Project Report

Capture device for documentation in fabrication workshops

May - July 2024

Ali EL GHOUL
Master SAR
28717628

Siyu ZHOU
Master SAR
21100712

Sorbonne Université
Faculté des Sciences et de l'Ingénierie

Contents

Acknowledgements

We would like to thank our project mentors, Gilles Bailly and Aline Baudry, for their patience, kindness and expertise in the field of robot/human interaction. Thanks to their guidance and advice, we have been able to combine Python programming, computer-aided design (CAD), and electronics to create an exciting project that we hope will be practical and make it easier to save the work done at FabLab.

We would also like to thank Sorbonne University's FabLab and the Institut des Systèmes Intelligents et de Robotique (ISIR) for providing us with the equipment we needed to complete the hardware and build the video capture system.

Finally, we would like to express our gratitude to all those who have contributed to this project in one way or another, whether through constructive discussions, criticism or moral support. Their help has been invaluable to the success of this project.

1 Introduction

This report introduces an innovative documentation system designed for fabrication labs, envisioned by Clara Rigaud. It begins by discussing the project's context and development, followed by a description of its hardware and software components. Key topics include the board selection and specifications, camera choices, and input/output interfaces. The mechanical design section covers box design, manufacturing, and interface specifics such as connectors and user interface design. Development efforts and documentation procedures are detailed, alongside an exploration of applications, advantages, limitations.

2 Project Context and Development

This project follows Clara Rigaud's research on knowledge acquisition in fabrication workshops [1]. In workshops, fabrication and creating knowledge resources are often conflicting activities, both cognitively and physically. Makers tend to focus entirely on their tasks, making documentation a challenging process that requires shifting attention away from fabrication, often leading to oversights, especially during problem-solving or urgent tasks.

The search for a suitable system began with several key requirements:

- **Variety of Units**: Different units should have varied properties to address specific needs of different activities. This includes variations in camera quality, size, and functionalities.
- **Feedback and Feedforward Mechanisms**: The system should provide real-time feedback on automated parameters and indicate the areas that devices can occupy or capture.
- **Capture Units**: Develop small, lightweight, and customizable capture units with high-quality cameras, easy to build and using commonly available workshop materials.
- **Automation Settings**: Include settings for users to automate functions such as start/stop recording and focus adjustments.
- **Privacy Controls**: Implement privacy controls like automatic blurring of backgrounds or sensitive areas.
- **Central Repository**: Store and manage all captured content on a local network for easy access and retrieval.
- **User Interface**: Design an intuitive interface for setting preferences, monitoring capturing activities, and managing recorded content.

The initial prototype was large (30cm x 20cm x 10cm) and heavy, equipped with one rotary encoder, two buttons, and two LEDs. It lacked instructions, was confusing to use, and required a direct connection to a screen, keyboard, and monitor to access the recorded videos, as accessing the private server was not available to retrieve them. The

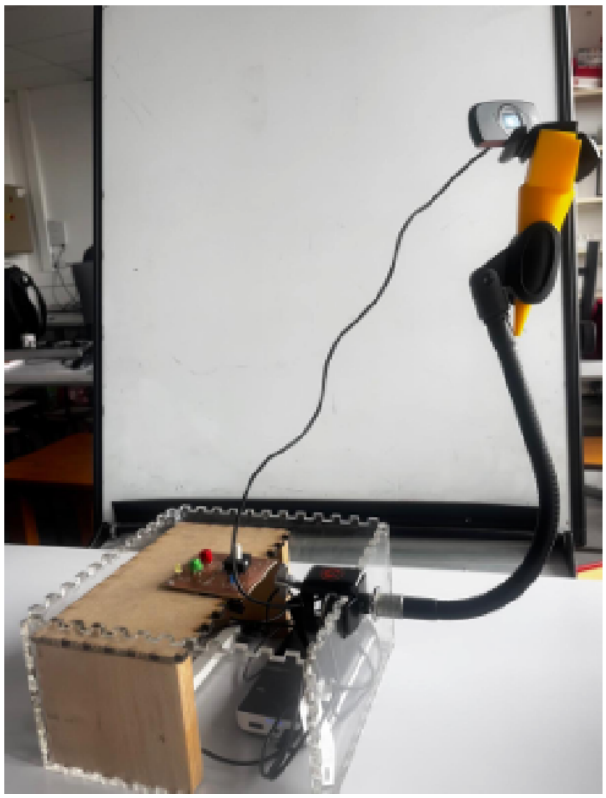

Figure 1: The old System

recordings were in Full HD but at a frame rate of only 5 fps. We decided to start from scratch and develop a more user-friendly, efficient system.

3 A Brief Description

CA-Push is equipped with a rotary encoder and a button for navigating and selecting settings, and an LCD screen for visualizing and adjusting the camera view. For video download, CA-Push hosts a web page that is easily accessible by any web browser. It runs on a power bank for ease of maintenance, permitting uninterrupted operation from plugging to unplugging, with 4 hours of autonomous filming (depends on the power bank) in 5 modes: Slow Motion, Normal, Eco-Normal, Time Lapse, and Eco-Time Lapse.

Additionally, the box dimensions are 115mm x 85mm x 74mm. The internal structure is divides into two layers, with the middle layer serving as a single-chip computer bracket for fixing the Raspberry Pi Zero 2W, and the bottom layer for placing the external power supply. The material used for the box is 3mm natural MDF. The dovetail design used as a hook system. This design ensures a secure fit and easy attachment, with the dovetail mortise fixed to the box using M3 screws.

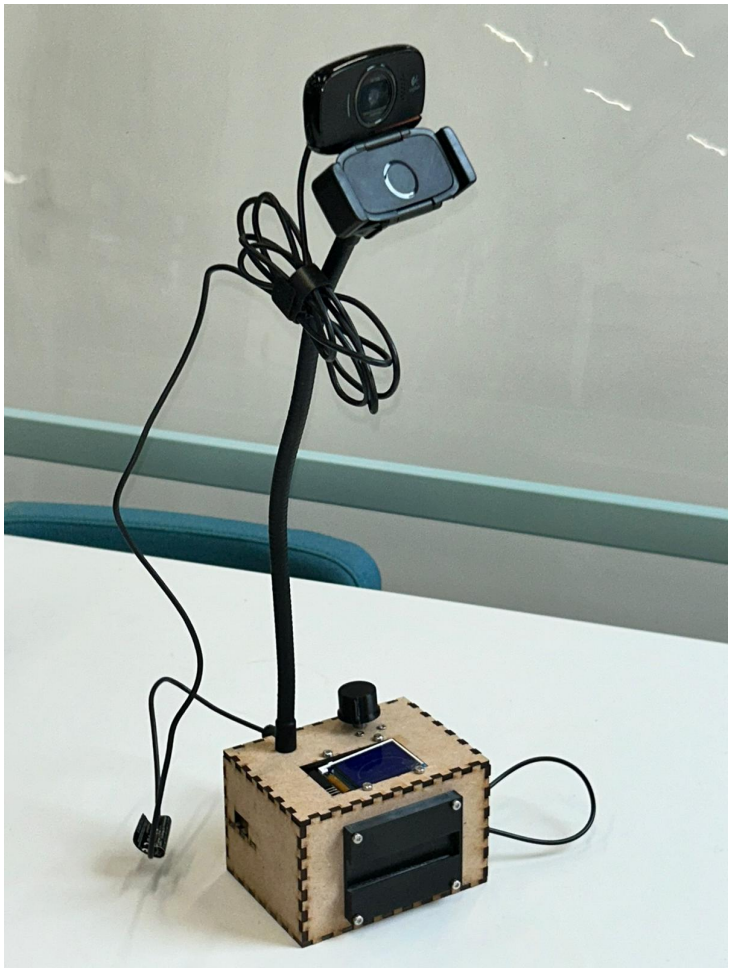

Figure 2: Ca-Push

4 Hardware and Software

In this section, we will discuss the integration between the hardware and software components of our documentation system. This includes an in-depth look at how the chosen hardware interacts with the software to achieve the desired functionality

4.1 The board

4.1.1 Choice of Board

For the board, we had a choice between a microcontroller computer and a single-board computer. Our tutors suggested that a microcontroller wouldn't have enough power to handle capturing and uploading videos, and its limited versatility could pose integration challenges for future system features compared to a full computer. We aimed for a compact board with low power consumption to ensure long autonomy. Therefore, we opted for a Raspberry Pi, one of the most popular single-board computers known for its accessibility and extensive documentation, specifically choosing the Zero 2W model.

4.1.2 Specifications of the Board

Capush runs on the Raspberry Pi Zero 2W[2] , the successor to the Raspberry Pi Zero W. It is up to five times faster than the original Raspberry Pi Zero while retaining the same compact form factor and maintaining a price below 20$. This tiny board possesses sufficient power to handle its designated tasks. It features a quad-core 64-bit ARM Cortex-A53 processor clocked at 1GHz, 512MB of SDRAM, and storage capacity up to 30GB (depending on the SD card used). Equipped with one mini-HDMI port and one micro USB port, we required a USB hub to connect peripherals such as the webcam, or a USB stick which was instrumental during the initial development stage for file transfers. We chose the Zero4U from UUGear, a 4-port USB hub designed for the Raspberry Pi Zero. It can be mounted back-to-back with the Raspberry Pi Zero, utilizing 4 pogo pins to connect to the PP1, PP6, PP22, and PP23 testing pads, thus eliminating the need for soldering. The hub operates in self-power mode, drawing power directly from the power supply, and can supply up to 2,000mA of current across all four USB ports.

Figure 3: Zero4U mounted on the Raspberry Pi

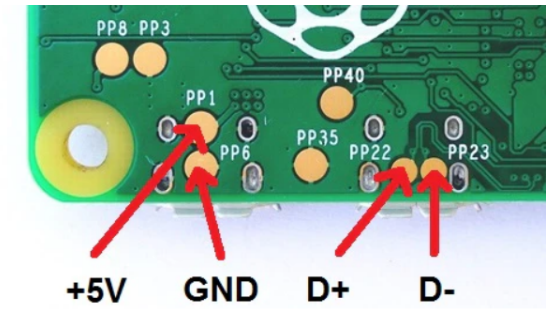

Figure 4: Pogo pins of the Raspberry Pi

4.1.3 Choice of Software

Development started on the Raspberry Pi 4, the board used for the old system, while waiting for the new components. Working with the RPi 4 was fairly easy and straightforward. We connected it to an external monitor via the micro HDMI port and used a keyboard and mouse for interacting with the system. We did not encounter any computing power issues during this phase. Transitioning to the RPi Zero 2W was more complex.

We installed Debian GNU/Linux version 12 with a desktop environment on the Raspberry Pi, with future plans to switch to an OS without a desktop environment to reduce system overhead. We aimed for a minimal setup, avoiding dependence on peripherals for system development: Headless Mode. This approach was chosen to facilitate replication of the system by e-makers without relying on potentially unavailable peripherals (such as a mini HDMI to HDMI cable) and to enable easier transitions to a non-desktop environment if desired. The USB hub could be omitted, and the camera could connect directly to the micro USB port of the Pi using a USB to micro USB connector.

Initially, we connected to the Raspberry Pi via SSH, but this proved insufficient for the development stage. We discovered that having a desktop environment was beneficial for various tasks such as scripting, file transfer, and file management. We fully utilized the desktop version by connecting to the Pi through VNC, enabling wireless projection of the HDMI output to our screen. However, this functionality is unnecessary for replicating or maintaining the system.

4.2 The camera

4.2.1 Choice of Camera

Using a webcam for our system was an obvious choice. The Raspberry Pi camera wasn't suitable for this project due to its fragile CSI connector, which could easily tear during system use. Additionally, its short length makes it difficult to extend, replace, or upgrade, especially when compared to a simple USB webcam. Instead, we opted to use the webcam from the previous project. The Logitech HD Webcam C525 supports video capture resolutions up to 720p and has a maximum frame rate of 30fps at 640x480. Unofficially, it can achieve 1080p resolution at a minimum frame rate of 5fps. The webcam features an activity/power LED indicator and includes auto-focus capabilities. The OpenCV library is used to control the camera.

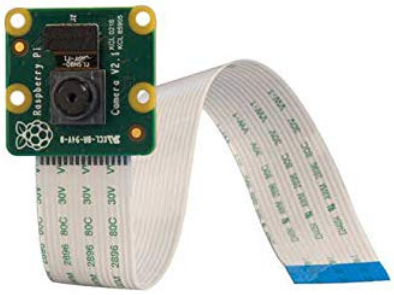

Figure 5: Raspberry Pi Camera V2 8MP

4.2.2 Capturing Settings

Capturing settings are composed of the capturing rate and the camera resolution, which are interlinked. The camera resolution determines the capturing rate. These

settings are intrinsic to the camera itself and defined by its hardware limitations. For example, you can record at a capturing rate of 24 FPS(Frames Per Second) with a resolution of 864x480 or up to a maximum resolution of 800x600. On the other hand, at 800x600 resolution, you can achieve a frame rate of only 24 FPS.

Here are some examples of commonly used FPS values with their corresponding resolutions from our camera:

- 1920x1080 at 5 FPS (max resolution)
- 1280x960 at 7.5 FPS (max resolution)
- 1280x720 at 10 FPS
- 960x720 at 15 FPS
- 960x544 at 20 FPS (max resolution)
- 800x600 at 24 FPS (max resolution)
- 800x448 at 30 FPS (max resolution)

That being said, setting the resolution will automatically adjust the associated capturing frame rate.

4.2.3 Writing Settings

Writing settings are composed of the writing frame rate and the choice of file extension of the video.

The writing frame rate defines the rate at which frames are written into the video. For example, at 60 FPS writing, one second of the video will contain 60 images, independent of the capturing frame rate, and it can vary from 1 to infinity.

How do these two factors work together? Let's consider recording at 30 FPS, where we capture 30 images per second and write at x frames per second. The duration of the video can be calculated as: duration of video $= \frac{\text{duration of recording} \times \text{capturing frame rate}}{\text{writing frame rate}}$.

If we choose to write at 60 FPS, we will get an accelerated video with a length equal to half the capture duration; this is known as time-lapse. More images are written per second than captured. On the other hand, if we choose a writing FPS lower than the capturing FPS, the resulting video will be longer, resembling slow motion.

Setting the writing frame rate equal to the capturing frame rate theoretically results in a normal speed video, although this may vary slightly depending on factors such as video duration and system conditions (e.g., temperature), typically up to 4-5 seconds with the original duration.

For the file extension, we chose .avi due to its ability to maintain higher video quality than .mp4 files. We utilized the XVID FourCC codec [3] for encoding, which is commonly used for AVI files. However, it is worth noting that .avi files require significantly more storage[4] space compared to .mp4 files and are less compatible with common media players.

4.3 Inputs

4.3.1 The Rotary Encoder

This encoder features a flattened shaft and an integrated push button, adding to its functionality. It operates with infinite rotation capability and provides 20 cycles per revolution. Powered by 5V from the Raspberry Pi, it incorporates 3 integrated 10K resistors (2 for the rotary encoder and 1 for the button). These resistors are crucial for limiting current input to the GPIO(General Purpose Input Output) pins and also serve as pull-up resistors.

The rotary encoder outputs two signals: clk and DT. When the clock (clk) state changes, it indicates that the encoder has been rotated. The system then checks if the data (DT) state is different from the clock state and if it's low. If so, the position value is incremented, indicating a clockwise rotation. Conversely, if the data state matches the clock state, the position value is decremented, indicating a counter-clockwise rotation.

This rotary encoder is utilized to navigate through menu options effectively.

4.3.2 The Button

The button integrated in the rotary encoder serves two tasks: turning on and off the system, and selecting options from the menu. When pressed, the signal SW is low, and when released, it's high. A simple click is used for selecting options; the system tracks the change in the state of the signal by comparing the last recorded state with the current state. It waits for the button to be released to register a click and only clears the registered click after the system has taken it into consideration, ensuring that one click is registered regardless of how long the button is held and no clicks are missed.

For turning the system on and off, a long-click feature is integrated. By adding a counter, if the button is pressed (SW state is low) for more than 6 seconds, the system registers a long click and initiates the shutdown action immediately (after saving files, if recording). After shutdown, the Raspberry Pi enters a halted state where all processes are terminated, though it still draws a small amount of power from the supply. To turn it on again, a simple click using the same button is sufficient.

To enable the functionality of waking the Raspberry Pi from its halting state when the button is pushed, the SW pin of the rotary encoder must be connected to GPIO3 pin of the Raspberry Pi, shorting this GPIO pin to ground [5] .

4.3.3 Development

To ensure fluidity in the rotary encoder and to never miss a click, the system must be able to prioritize inputs from these pins. There are two methods[6] for reading input signals from GPIO pins: polling and interrupt handling. In interrupt handling, the system informs the CPU that it needs attention when there are rising or falling edges on the GPIO pins. It can also include a debounce feature to filter out button noise. On the

other hand, polling involves the CPU constantly checking the status of the GPIO pins, which consumes more power and system resources.

Clearly, the interrupt handling option is best suited for our application. However, unfortunately, things did not go as planned. Let me explain:

There are three main libraries for controlling GPIO pins:

- **lgpio**: A modern, kernel-friendly, full-featured library for advanced and stable GPIO control.[7]
- **RPi.GPIO**: A legacy library providing direct hardware access for basic GPIO operations, now considered less stable and more limited.[8]
- **gpiozero**: A high-level, user-friendly library that abstracts GPIO details and can use different backend drivers (including lgpio and RPi.GPIO) to suit different needs and environments.

Initially, we chose to use the lgpio library to benefit from its interrupt mode, as RPi.GPIO cannot handle edge detection[9]. However, we encountered an issue when integrating the screen. The screen drivers required the old RPi.GPIO library, and installing both lgpio and RPi.GPIO in the same Python environment was not feasible because both packages install a module named RPi.GPIO, which causes conflicts.

Therefore, we settled for polling. Initially, we did not observe any noticeable issues with the button, but when it came to the rotary encoder, we experienced significant drops in fluidity. This was especially apparent when integrating the screen and menu graphics, which demanded processing power. After numerous iterations, we managed to resolve these issues.

4.4 Outputs

4.4.1 The Screen

The 1.8” display offers a resolution of 128 x 160 pixels and is equipped with an ST7735R IC driver operating at a logic level of 3.3V. It utilizes an SPI interface for communication and features a non-controllable LED backlight.

Initially, we developed visual indicators for user feedback using LEDs: a green LED for power, a red one for errors, and white and blue LEDs for recording and file writing statuses. However, this setup proved insufficient. With only LEDs, users could only start or stop recording, lacking control over video quality and unable to preview the recording without connecting to a monitor.

This small screen provided the optimal solution. Despite consuming approximately 100 mA (similar to the combined consumption of five LEDs at approximately 20 mA), it significantly enhanced the user experience. The screen played a crucial role in providing users with visual feedback, enabling them to control recording settings and benefiting from all available camera options.

4.4.2 The Menu

The menu serves as the central hub where all the components are linked together. Developing the menu took a lot of time and iteration to find the sweet spot between user-friendliness and functionality. Ultimately, we ended up with five OS configurations:

1. **OS1:** Uses LEDs as visual indicators (as described above).

2. **OS2:** The first OS to integrate the screen. It displays a blue-filled screen with the 'FPS = writing frame rate' written in white. Rotating the rotary encoder changes the writing frame rate only, and a simple click initiates the recording. This OS served as a proof of concept, demonstrating the potential of the screen. Although a bit confusing and limited in options, it is super fluid due to its simplicity.

3. **OS3:** This OS introduced a major upgrade to the menu. It integrates a new feature at the heart of our system: 'Calibrate Position' . This option allows the user to temporarily view the camera feed to adjust it according to their preference. It also includes options for FPS adjustment and recording. Navigation through the system is done using the rotary encoder, using bounding boxes to display the selected option. One click validates the option, and another click returns to the menu.

4. **OS4:** OS4 builds upon OS3 by introducing a new feature called "speed" instead of FPS adjustment. Instead of directly adjusting FPS, the speed menu displays pre-defined multipliers (e.g., 0.25x, 1x, 2x, 4x, 8x) to set the writing FPS based on the capturing FPS. An advanced options section was added for changing capturing frames and resolution, but this feature was later abandoned due to its complexity for users.

5. **OS5:** OS5 represents a complete remake of the menu. Users are presented with pre-defined options for camera settings based on the duration of their video, and replacing 'Capture Position' by 'Cam view'. This version offers a much more friendly GUI, allowing users to focus on their tasks without getting confused by different settings. It provides a simple yet powerful and intuitive way to control the system. Navigation through options is done using the rotary encoder, with a simple click to validate or start recording, and another click to stop or go back. In this version, the long push feature was introduced to turn the system on and off. The fluidity issue in interactions between the rotary encoder and display was resolved. During recording, the screen displays the message "Recording" along with the resolution and recording mode.

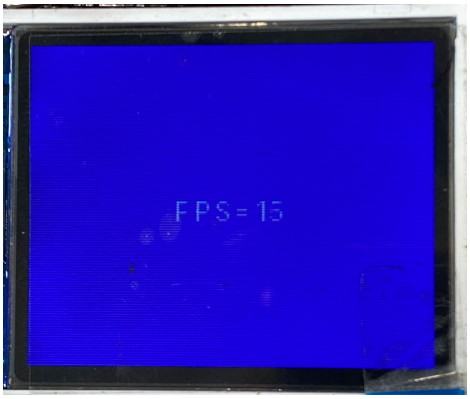

Figure 6: os2

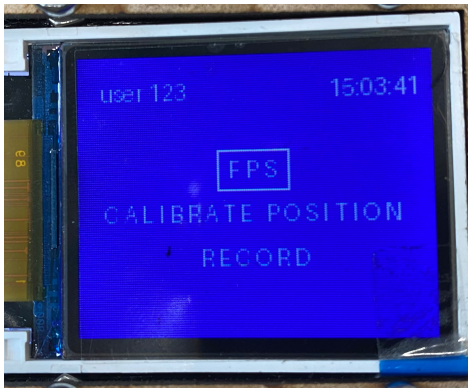

Figure 7: os3

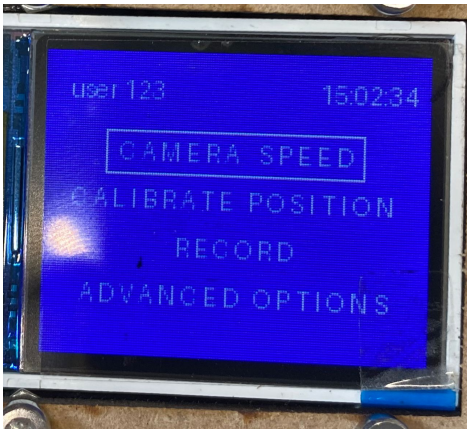

Figure 8: os4

Figure 9: os5

- 1 minute: `writing frame rate = 10, resolution = Wide VGA (800 x 448), mode='Slow Motion'` (capture frame rate = 30)
- 5 minutes: `writing frame rate = 22, resolution = Super VGA (800 x 600), mode='Normal'` (capture frame rate = 24)
- 30 minutes: `writing frame rate = 28, resolution = Wide VGA (800 x 448), mode='Eco - Normal'` (capture frame rate = 30)
- 1 hour: `writing frame rate = 100, resolution = Full HD (1920 x 1080), mode='Time-lapse'` (capture frame rate = 5)
- 3 hours: `writing frame rate = 600, resolution = HD (1280 x 720), mode='Eco - Time-lapse'` (capture frame rate = 10)

We wanted to utilize the full spectrum of capabilities of our camera. Shooting at high resolutions like 1080p or 720p in normal mode wouldn't look good due to the low frame rates, but these resolutions work perfectly for time-lapse videos where fewer frames per second are captured. For example, in a 3-hour long video, a lower resolution helps save storage space. Hence, we chose 720p resolution, written at a frame rate of 600, calculated to condense the 3-hour footage down to 3 minutes.

For normal recording options, a smooth frame rate of around 24-30 FPS was targeted, with higher resolutions allocated to shorter recording periods. For the 1-minute slow-motion option, we used our highest capturing FPS capability, writing it at one-third the speed to achieve a smooth and detailed slow-motion effect.

4.4.3 Development

Making sure that the rotary encoder worked smoothly with the menu was a critical part of our project. To achieve this, we first optimized how images are displayed on the screen. We built static screen menus, such as the shutdown screen and the main menu with options, which are generated only once at startup. This approach reduces the system load as the only element being dynamically rendered is the bounding box around the options. A white box is drawn around the selected option, and a background color box around the others. The rendered image is displayed only when a turn is detected, ensuring that system resources are not wasted on unnecessary refreshes.

Despite an initial improvement in fluidity, it wasn't sufficient. We further tested the processing time of the rotary encoder by measuring the time taken by the function responsible for detecting a turn. Two methods were tested:

- **Method 1:** A two-phase detection algorithm where the system detects two state changes for each turn (high to low, then low to high), incrementing or decrementing the encoder by 0.5. The measured time was 1.93×10^{-5} seconds.
- **Method 2:** A single-phase detection where the system detects only when the input is low, incrementing or decrementing by 1 (an integer operation). The measured time was 1.51×10^{-5} seconds, which is approximately 1.2 times faster.

Further testing confirmed these results. We also measured the time taken to execute the entire menu control method (`control_menu_0()`):

- **One-phase detection:** processing time when turned = 0.032 seconds; processing time when not turned = 0.007 seconds.
- **Two-phase detection:** Similar results as the one-phase detection.

We then decided to render all graphics and handle inputs from the rotary encoder only when a turn is detected. The results were phenomenal:

- processing time when not turned = 1.382×10^{-5} seconds.
- processing time when turned = 9.53×10^{-6} seconds.

This approach increased reactivity by 522 times, ensuring smooth system operation, and resulted in a 20% drop in CPU usage when running directly from a terminal. You can find a video documenting the fluidity difference on our YouTube channel.

4.5 Additional Notes

4.5.1 File Upload and Server Hosting

In an ideal scenario, the system would upload files to an external server for storage and additional processing, such as labeling and face blurring. However, integrating the Raspberry Pi with an external server proved to be too challenging.

As an alternative, we hosted a server directly on the Raspberry Pi using the open-source Nginx web server. Nginx is known for its high performance, stability, and low resource consumption, making it an ideal choice for our needs. The server serves a PHP file that indexes and displays the available videos, allowing users to easily navigate and download them.

Devices connected to the system's network can access and download stored videos by simply entering the associated IP address into a web browser. This process is user-friendly and can be done using a computer or a smartphone. The Nginx server has little to no impact on the system's performance when not actively used for downloading videos, ensuring that the primary functions of the system remain unaffected.

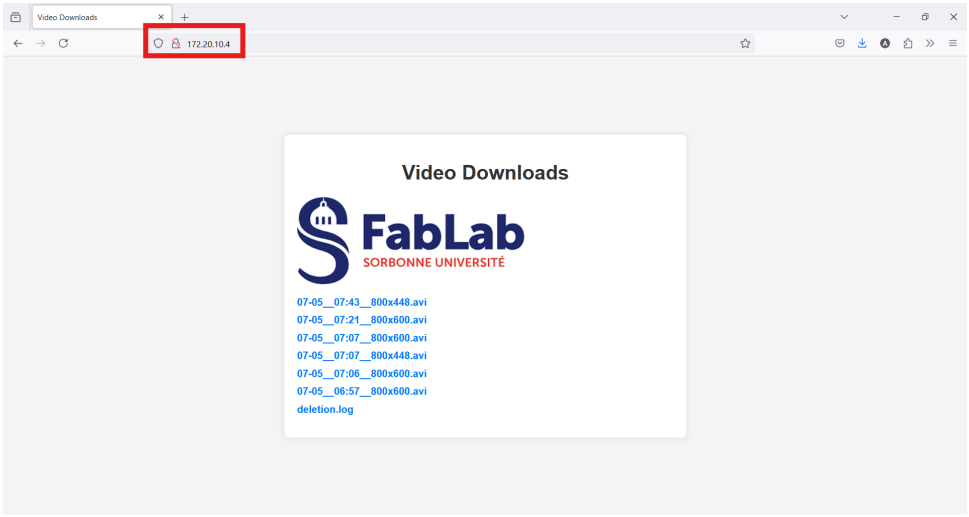

Figure 10: Video Download page

4.5.2 Services and Cleanup

For system maintenance, a cleanup service was created to automatically delete all videos after 2:05 AM each day. This ensures that the storage space on the Raspberry Pi is managed efficiently and prevents the system from becoming overloaded with excessive video files. When these videos are deleted, they are permanently removed from the system, ensuring that no unnecessary files occupy storage space.

The cleanup service is implemented as a cron job, a time-based job scheduler in Unix-like operating systems. This cron job runs a script that scans the video directory and removes all files to free up space for new recordings. This automatic deletion process minimizes the need for manual maintenance and ensures the system remains functional and efficient without user intervention.

It is crucial to note that once deleted, these videos cannot be recovered, so users are advised to download any important footage before the scheduled cleanup time. This approach maintains the balance between efficient storage management and user accessibility, ensuring the system operates smoothly and reliably.

4.5.3 Power Management

To supply the system with continuous power, we propose two solutions:

- **Using a Power Bank:**

  This approach is straightforward to implement and use. It allows for the system to be plugged into a power source while recharging the battery simultaneously, ensuring uninterrupted operation even when unplugged. Using a power bank is a practical choice due to its ease of replacement, wide availability, and straightforward management.

- **Using a Li-Po Battery Connected to a PowerBoost 1000 Charger:**

  This option involves utilizing a Li-Po battery connected to a power boost such as the PowerBoost 1000 Charger. This method provides greater control over the system, enabling monitoring of battery levels and the addition of LED indicators to signal when the battery drops below 20%. However, this approach is more complex in terms of compatibility and is more costly compared to a simple power bank that integrates these features.

We decided on the power bank solution.The Raspberry Pi Zero consumes between approximately 140 mA when idle and 230 mA under load at 5V. The webcam draws 500 mA at 5V, while an LCD screen uses about 100 mA at 3.3V. The rotary encoder's power consumption is negligible and the USB hub has a static consumption of around 1 mA. Adding up the current draws—230 mA for the Raspberry Pi, 500 mA for the webcam, and 100 mA for the LCD screen—gives a total of 830 mA, which is roughly 0.9A at 5V. To ensure smooth operation, we will aim for a current of 1A at 5V, resulting in a total power consumption of 5W. With a 5000 mAh battery at 3.7V, which translates to 18.5 Wh, the estimated run time is 3.7 hours (18.5 Wh / 5 W). During testing, we used an old power bank from 2014 with a theoretical capacity of 8500 mAh, acknowledging that this

is not the actual capacity. The system operated for 3 hours and 30 minutes at a Full HD resolution to stress-test its limits. File transfer through the website took an additional half hour, after which the system ceased operation. Unfortunately, no further testing was conducted. A documented video of our testing is available on our YouTube page.

5 Mechanical Design

In order to improve the ergonomics of our system, we designed a box using SolidWorks to make it easier to use. The latter included designing the housing, interfaces and fixing system of the video capture unit to ensure that it works consistently and meets the needs of the user.

5.1 Box design

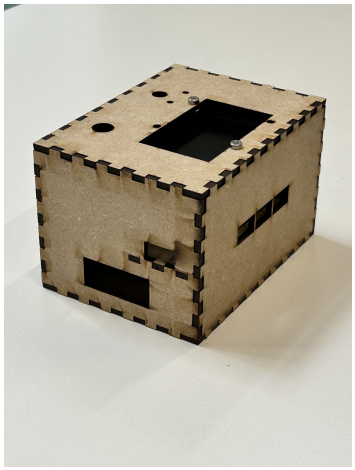

Figure 11: Box appearance

Figure 12: Internal structure

The box follows the design of the prototype, which is a rectangular prism. However, the prototype is too bulky and large, with dimensions of 300mm x 200mm x 100mm and a weight of 3kg. Therefore, we need a more compact and practical box to replace it. We aim to design a compact and sturdy box that can house all necessary electronic components and provide user-friendly interfaces.

Our initial design approach was to use a 3D printer to print the entire box, leaving one side open for assembling the electronic components. The open panel could be closed like a sliding door. After completing the first version, we found that it was difficult to assemble the electronic components due to the limited space inside the box. Therefore, printing each side of the box separately and then assembling them became a better solution.

We began designing such a box. To solve the connection problem between each panel, we used threaded inserts. Threaded inserts are often used for connecting 3D printed parts, they greatly improve the durability of the box. By heating the inserts with a soldering iron and embedding them into pre-drilled holes on the edges of the panels, the panels can be screwed together for assembly.

Later, using laser cutting to create the box proved to be a better choice because it is faster and more precise. In the final version, we designed and made a laser-cut box.

The main material used for the box is 3mm natural MDF. MDF is a common material for making boxes, easy to find in Fablab, very easy to process, and has good mechanical strength and usability. The box dimensions are 115mm x 85mm x 74mm. The internal structure of the box is shown in the diagram; it is divided into two layers. The middle layer serves as a single-chip computer bracket for fixing the Raspberry Pi Zero 2W. The bottom layer is for placing the external power supply. This design reduces the footprint of the box, ensures the center of gravity is not too high, and increases practicality.

5.2 Manufacturing and Assembly

The panels are manufactured using laser cutting technology, which greatly improves manufacturing efficiency compared to 3D printing. Once the panels that make up each side of the box have been cut, they can be assembled with finger joints and reinforced with woodworking glue. The finger joints measure 6mm x 3mm.

5.3 Interface design

5.3.1 Connectors and Power interfaces design

In order to ensure that the box can accommodate all the necessary electronic components and provide a user-friendly interface, the parameters of each opening have been designed in detail. These openings include screens, rotary decoders, connectors, and more. Below are the specific parameters of each opening:

Decoder hole :

- Location: Top of box
- Diameter: 7.3mm
- Shape: Round
- Quantity: 1
- Purpose: Used to operate the start, stop and other functions of the device.

Screen holes :

- Location: Top of the box
- Size: 58mm x 35.5mm
- Shape: Rectangular
- Quantity: 1
- Purpose: To hold the screen

USB Hole :

- Location: Front, side of box
- Size: 15mm x 9mm
- Shape: Rectangular
- Quantity: 5
- Purpose: for connecting external devices (e.g. webcam)

Camera hole :

- Location: top of the box
- Size: 10mm

- Shape: round
- Quantity: 1
- Purpose: for fixing the camera bracket

External power supply holes :

- Location: Side of the box
- Size: 35mm x 15mm
- Shape: Round
- Quantity: 1
- Purpose: for plugging in the power adapter

5.3.2 User interface design

To ensure a better user experience, we designed and produced a knob cap and a screen bracket. The knob cap is specifically designed for the rotary decoder. By fitting the cap onto the decoder, users can turn the knob more smoothly. The screen bracket effectively holds the screen in the opening at the top of the box, preventing it from becoming loose and enhancing the practicality of the system.

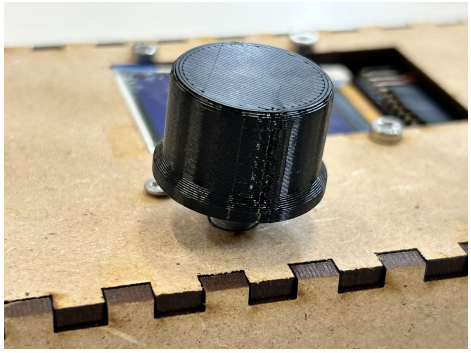

Figure 13: Knob cap

5.4 Modular hooking system design

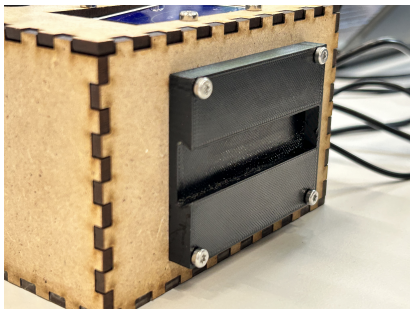

Figure 14: Dovetail mortise

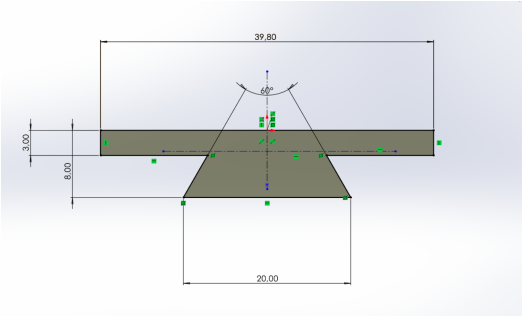

Figure 15: Dovetail tenon sketch

We adopted the dovetail joint as a hook system, which consists of two parts: the dovetail tenon and the dovetail mortise. The dovetail mortise can be fixed to the side of the box using four M3 screws, while the dovetail tenon can be attached to any surface. The dovetail tenon is highly secure and not prone to loosening, making it suitable for

scenarios where long-term hanging is required. Moreover, with our modular design, this system can be quickly assembled and disassembled, increasing flexibility.

The dovetail joint we designed has a head height of 3mm and a tail height of 5mm, with a width of 20mm and a slope angle of 60 degrees. It is made through 3D printing. Notably, to ensure the final product can be assembled smoothly after printing, we opted to add a 1mm fillet to the two side edges of the dovetail tail after multiple experiments. Additionally, to ensure that the dovetail can still pass through smoothly after the dovetail slot is fixed with screws, we made the slot width larger. The overall width and length of the dovetail slot are 54mm and 68mm, respectively, while the dovetail is 39.8mm and 66mm.

5.5 Webcam integration

To integrate the camera, we need to design a suitable camera mount for the box. One end of this mount should be fixed to the box, and the other end should hold the camera.

There are two options available. The first option is to design a 3D printed adjustable mechanical arm. The second option is to use a commonly available flexible phone holder provided by Fablab.

Initially, we considered creating a 3D printed adjustable mechanical arm. As a reference, we found a suitable 3D model online. This model weighs approximately 0.25 kilograms, with each arm having a working length of 150 millimeters, making the total length of three arms 450 millimeters. The mechanical arm has thumb screws at both ends, which can secure the box and the camera. However, considering the high time and material costs of designing and producing such a model, along with its limitations in flexibility and freedom of movement, we ultimately decided to use the existing phone holder.

The flexible phone holder provided by Fablab can rotate 360 degrees and has a 10mm screw and nut at the base for secure attachment. By making a corresponding hole on the top of the box, the holder can be easily installed and removed. Its total length is 25cm, and the top spring clamp can securely hold the network camera. This holder perfectly fits our system requirements.

5.6 Testing and Improvement

To test the adaptability of the box in the Fab Lab environment, we selected several usage scenarios. The first scenario is the most common: a workstation with a large desktop area. In this setting, we only need to place the box on the desktop to record. The second scenario is in the rooms with 3D printers and laser cutters. In this case, there is a lack of large work surfaces, so we need to place the box steadily on a chair in the workspace and adjust the camera angle to aim at the machines that need to be recorded. It is worth mentioning that in the 3D printer room, there are outlets below each machine, which means our capture device can work for extended periods, perfectly matching the long-duration nature of 3D printing.

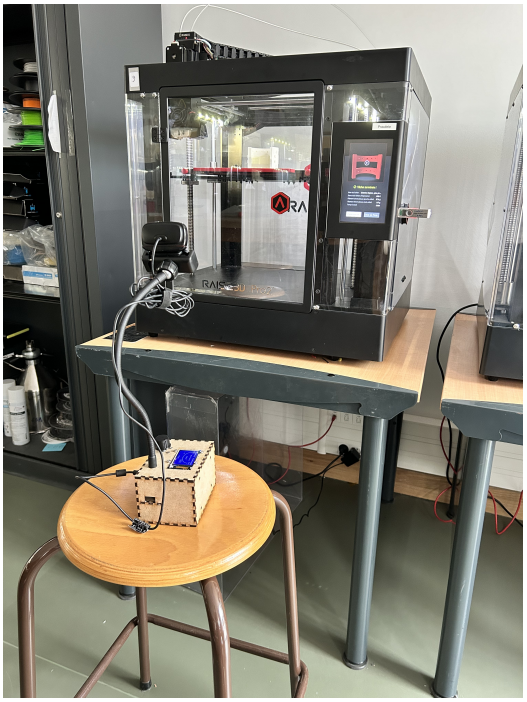

Figure 16: Recording scenario for 3D printers

Despite this, the box still has many areas that need improvement. For example, the finger joint design currently does not fit tightly together and requires the use of glue for reinforcement, making the assembly irreversible. As a result, to facilitate the smooth installation of electronic components during assembly, the top and side panels of the box were not glued together. This compromises the overall sturdiness of the box.

6 Development and Documentation

Given that this project is a documentation system, it was imperative to ensure thorough documentation of every step. We meticulously recorded all encountered problems, their solutions, and the development process leading to the final product. The adage "The first prototype always fails, the faster it is made the better" guided our approach. Rapid prototyping allowed us to quickly identify and address issues, laying a solid foundation for future iterations.

Multiple versions of the system were preserved to illustrate the evolution of the design.Detailed instructions on how to build the system, are provided in the Fablab Wiki. A dedicated YouTube page, which includes demos and testing videos were created with addition to a Git repository to store all related files.

7 Applications and Advantages and Limitations

The system's applications are diverse, making it suitable for various documentation needs in different environments. Its main advantages include its simplicity, robustness, and ease of use. The ergonomic design ensures it does not disrupt the workspace, while the straightforward video download process enhances user convenience.

Despite the success of our website solution, which operates smoothly, the absence of an external server for video uploads limits the system's capabilities. Running complex processes such as face detection or automated video labeling on the Raspberry Pi Zero 2W proved to be nearly impossible. While face blurring algorithms were successfully tested on our machines, we couldn't implement them on the Pi.

8 Conclusion

After six weeks of intense work, we successfully developed a compact and efficient documentation system tailored for the FabLab environment. The final product is a powerful, user-friendly device that is capable of capturing high-quality video documentation while minimally disrupting the workspace.

We started with a bulky prototype that, despite its functionality, proved unsuitable for regular use due to its size and weight. By iterating through multiple design versions, we transitioned to a smaller, more practical box using 3D printing and laser cutting techniques. This allowed us to fine-tune the balance between durability, ease of assembly, and functionality.

Our choice of hardware, specifically the Raspberry Pi Zero 2W, allowed us to maintain a compact form factor while ensuring sufficient processing power for video capture and basic image processing tasks. However, the lack of an external server for advanced video processing remained a limitation, highlighting an area for future improvement.

The testing phase in various FabLab scenarios demonstrated the adaptability and robustness of the system. Despite some initial challenges, such as ensuring the stability of the finger-joined connections, the final design proved to be reliable and effective in real-world applications.

In summary, this project not only resulted in a functional documentation system, but also provided valuable insights into rapid prototyping, iterative design, and the integration of various fabrication techniques. The knowledge and experience gained in this project will undoubtedly benefit future endeavors in similar fields.

9 References

[1] Clara Rigaud. Designing for Knowledge Capture in Fabrication Workshops. Human-Computer Inter-action [cs.HC]. Sorbonne Université, 2023. English. NNT :tel-04148164 link

[2] Raspberry Pi Zero 2W Documentation link

[3] FourCC link

[4] AVI vs. MP4: Which is the Best Video Format link

[5] How to Add a Power Button to Your Raspberry Pi link

[6] Difference between Interrupt and Polling link

[7] lgpio documentation link

[8] RPi.GPIO documentation link

[9] Failed to add edge detection - On Raspberrypi link