.
      * Potential latency issues.
    - Algorithm, Python/OpenCV optimisations.
  + Underwater testing.
    - Takes a long time, can be achieved with bubbles to replicate backscatter in controlled environment such as a tank.
    - *Probably best to carry out at the end of the project?*
  + Backscatter depth perception.
  + ML-based backscatter position tracking.
* Pursuing one point from the above four is complex enough for the project.
* Next meeting scheduled for 2pm on Thursday 9th.
  + In-person at the ISA.
  + Paul to send calendar invite.
  + Tour of the ISA, and desk allocation for me to work at.
* Actionable points:
  + For each of the four future work objectives, I’ll be carrying out preliminary research to understand what technologies, technicalities and techniques are required.
  + At the next supervision meeting at the ISA, I will be introduced to the end-result of Katie’s work. Using this I can better gauge what point to pursue.

# W/B 06-11-2023

## Intro to Real-Time Software [1]

Real time is a type of application that requires a guaranteed response within a strict timing constraint. There are two types of real-time and two metrics:

**Hard real-time:** If the event is not processed in a strict timing window, then bad things will happen.

**Soft real-time**: If the event is not processed in a not as strict timing window, then the system may degrade but it’s not as bad.

**Interrupt latency:** Time to process an interrupt.

**Scheduling latency:** Time for the OS Scheduler to start a processing task (Nominal and Worst-case).

RT can be handled by Raspberry Pi with a lot of tweaking:

* Current Linux / BSD kernels for the Raspberry Pi do not support real-time [as of 2015 when this article was made].
* Standard Linux or BSD install includes services that generate lots of overhead.
* Pi doesn’t have a real-time clock [But I think the Pi 5 comes with one now].
  + Expects device to be always connected to the internet.
  + Cannot generate deterministic timing pulses to control things like DC motors

Real Time Executive for Multiprocessor Systems (RTEMS) for Raspberry Pi:

* Open-source Real Time Operating System (RTOS).
* Used in Spaceflight, medical, and other real-time embedded applications.
* Ported to the Raspberry Pi *(w/ some limitations).*

## Supervision Meeting

* Meeting with Paul Mitchell and Benjamin Henson on Thursday at the ISA.
* Discussed real-time research findings.
* Tour of the ISA, and desk assignment.
* Scheduled bi-weekly in-person supervision meetings.
* Discussion about hypervisors with Ben
  + Using something like Docker, script can be packaged and assigned to a CPU core.
  + This can solve scheduling issues that was experienced when running directly on OS.
* Shown interest in the ‘Improving real-time computing’ aspects of this project.
  + Will be focusing on this, with aspects of underwater testing. If there is time at the end of the project, we can use the underwater-testing tank facility (if construction is completed by then).
* Rewrite project code in C.
  + Eliminate the overhead that Python introduces.
* More efficient logic.
  + Paul and Ben both mentioned to start simple – start with a simple thresholding and add fine tune.
  + Read older literature (e.g. 2010-era) on digital image processing – this should be a good starting point to build up from since there won’t be any added complexities.
* Compare algorithms for detecting and eliminating backscatter.
* Ben will send me some underwater recordings (GoPro footage), I can use this to program the code without needing to perform any underwater testing.
  + Since the existing code isn’t readily available (backed up to Git), I have asked Ben to take a backup of the RPi’s SD card (the one that’s currently in the lighting system, with the code that Katie produced).
    - Code has been printed in Katie’s report, however, Driver.py has been cut-off due to formatting issues (code starts from line 60 and previous lines are missing).

# Bibliography

|  |  |
| --- | --- |
| [1] | S. Doran, “How to Perform Real-Time Processing on the Raspberry Pi - SCALE 13X,” 19 June 2015. [Online]. Available: https://www.socallinuxexpo.org/sites/default/files/presentations/Steven\_Doran\_SCALE\_13x.pdf. [Accessed 6 11 2023]. |