





# Implementing Video Monitoring Capabilities by using hardware-based Encoders of the Raspberry Pi Zero 2 W

Thomas Ederer , <sup>a</sup>, Igor Ivkić , <sup>a,b</sup>

<sup>a</sup>*University of Applied Sciences Burgenland, Eisenstadt, Austria*

<sup>b</sup>*Lancaster University, Lancaster, UK*

---

## Abstract

Single-board computers, with their wide range of external interfaces, provide a cost-effective solution for studying animals and plants in their natural habitat. With the introduction of the Raspberry Pi Zero 2 W, which provides hardware-based image and video encoders, it is now possible to extend this application area to include video surveillance capabilities. This paper demonstrates a solution that offloads video stream generation from the Central Processing Unit (CPU) to hardware-based encoders. The flow of data through an encoding application is described, followed by a method of accelerating image processing by reducing the number of memory copies. The paper concludes with an example use case demonstrating the application of this new feature in an underwater camera.

*Keywords:* Raspberry Pi, GPU, Video, Encoder, Monitoring

---

## Metadata

Nr.	Code metadata description	Please fill in this column
C1	Current code version	v0.4.0
C2	Permanent link to repository	<a href="https://github.com/tederer/octowatch-videoservice">https://github.com/tederer/octowatch-videoservice</a>
C3	Permanent link to Reproducible Capsule	<a href="https://github.com/tederer/octowatch-videoservice/tree/v0.4.0">https://github.com/tederer/octowatch-videoservice/tree/v0.4.0</a>
C4	Legal Code License	MIT License
C5	Code versioning system used	git
C6	Software code languages, tools, and services used	C++
C7	Compilation requirements, operating environments & dependencies	Raspbian GNU/Linux 12(bookworm)
C8	Link to developer documentation/manual	<a href="https://underwater-camera-project.github.io">https://underwater-camera-project.github.io</a>
C9	Support email for questions	thomas@orchideenvermehrung.at

Table 1: Code metadata

## 1. Motivation and Significance

The study of animals and plants in their natural habitat requires undisturbed observation, as this is the only way to ensure the collection of reliable data. The observational data collected helps to better understand how they function within the ecosystem and can be used to advocate for the protection of threatened environments. To achieve this goal, researchers in the field of biology [1, 2] are already using inexpensive single-board computers such as the Raspberry Pi [3, 4] to build measurement and control systems. Previous publications [5–18] demonstrate the wide range of applications for the *Raspberry Pi*. These projects have in common the connection of a camera module to a single-board computer. Software executing on the CPU of the computer is responsible for converting the camera’s raw images into a video stream. However, this type of video generation consumes computing time of the CPU, which cannot be used for other tasks. The following figure shows a comparison of software- and hardware-based Joint Photographic Experts Group (JPEG) encoders.

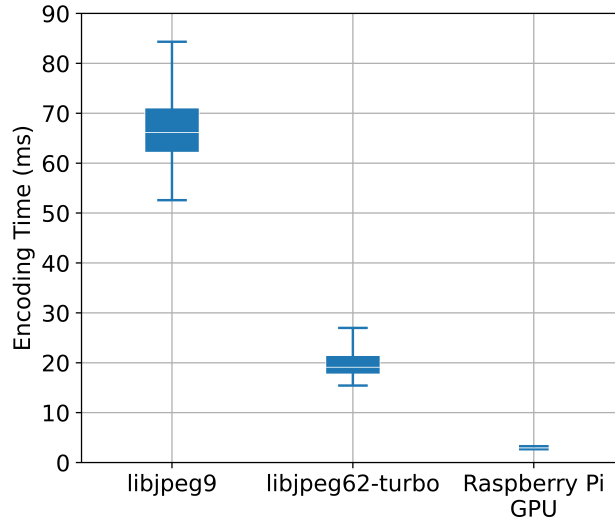


Figure 1: Comparing the Encoding Time required for Converting Raw Camera Images (YUV420, 800 x 600 pixels) to JPEG Format using various Encoding Methods.

The performance of the encoding was evaluated by measuring the time taken to encode 1,000 images. Before using the images, they were converted to the output format (YUV420) of the camera module, and then each image was encoded 100 times. The time required for this process was measured, and the results demonstrated that the speed of *libjpeg9* is not sufficient to produce a video stream with 30 frames per second. Using *libjpeg62-turbo*

is a viable option; however, it occupies one CPU core almost entirely. The optimal solution is to perform the encoding of the images in hardware, as this is fast (approximately 4 milliseconds per image) and reduces the load on the CPU.

In our previous publication [19], we demonstrated a way to overcome this limitation by using the hardware-based image and video encoders of the *Raspberry Pi Zero 2 W* for video stream generation. An underwater camera for remote monitoring of octopuses in the northern Adriatic Sea was developed to generate two video streams: one for local recording and another integrated into a web page for remote access via the Internet. This eliminates the necessity of observers being on-site, thereby reducing the on-site presence and assisting research projects in undertaking their observations in a more cost-effective manner. The following figure shows the housing (made of water pipes) of the underwater camera and its components mounted in an adaptable installation framework:

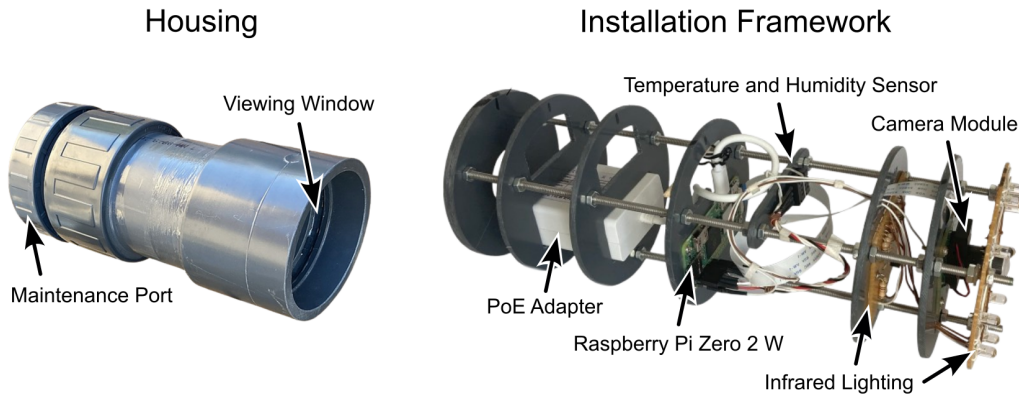


Figure 2: Underwater Camera using the hardware-based Image and Video Encoders of the Raspberry Pi Zero 2 W for Video Stream Generation.

This paper is a continuation of [19] and shows how to use the hardware-based video and image encoders of the *Raspberry Pi Zero 2 W* to integrate video surveillance capabilities into existing measurement and control tasks. First, we present the overall software architecture of the video service and explain how the data flow is used to generate two video streams from a sequence of raw images. Next, we discuss the effectiveness of the underlying framework in minimizing image copying. Finally, we show how a combination of implemented software and freely available hardware components can be used to provide two independent video streams in an underwater camera.

## 2. Software description

The purpose of the Video Service is to convert the images from the camera module into two independent video streams and make them available over the network. At startup, the service configures the camera module and then waits for images from the scene in front of the camera. Each image received is passed to the hardware-based encoders for conversion to H.264 and Multipart JPEG (MPJPEG) formats, which generate a sequence of Network Abstraction Layer (NAL) packets (for H.264) and JPEG images (for MPJPEG). Before the packets are sent, each JPEG image must be augmented with additional data so that it can be processed by the receiver as a MPJPEG video stream. The converted data can be consumed independently of each other via separate network ports.

### 2.1. Software architecture

The software is designed as a Linux service implemented in the programming language C++ on top of the libraries *libcamera* (v0.4.0), *Boost.Asio* (1.82.0) and the Video for Linux API Version 2 (V4L2) Application Programming Interface (API) (linux-libc-dev:6.1.140-1). The object-oriented interface of *libcamera* reduces the time taken to integrate camera modules into applications by hiding device-specific implementation details. Figure 3 illustrates the software stack of the Video Service consisting of existing components (light grey) and those to be implemented (dark grey).

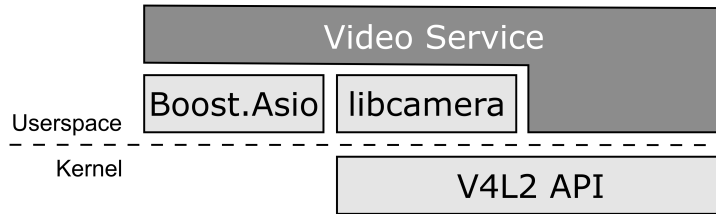


Figure 3: Software stack of the Video Service, consisting of existing Components (light grey) and those to be implemented (dark grey).

As hardware-based encoders are not yet supported by the *libcamera* library, the underlying V4L2 API had to be used for this part of the implementation. The network communication components are based on *Boost.Asio*, which is a cross-platform C++ library for network and low-level I/O programming. It is possible to modify camera settings during runtime by utilizing the Remote Control Interface of the service. The core of the Video Service is a game loop [20], which uses a number of components. At the consuming end the service receives two images in different resolutions from the camera module

and transfers them to the corresponding encoder. This offloads image scaling from the CPU to the camera module. Images with a high resolution (1920 x 1080 pixels) are used to generate the H.264 video stream [21], while those with a lower resolution (800 x 600 pixels) are used to generate the MPJPEG video stream. Furthermore, the multipart streamer component adds metadata [22] to each JPEG image to enable web browsers to render the MPJPEG stream as part of a web page. In order to facilitate the delivery of the two video streams via the network, two TCP servers are used, each accepting connections on a distinct port. The following figure illustrates the data flow in the Video Service:

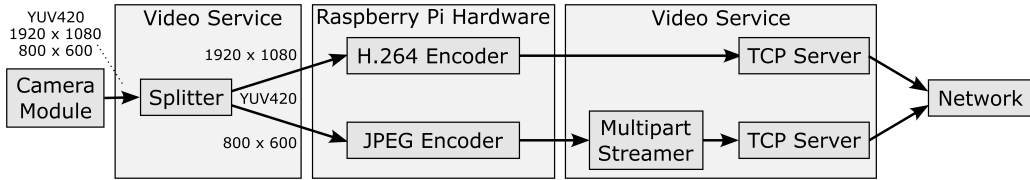


Figure 4: Data flow in the Video Service used to Produce two Video Streams from a Sequence of Raw Images.

To convert the sequence of images (provided by the camera module) into a video stream, each image has to be processed by different components of the Video Service. In the worst case, images have to be copied when they are provided to, for example, the encoder. Such copy operations take a long time and have a large impact on the maximum throughput. To reduce copying to a minimum, V4L2 uses the *DMABUF* method. When a new image is downloaded from the camera module, the consumer of the image receives two integer numbers, a file descriptor and a memory offset. With these two values it is possible to map the memory of the image to the memory of the current process without copying data. This reduces the number of copies to one, as the network interface is the only one that requires a copy of the data due to the asynchronous nature of the data transfer.

## 2.2. Software functionalities

The video service presented can be used to integrate video surveillance capabilities into control and measurement systems using a low-cost *Raspberry Pi Zero 2 W* and a camera module. By using the service, the camera module gets configured to provide two images with different resolutions to generate two video streams: one in H.264 format and another in MPJPEG format. There is also an interface to modify the properties of the camera module.

### 2.2.1. H.264 video stream

The video stream is provided by the service via network port 8888, with a resolution of 1920 x 1080 pixels. Using H.264 encoding with its ability to compress data over multiple frames in the video stream ensures a bandwidth requirement of 10 MBit/s. The encoded video data is transmitted using NAL packets, which can be stored locally or on a server in the cloud for later playback and processing. Real-time playback can be facilitated by using the free and open source cross-platform multimedia player *VLC*, for example. Free tools such as *FFmpeg* can be used for recording the video stream.

### 2.2.2. MPJPEG video stream

Network port 8887 is used by the service for the transmission of the stream, with a resolution of 800 x 600 pixels. MPJPEG encoding enables the integration of the stream into web pages, and by adjusting the image compression rate to its maximum, a bandwidth requirement of 2.5 MBit/s can be achieved. The combination of H.264 for local recording and MPJPEG for internet transmission can reduce the on-site presence, thereby assisting research projects in conducting their observations in a more cost-effective manner. Environment variables can be used to configure the compression rate and the type (Graphics Processing Unit (GPU) or CPU) of the JPEG encoder used.

### 2.2.3. Required Network Bandwidth

In order to measure the required network bandwidth (Figure 5) for the transmission of the described video streams, three scenarios were selected for the MPJPEG video stream, covering the range of supported image quality values (0 – 95). Two scenarios were used to evaluate the network bandwidth of the H.264 video stream: (1) with no activity in front of the camera, and (2) with activity in the recorded scene.

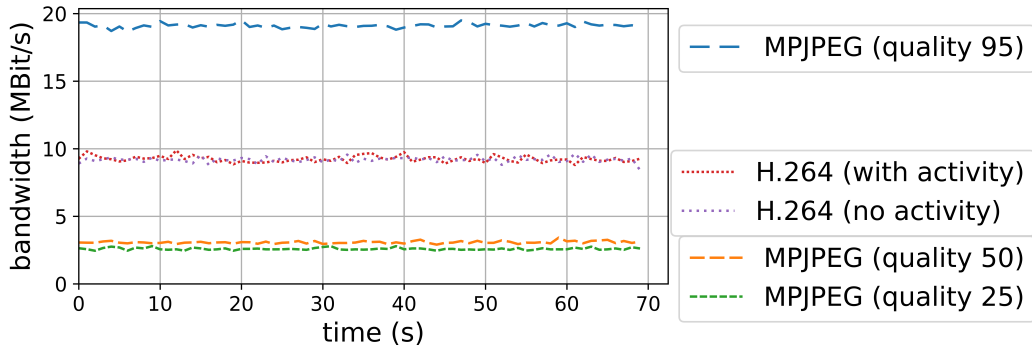


Figure 5: Required Network Bandwidth depending on Encoding and Image Quality.

### 3. Illustrative example

In the context of the *Octopus Intelligence* project of **Mare Mundi**, an underwater camera was constructed for the observation of octopuses in the northern Adriatic Sea [19]. The video streams were generated using a *Raspberry Pi Zero 2 W*, a camera module and the Video Service presented in this paper. Given the high compression rate, the H.264 format was used to record the camera output on-site for later processing. In order to facilitate real-time observation, additional services had to be developed and are available for free at the **Underwater Camera Project**. A web server was used to provide a web page [23] containing the MPJPEG video stream available from any location with internet access, as well as controls for modifying the camera settings (e.g. brightness). The following figure illustrates the architecture of the video-related components of the underwater camera:

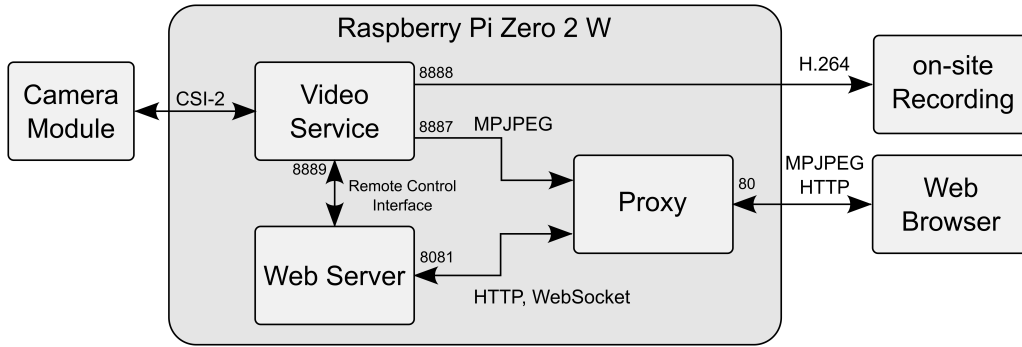


Figure 6: Video Related Part of the Architecture used to Monitor Octopuses in the northern Adriatic Sea.

A proxy is required to make the content of the Video Service and the web server available on the same port. Direct access to the web server is technically possible, but it makes it impossible for web browsers to determine whether the content on the website is from the same source as the video. As a result, the video download is blocked by the browser. A proxy (like *nginx*) or application gateway hides both services behind a single interface to avoid requests getting blocked. All those services were configured to run as Linux services. This enables the operating system of the *Raspberry Pi* to start them at system boot and to collect their log output in the system journal. To reduce the risk of getting attacked, the firewall was configured to drop all incoming connections to port which are different to those of the H.264 video stream and the proxy.

## 4. Impact

The Video Service presented was developed as part of the implementation of a low-cost underwater camera used by **Mare Mundi** to study the intelligence of octopuses in the northern Adriatic Sea [19]. The findings demonstrate that the hardware-based video and image encoders of the Raspberry Pi Zero 2 W possess sufficient capabilities to concurrently generate two video streams (H.264 and MPJPEG) with differing transmission bandwidth requirements. This facilitates the transmission of the video stream over the internet and its local recording. In contrast, commercially available camera systems generally provide a single video stream, without the possibility of customization, such as adding additional sensors or actors. The proposed solution overcomes these limitations by leveraging the *Raspberry Pi's* ability to provide a variety of external interfaces for connecting devices, and its software can be modified and extended by anyone. From a financial perspective, the transmission of video content via the internet, in conjunction with local recording, has the potential to reduce the necessity for on-site researchers, thereby leading to cost reductions while maintaining the capacity to conduct observations. The local recording of video output is another potential cost-saving measure, as it allows users to access videos when they have time to do so.

In the context of future projects, the potential exists for detecting activity in the video and providing users with the corresponding sequences from the high quality local recording for later analysis. The elimination of time-consuming viewing of the recording further reduces project costs.

## 5. Conclusions

The use of video surveillance presents certain challenges for research projects, as the acquisition of cameras is expensive and the devices cannot be extended with sensors and actuators. Low-cost single-board computers such as the *Raspberry Pi* have demonstrated their efficacy for measurement and control tasks in numerous publications. With the *Raspberry Pi Zero 2 W's* new hardware-based video and image encoders, it is now possible to produce high quality video at low cost. In this paper, we use a practical example to show how the efficient use of encoders has turned the Raspberry Pi into an underwater camera that generates two video streams. One stream (H.264) is used for on-site recording, while the other video stream (MPJPEG) is embedded in a website that can be accessed via the Internet. Using these two video streams allows researchers to observe the underwater environment without disturbing it with their presence, also eliminating the need to be present on site and reducing the costs.



## References

- [1] J. W. Jolles, Broad-scale applications of the Raspberry Pi: A review and guide for biologists, *Methods in Ecology and Evolution* 12 (9) (2021) 1562–1579. doi:10.1111/2041-210X.13652.
- [2] B. M. Allan, D. G. Nimmo, D. Ierodiaconou, J. VanDerWal, L. P. Koh, E. G. Ritchie, Futurecasting ecological research: The rise of technoecology, *Ecosphere* 9 (5) (2018) e02163. doi:10.1002/ecs2.2163.
- [3] S. E. Mathe, H. K. Kondaveeti, S. Vappangi, S. D. Vanambathina, N. K. Kumaravelu, A comprehensive review on applications of Raspberry Pi, *Computer Science Review* 52 (2024) 100636. doi:10.1016/j.cosrev.2024.100636.
- [4] A. Nayyar, V. Puri, Raspberry pi-a small, powerful, cost effective and efficient form factor computer: A review, *International Journal of Advanced Research in Computer Science and Software Engineering (IJARCSSE)* 5 (2015) 720–737.
- [5] V. J. D. Almero, M. G. B. Palconit, J. D. Alejandrino, R. S. Concepcion, R. R. P. Vicerra, E. Sybingco, A. A. Bandala, E. P. Dadios, Development of a Raspberry Pi-based Underwater Camera System for Inland Freshwater Aquaculture, in: 2021 IEEE 13th International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment, and Management (HNICEM), IEEE, Manila, Philippines, 2021, pp. 1–6. doi:10.1109/HNICEM54116.2021.9731987.
- [6] F. Cazenave, C. Keczy, M. Risi, S. H. D. Haddock, SeeStar: A low-cost, modular and open-source camera system for subsea observations, in: 2014 Oceans - St. John's, IEEE, St. John's, NL, 2014, pp. 1–7. doi:10.1109/OCEANS.2014.7003077.
- [7] G. Coro, M. Bjerregaard Walsh, An intelligent and cost-effective remote underwater video device for fish size monitoring, *Ecological Informatics* 63 (2021) 101311. doi:10.1016/j.ecoinf.2021.101311.
- [8] E. P. Dadios, V. J. Almero, R. S. C. Ii, R. R. P. Vicerra, A. A. Bandala, E. Sybingco, Department of Manufacturing Engineering and Management, De La Salle University (DLSU) 2401 Taft Avenue, Malate, Manila 1004, Philippines, Center for Engineering and Sustainability Development Research, De La Salle University (DLSU) 2401 Taft Avenue, Malate, Manila 1004, Philippines, Department of Electronics and

- Computer Engineering, De La Salle University (DLSU) 2401 Taft Avenue, Malate, Manila 1004, Philippines, Low-Cost Underwater Camera: Design and Development, *Journal of Advanced Computational Intelligence and Intelligent Informatics* 26 (5) (2022) 851–858. doi:10.20965/jaciii.2022.p0851.
- [9] I.-J. Huang, C.-C. Hung, S.-R. Kuang, Y.-N. Chang, K.-Y. Huang, C.-R. Tsai, K.-L. Feng, The Prototype of a Smart Underwater Surveillance System for Shrimp Farming, in: 2018 IEEE International Conference on Advanced Manufacturing (ICAM), IEEE, Yunlin, 2018, pp. 177–180. doi:10.1109/AMCON.2018.8614976.
  - [10] S. H. Jury, H. Howell, D. F. O’Grady, W. H. Watson Iii, Lobster trap video: In situ video surveillance of the behaviour of *Homarus americanus* in and around traps, *Marine and Freshwater Research* 52 (8) (2001) 1125. doi:10.1071/MF01096.
  - [11] X. Mouy, M. Black, K. Cox, J. Qualley, C. Mireault, S. Dosso, F. Juanes, FishCam: A low-cost open source autonomous camera for aquatic research, *HardwareX* 8 (2020) e00110. doi:10.1016/j.ohx.2020.e00110.
  - [12] D. Novy, L. Kawasumi, J. Ferguson, M. Sullivan, P. Bell, J. S. Chow, J. B. de Sousa, K. A. Cantner, B. Woodward, A. Adams, K. L. Bell, Maka Niu: A low-cost, modular imaging and sensor platform to increase observation capabilities of the deep ocean, *Frontiers in Marine Science* 9 (2022). doi:10.3389/fmars.2022.986237.
  - [13] B. T. Phillips, S. Licht, K. S. Haiat, J. Bonney, J. Allder, N. Chaloux, R. Shomberg, T. J. Noyes, DEEPi: A miniaturized, robust, and economical camera and computer system for deep-sea exploration, *Deep Sea Research Part I: Oceanographic Research Papers* 153 (2019) 103136. doi:10.1016/j.dsr.2019.103136.
  - [14] A. Purser, U. Hoge, J. Lemburg, Y. Bodur, E. Schiller, J. Ludszuweit, J. Greinert, S. Dreutter, B. Dorschel, F. Wenzhöfer, PlasPI marine cameras: Open-source, affordable camera systems for time series marine studies, *HardwareX* 7 (2020) e00102. doi:10.1016/j.ohx.2020.e00102.
  - [15] J. Giddens, A. Turchik, W. Goodell, M. Rodriguez, D. Delaney, The National Geographic Society Deep-Sea Camera System: A Low-Cost Remote Video Survey Instrument to Advance Biodiversity Observation

- in the Deep Ocean, *Frontiers in Marine Science* 7 (2021) 601411. doi:10.3389/fmars.2020.601411.
- [16] P. Lertvilai, The In situ Plankton Assemblage eXplorer (IPAX): An inexpensive underwater imaging system for zooplankton study, *Methods in Ecology and Evolution* 11 (9) (2020) 1042–1048. doi:10.1111/2041-210X.13441.
  - [17] M. Ambrož, Raspberry Pi as a low-cost data acquisition system for human powered vehicles, *Measurement* 100 (2017) 7–18. doi:10.1016/j.measurement.2016.12.037.
  - [18] H. Baghdadi, K. Rhofir, M. Lamhamdi, Smart portable system for monitoring vibration based on the Raspberry Pi microcomputer and the MEMS accelerometer, *International Journal of Informatics and Communication Technology (IJ-ICT)* 12 (3) (2023) 261. doi:10.11591/ijict.v12i3.pp261-271.
  - [19] T. Ederer, W. Slany, I. Ivkić, Reducing Underwater Observation Costs by Leveraging Cloud Technology (Accepted for Publication, 2024, December), 11th International Conference on Computational Science and Computational Intelligence (2024).
  - [20] L. Valente, A. Conci, B. Feijó, Real time game loop models for single-player computer games, *Proceedings of the IV Brazilian Symposium on Computer Games and Digital Entertainment* (2005) 89–99.
  - [21] ITU-T, H.264 : Advanced video coding for generic audiovisual services (Aug. 2021). doi:11.1002/1000/11830.
  - [22] N. Borenstein, N. Freed, MIME (Multipurpose Internet Mail Extensions): Mechanisms for Specifying and Describing the Format of Internet Message Bodies, Tech. Rep. RFC1341, RFC Editor (Jun. 1992). doi:10.17487/rfc1341.
  - [23] M. S. Mikowski, J. C. Powell, *Single Page Web Applications: JavaScript End-to-End*, Manning, Shelter Island, NY, 2014.