A Low-Cost, Off-Grid Camera-Based System for Water Level Monitoring

Development and Field Validation of a Proof-of-Concept for Remote River Monitoring using Edge Computing

AEC (Bjorn) Rens

Delft University of Technology

TU Delft

A Low-Cost, Off-Grid Camera-Based System for Water Level Monitoring

Development and Field Validation of a Proof-of-Concept for Remote River Monitoring using Edge Computing

by

AEC (Bjorn) Rens

to obtain the degree of Master of Science

at the Delft University of Technology,

to be defended publicly on Wednesday June 18, 2025 at 14:00 AM.

Student number:	5654254	
Project duration:	November 1, 2024 – June 18, 2025	
Thesis committee:	Prof. dr. ir. N.C. van de Giesen,	TU Delft, Water Resource Engineering
	Dr. R. Taormina,	TU Delft, Sanitary Engineering
	Dr. ir. R.W. Hut,	TU Delft, Water Resource Engineering
	Dr. H.C. Winsemius,	Rainbow Sensing

TU Delft

Preface

This thesis marks the end of an exciting and rewarding journey through my Master's degree in Environmental Engineering at the Delft University of Technology. The core inspiration for this project stemmed not only from my academic curiosity but also from my internship experience in Nepal—a resource-constrained environment where local engineers were incredibly driven yet faced significant barriers due to high equipment costs. Witnessing their challenges firsthand underscored the importance of creating accessible and sustainable monitoring technologies.

Developing this low-cost, off-grid, camera-based water-level monitoring system has significantly tested and expanded my technical skills, ranging from computer vision and embedded systems to practical field experimentation. Additionally, I developed essential hardware skills, such as soldering and hands-on tinkering with electronics.

I would like to sincerely thank my supervisor, Prof. dr. ir. Nick van de Giesen, not only for his guidance throughout the research but also for his support and good company during fieldwork. A huge thanks to Dr. ir. Rolf Hut for his creative insights and hands-on help with hardware tinkering, and to Dr. Hessel Winsemius for generously providing access to his OpenRiverCam platform and sharing his extensive experience. Special thanks also go to Dr. Riccardo Taormina, whose feedback, though he was less directly involved, significantly contributed to my research.

I am deeply grateful for the unwavering support from my girlfriend Daphne, who stood by me through the ups and downs of this academic journey. To my family, thank you for continually encouraging me as I pursued my academic goals. I am also immensely appreciative of the friendly and enjoyable environment created by everyone in the "afstudeer hok". Special thanks to Mannes, Christine, Karen, and Sophie for enduring the struggles with me.

I hope this work contributes to more accessible and effective water management solutions, sparking further innovation in environmental engineering.

AEC (Bjorn) Rens
Delft, June 2025

Abstract

Accurate water-level monitoring is vital for effective river management, flood prevention, and environmental conservation efforts. The increasing frequency and severity of flooding events, driven by climate change, highlight the critical need for reliable, robust, and sustainable water-level measurement systems. Such systems are particularly necessary in remote and resource-constrained settings where conventional infrastructure is lacking or insufficient.

Traditional water-level measurement methods, including radar, ultrasonic, pressure sensors, and conventional imaging systems, encounter significant limitations. These include high acquisition and maintenance costs, susceptibility to environmental damage, vandalism risks due to conspicuous placement, and dependence on stable, continuous power sources. Consequently, there remains an unmet demand for affordable, resilient, and autonomous monitoring devices capable of functioning reliably in challenging and off-grid environments.

To address this gap, this study presents the development and validation of an innovative water-level monitoring prototype as a prove of concept. The proposed system integrates a low-cost Raspberry Pi-based imaging sensor equipped with infrared illumination to enable accurate measurements both during the day and at night. An onboard processing unit autonomously captures and analyses images in real-time using one of two distinct image-processing algorithms: the mean-difference method and the Kolmogorov–Smirnov (KS) test method. These algorithms quantify water levels by detecting pixel-intensity contrasts, thus eliminating the need for direct physical contact with the water body.

The prototype underwent field validation in a natural river environment, demonstrating robust and consistent performance. The system achieved an accuracy within ±4.6 cm at a 95% confidence interval. Notably, it exhibited stable performance under varying environmental conditions, with moderate bias differences between daytime and nighttime scenarios, and minimal sensitivity to precipitation effects. The mean-difference algorithm demonstrated superior precision, while the KS-test method offered enhanced robustness against environmental variability.

This research underscores the practical feasibility and significant potential of low-cost, autonomous camera-based systems for sustainable water-level monitoring. Such solutions can substantially improve disaster preparedness and environmental management, especially in remote or economically disadvantaged regions lacking traditional monitoring infrastructure. Future research directions include integrating renewable energy solutions such as solar power to enhance operational autonomy, exploring advanced image-processing algorithms for increased accuracy, and conducting extended validations across a broader range of hydrological and environmental scenarios.

Keywords: *Water-level monitoring; Autonomous sensors; Computer vision; Raspberry Pi; Off-grid operation; Flood management; Low-cost hardware; Open-source hardware*

Contents

Preface i

Summary ii

Nomenclature viii

1 Hardware in context 1
- 1.1 State of the art . . . 1
- 1.2 Research gap and objectives . . . 2
- 1.3 Introducing the proof of concept . . . 2

2 Hardware description 3
- 2.1 Imaging module . . . 3
- 2.2 Illumination system . . . 3
- 2.3 Onboard processing unit . . . 4
- 2.4 Data storage and communication . . . 4
- 2.5 Power supply and energy management . . . 4
- 2.6 Enclosure and mounting assembly . . . 4
- 2.7 Reference object for waterline detection . . . 5

3 Design files summary 8

4 Bill of materials summary 10

5 Build instructions 11
- 5.1 Hardware . . . 11
  - 5.1.1 Preparing power supply and PCB . . . 13
  - 5.1.2 Mounting mechanism & modular design . . . 14
  - 5.1.3 Installing camera module . . . 15
  - 5.1.4 Installing IR-illumination module . . . 16
  - 5.1.5 Device assembly . . . 17
  - 5.1.6 Reference object . . . 18
- 5.2 Software . . . 19
  - 5.2.1 System setup summary . . . 19
  - 5.2.2 Remote access and data retrieval . . . 20
- 5.3 Water level detection algorithm . . . 20
  - 5.3.1 Image pre-processing . . . 20
  - 5.3.2 Mean-difference method . . . 21
  - 5.3.3 KS-test method . . . 22

6 Operation instructions 24
- 6.1 Setup and orientation . . . 24
- 6.2 Calibration and verification . . . 25
- 6.3 Operation . . . 25

7 Validation and characterization 26
- 7.1 Experimental setup . . . 26
- 7.2 Data acquisition . . . 27
- 7.3 Results . . . 28
  - 7.3.1 Raw data and outliers . . . 28
  - 7.3.2 Diurnal patterns . . . 29
  - 7.3.3 Influence of precipitation . . . 30
  - 7.3.4 Performance statistics . . . 31

7.4 Power assessment . . . 33

8 Discussion **35**
8.1 Interpretation of field data . . . 35
8.2 Comparison to expectations & literature . . . 36
8.2.1 Comparison to traditional sensors . . . 36
8.2.2 Expected diurnal influence versus observed performance . . . 36
8.2.3 Expected influence of precipitation versus observed performance . . . 37
8.3 Limitations . . . 37

9 Conclusion **40**
9.1 Key findings . . . 40
9.2 Practical implications . . . 41

10 Future work **42**

References **43**

A Design Methodology **46**
A.1 Design Perspective . . . 46
A.2 Function Analysis . . . 47
A.3 Basic Design Cycle . . . 49
A.4 List of Requirements . . . 50
A.5 Morphological Chart . . . 52
A.6 SCAMPER Method . . . 53

B Design choices **54**
B.1 Operating system Raspberry Pi . . . 54
B.2 Reference object . . . 54
B.3 Algorithm development . . . 55
B.4 Hardware development . . . 56

C Criteria evaluation **57**
C.1 Measurement Performance . . . 57
C.2 Operational Aspects . . . 58
C.2.1 Summary of Performance . . . 59

D System Setup Manual **60**
D.1 Operating System . . . 60
D.2 First-Boot Configuration: Swapfile and Updates . . . 61
D.3 Package Installation . . . 62
D.4 Auto-Start Service . . . 63

E Remote access manual **64**
E.1 Remote access GUI mode; RealVNC . . . 64
E.2 Remote access CLI mode; SSH & PuTTY . . . 65
E.3 SFTP file transfer via WinSCP . . . 68

F Operation and calibration manual **69**
F.1 Calibration . . . 69
F.1.1 Device setup checklist . . . 70
F.1.2 Disable device checklist . . . 71
F.2 Operation . . . 72

G Development LOG **73**

List of Figures

2.1 Conceptual system architecture of the image-based water-level monitoring prototype. . 5
2.2 Hardware components used in the water-level monitoring system. For more detailed information on hardware components, see section 4.1 6
2.3 Final system assembly and reference object deployment. 7

3.1 Overview of Python file structure with descriptions. 9

5.1 Hardware schematic of the camera-based water-level monitoring system. 12
5.2 Overview of the soldered components, including a 1.2Ω, 5W resistor and MOSFET driver integrated on a 60×90mm PCB, with output leads for IR LED connection. 13
5.3 Assembly and mounting of the system enclosure. 14
5.4 Integration of the camera module into the system enclosure. 15
5.5 Integration of the IR LED module with heatsink, enclosure, and diffuser. 16
5.6 Final assembly and integration of power, processing, and connectivity components. . . 17
5.7 Construction and installation of the modular reference board for waterline detection. . . 18
5.8 Preprocessing workflow for image-based waterline detection. 20
5.9 Illustration of the circular nature of the Hue channel [47] 21
5.10 Schematic overview of the visual waterline detection process. 21
5.11 Algorithm results displaying the most probable waterline location (red dashed line) for each image channel. 22
5.12 Visual outputs for quality control and verification of the waterline detection process. . . . 22
5.13 Illustration of the Kolmogorov–Smirnov (KS) statistic.[7] 23
5.14 KS-test detection process in the HSV channel. 23

6.1 Schematic overview of the deployment setup 24

7.1 Field deployment location (50.80762°N, 5.91350°E). 26
7.2 Field deployment of the water-level monitoring system at the Kleine Geul. 27
7.3 Time series of detection error (cm) for both the mean-difference and KS-test methods across the 42-hour field deployment. 29
7.4 Detection error over time for the mean-difference method. A 24-hour sinusoidal curve (red line) highlights the diurnal bias, with a clear positive shift during daylight hours. The mean bias across the deployment was 1.57cm (orange dashed line). The green dashed line represents the zero-error reference level. Day (white) and night (grey) periods are shaded accordingly. 30
7.5 Detection error over time for the KS-test method. A sinusoidal fit (red line) is shown for comparison, but no clear diurnal pattern is observed. The average bias was 1.02 cm (orange dashed line), with relatively stable error across day and night intervals. Shaded regions distinguish day (white) and night (grey) conditions. 30
7.6 Detection errors over time using the mean-difference method, with precipitation periods shaded in blue. 31
7.7 Detection errors over time using the KS-test method, with rain periods indicated in blue. 31

A.1 Function analysis 48
A.2 Application of the Basic Design Cycle in this project. 49
A.3 Morphological chart 52

B.1 Design considerations reference object. 55

D.1 Overview of the SD-card imaging process. 60

D.2 Swapfile configuration. 61
D.3 Example of .service file 63

E.1 Enable SSH and VNC settings in RPI environment. 64
E.2 Setting up a remote connection using RealVNC. 65
E.3 Set interfacing options. 65
E.4 Configure Wireless LAN connection. 66
E.5 Location of IP-address 66
E.6 Insert Host name/ IP-address 67
E.7 Login interface for PuTTY 67
E.8 Setup file transfer configuration. 68

F.1 Switching RPI boot-up configuration 70

List of Tables

3.1 Inventory of design files and their locations 8

4.1 Bill of materials for the prototype 10

5.1 Logged detection output per image cycle. 22

7.1 Summary of accuracy and precision metrics for both detection methods over 270 field measurements. 32
7.2 Comparison of water-level detection performance during daytime and nighttime conditions for both the mean-difference and KS-test method. 32
7.3 Summary of detection performance under dry and rainy conditions for both the mean-difference and KS-test method. 33
7.4 Average processing characteristics per cycle. 33
7.5 Estimated average power consumption over 1 hour of operation. 33

A.1 List of prioritized system requirements based on the MoSCoW method. 51

C.1 Summary of requirement evaluation with MoSCoW classification and performance status. 59

Nomenclature

Abbreviations

Abbreviation	Definition
CLI	Command Line Interface
CSI	Camera Serial Interface
GND	Ground
GUI	Graphical User Interface
IR	Infrared
KS	Kolmogorov–Smirnov
LED	Light Emitting Diode
MAE	Mean Absolute Error
OS	Operating System
POC	Proof of Concept
RPI	Raspberry Pi
RGB	Red Green Blue (color model)
RMSE	Root Mean Squared Error
ROI	Region of Interest
SFTP	Secure File Transfer Protocol
SSH	Secure Shell
VNC	Virtual Network Computing

Symbols

Symbol	Definition	Unit
N	Number of samples	[–]
μ	Mean	[varies]
σ	Standard deviation	[varies]

1

Hardware in context

Collecting accurate water level data is crucial for efficient river management, enabling precise calculation of river discharge when combined with velocity and cross-sectional data. Real-time water level monitoring systems play an essential role in disaster prevention by facilitating flood early-warning systems, which are critical in protecting lives and property [22, 18]. With the intensification of climate change, the probability of short-term heavy rainfall events has increased significantly, leading to more frequent and severe flooding. Reliable water level monitoring has become increasingly important for proactive disaster management and environmental protection.

1.1. State of the art

Current technologies for water level measurements typically fall into two categories: contact and non-contact sensors. Contact sensors, including pressure and floater sensors, are relatively low-cost and straightforward to operate, but are vulnerable to corrosion and flood-related debris [45].

Floater sensors usually only detect whether the water level is crossing a predefined threshold, whereas pressure sensors are more robust and able to store exact water level data. Floater sensors are therefore mostly used in the industrial sector to monitor tank levels and are budgeted around ~€100 [31]. Pressure sensors are commonly used for water level monitoring and are often placed in combination with a shaft to protect the device. Low-cost versions can be acquired from around ~€300 with an accuracy of ±1 cm [30].

Non-contact sensors such as radar [27] and ultrasonic sensors [5] offer protection from high water velocities and floating debris. Ultrasonic sensors are available in a wide cost range, but low-cost versions start around ~€100 with an accuracy of ±1 cm [23]. However, these sensors are sensitive to environmental factors like wind and vibrations. Radar is significantly more expensive, starting from ~€1000, but provides higher accuracy of ±2 mm [41] and is more stable under varying conditions such as fog and rain. Both types, however, are prone to vandalism due to their visibility on bridges or infrastructures directly above river channels.

Imaging technologies provide an alternative approach for water level monitoring and include three main methods: satellite imagery [19, 35, 36], images of water gauges [21, 20], and images directly capturing water level lines [33, 11, 13]. Advances in deep learning have improved automation in detecting water levels, using object recognition and semantic segmentation [26, 6, 8, 34, 12, 17, 39, 40, 15, 48].

However, these techniques often require the installation of water gauges and illumination systems for nighttime capturing which are challenging to maintain sustainably, especially in remote or resource-limited settings. Additionally, they often depend on a reliable power connection and a high-performance processing unit, increasing both cost and operational demands.

1.2. Research gap and objectives

There is a growing need for affordable, compact, sustainable, and off-grid water level monitoring systems that are suitable for remote regions and capable of operating both during the day and at night. While existing methods using floaters or pressure sensors are relatively low-cost, they are vulnerable to damage under flood conditions. Radar, ultrasonic, and advanced camera-based systems depend on stable power and are prone to vandalism.

1.3. Introducing the proof of concept

This paper presents a proof of concept (POC) for a compact, autonomous, and off-grid water-level monitoring system, tailored for use in remote or resource-limited environments. The system is energy-efficient, cost-effective, and designed to be less susceptible to vandalism and flood damage.

It integrates a low-cost imaging sensor with an onboard processor, housed in a discreet mounting setup to minimize exposure. Water level detection is achieved via visual sensing, leveraging optical differences, such as brightness and texture, between the water and its surroundings. This eliminates the need for contact-based measurement components.

The system was developed based on four core design criteria:

- **Flood & vandalism protection:** Low vulnerability to extreme water levels and vandalism.
- **Availability & continuity:** Water level data must be available within 5 minutes of capture; at least one detection every 10 minutes.
- **Cost-effectiveness:** Total material costs should not exceed €400.
- **All-condition operation:** Reliable detection both day and night, regardless of ambient light.

In this paper, we present the final product resulting from multiple design choices and iterative development. This is structured in the style of the *HardwareX* journal, with an emphasis on reproducibility, aiming to provide a solid foundation for potential future developments of this POC.

The following chapters are structured as follows: Chapter 2 provides a hardware description, listing all components and highlighting key design decisions. Chapter 3 presents a design file summary. Chapter 4 contains a bill of materials. Chapter 5 describes hardware and software build instructions and the main algorithm. Chapter 6 outlines operation procedures, Chapter 7 presents the field setup and results, Chapter 8 contains the discussion, and Chapter 9 concludes the findings. Fianally, chapter 10 outlines potential future work.

To ensure reproducibility and to illustrate the structured nature of the development process, the annex provides detailed documentation of each step. Annex A outlines the theoretical design methodology. Annex B presents the rationale behind key design decisions. Annex C evaluates the final system against the predefined design criteria. Annexes D, E, and F together form a user manual, offering step-by-step instructions for system setup, remote access, operation, and calibration. Finally, a development log is included in Annex G, documenting implementation notes and key design considerations.

2

Hardware description

The system consists of several components that collectively meet the criteria introduced in Chapter 1. The imaging module captures images of the waterline to be detected. A supporting illumination system, using infrared lighting, enables image capture during nighttime conditions. An onboard processing unit analyses the images and determines the water level. Once the water level is identified, the data is stored locally and transmitted to an online platform. The system is powered by a battery pack and housed within an enclosure to protect it from environmental influences. A simple mounting mechanism allows for easy short-term deployment. Finally, a reference object is used to enhance the accuracy of the water level detection. These components are described in more detail below.

A conceptual schematic of the system is shown in Figure 2.1, while Figure 2.2 and 2.3 displays the individual components used. Additionally, Figure 5.1 in Chapter 5 presents the hardware schematic, showing the full electrical layout of the system. A comprehensive overview of the design methodology (Annex A), component selection (Annex B), and criteria evaluation (Annex C) is provided in the Appendix.

2.1. Imaging module

The Raspberry Pi Camera Module V2 NoIR (Figure 2.2a) was selected for its Sony IMX219 sensor, which delivers 8 MP stills at up to 3280 × 2464 and supports 1080p30, 720p60, and 640 × 480 p60/90 video modes [28]. The sensor's 1.12 µm × 1.12 µm pixel pitch and 3.68 × 2.76 mm imaging area (¼″ optical format) offer high sensitivity and fine spatial resolution [10]. As the NoIR variant, this camera module omits the infrared (IR) cut filter, allowing near-infrared wavelengths to reach the sensor. This extends the detectable wavelength range from approximately 400–700 nm (visible spectrum) to about 400–1000 nm. The extended range enables reliable low-light and nighttime imaging, particularly when used with an IR LED illuminator [28].

The camera module is mounted flush in the enclosure lid to maintain a fixed, repeatable viewing angle. A polarization filter (Figure 2.2h) is affixed to the outside of the lid directly over the lens opening. This both seals the camera hole against dust and moisture and reduces specular reflections from the water surface, improving contrast and lowering noise in the captured images. The camera attaches via the MIPI CSI-2 interface and is driven using libcamera through the Picamera2 Python library [28].

2.2. Illumination system

The "ILH-IN01-85SL-SC211-WIR200" infrared LED emitter module (Figure 2.2b) is used to enable nighttime imaging at a wavelength of 850 nm, which falls well within the detectable range of the camera sensor. While both 850 nm and 940 nm modules are commonly used, the 850 nm option delivers higher radiant output per watt and longer effective range, whereas 940 nm LEDs are invisible to the naked eye and better suited for stealth applications. Since image quality has priority, the 850 nm module was selected. It operates at a 3.4 V forward voltage, a 5 V reverse voltage, and a maximum current of 1.5 A, with a 50° viewing angle and a radiant intensity of 1.23 W/sr [14]. The IR-LED control circuit was

implemented using a 1.2 Ω, 5 W resistor in series with an Adafruit MOSFET driver (Figure 2.2e). The MOSFET is controlled from a GPIO pin on the processing unit (Raspberry Pi Zero 2 W), so that the IR LED is powered only during image capture. Both the resistor and MOSFET are soldered onto a custom 60 × 90 mm PCB for compact integration. Building instructions are presented in Chapter 5.1

The LED is mounted in the enclosure lid, 65 mm from the camera lens and aligned using the same orientation. It is bonded to a heatsink (Figure 2.2c) with thermal interface fluid to ensure effective heat dissipation. Additionally, a diffusive plastic filter (Figure 2.2g) is installed over the IR emitter hole in the enclosure lid. This piece both seals the opening against dust and moisture and scatters the 850 nm illumination into a uniform field, reducing hot-spot glare and reflections that can introduce noise into the waterline detection.

2.3. Onboard processing unit

The Raspberry Pi Zero 2 W (Figure 2.2d) was selected for the processing unit because its 1 GHz quad-core ARM Cortex-A53 CPU and 512 MB of LPDDR2 SDRAM are sufficiently powerful to support real-time image analysis [29]. It includes onboard 2.4 GHz 802.11 b/g/n wireless LAN allowing wireless data transfer [29]. The 65 mm × 30 mm board features a CSI-2 camera connector for direct attachment of compact camera modules and runs Raspberry Pi OS with native Python support [29]. At 5 V the board consumes about 100 mA at idle and peaks at 449 mA under full load [3], making it compatible with small battery packs for reliable off-grid operation. While more energy-efficient alternatives such as the AMB82-Mini (RTL8735B) offer lower idle currents and deep-sleep modes [2], the extensive software ecosystem, community support, and compatibility with Python-based workflows made the Raspberry Pi Zero 2 W a practical and robust choice for rapid prototyping and algorithm development.

2.4. Data storage and communication

For basic data storage and operation, the Raspberry Pi Zero 2 W uses a 32 GB microSD card. However, because microSD cards can suffer failures in environments with wide temperature swings, a USB flash drive connected via a micro-USB-to-USB-A adapter cable is added. This drive serves as a redundant backup for both raw images and processed output. In addition to local storage, detected water levels are posted to the OpenRiverCam server [25] using API POST requests with an access token. These posts are scripted and integrated into the water level detection algorithm, enabling the system to upload its measurements automatically during each capture cycle. Internet connectivity is provided via the onboard Wi-Fi module of the Raspberry Pi Zero 2 W. During desk testing, the device connected through a local Wi-Fi network; in the field, a mobile hotspot was used to establish the connection.

2.5. Power supply and energy management

The power system centres on a Voltaic Systems V50 battery pack (13,400 mAh, 48.24 Wh Li-Ion) that supports “Always On” mode and is optimized for solar recharging (Figure 2.2f). Its internal charge controller maintains a maximum power point of approximately 5.2 V and accepts up to 2 A of solar input, potentially allowing direct connection of a small solar panel via USB. At 5 V, the V50 draws about 7 mA in ”Always On” standby and can deliver up to 4 A across two USB-A ports.

The system’s peak draw ($\sim$500 mA at 5 V for the Pi Zero 2 W and camera) yields about 26 h of autonomy on battery alone. Though not explored in this research, pairing the battery pack with a modest solar panel (such as a 6 W, 6 V panel [43] providing $\sim$1 A in peak sun) could enable continuous daylight recharging and sustain true autonomous off-grid operation. The 114 × 78 × 26 mm unit mounts securely inside the enclosure, and its dual USB-A outputs feed the Pi and illumination module.

2.6. Enclosure and mounting assembly

A Schneider Electric Thalassa TBS casing (Figure 2.2i) was used as weather-resistant enclosure, with seals retained everywhere except at the IR-LED and camera-lens openings, where holes were drilled into the lid. To maintain modularity, a double-floor platform made from a PVC cutout lined with velcro tape was installed. On the back of the enclosure, a custom hinge assembly fabricated from steel L-profiles and plywood enables straightforward short term deployments without compromising the en-

closure's integrity.

2.7. Reference object for waterline detection

To increase the effectiveness of the water level detection system, a uniform background is preferred. This can be achieved in various ways, depending on site-specific conditions. For example, white paint could be applied to a stable structure located in the river, such as a bridge pillar or rock formation. For the field testing conducted in this research, a reference object was constructed with a modular design, allowing for easy installation and removal.

To create contrast between water and air, a 4 mm matte-white foam PVC sheet (600 × 500 mm) was mounted onto 15 mm plywood. Two 25 mm-diameter rods secured to the plywood combined with tension straps allow stable depth and height adjustment (Figure 2.3d-2.3e). The PVC's high diffuse reflectance makes waterlines more detectable compared to the noisy natural background.

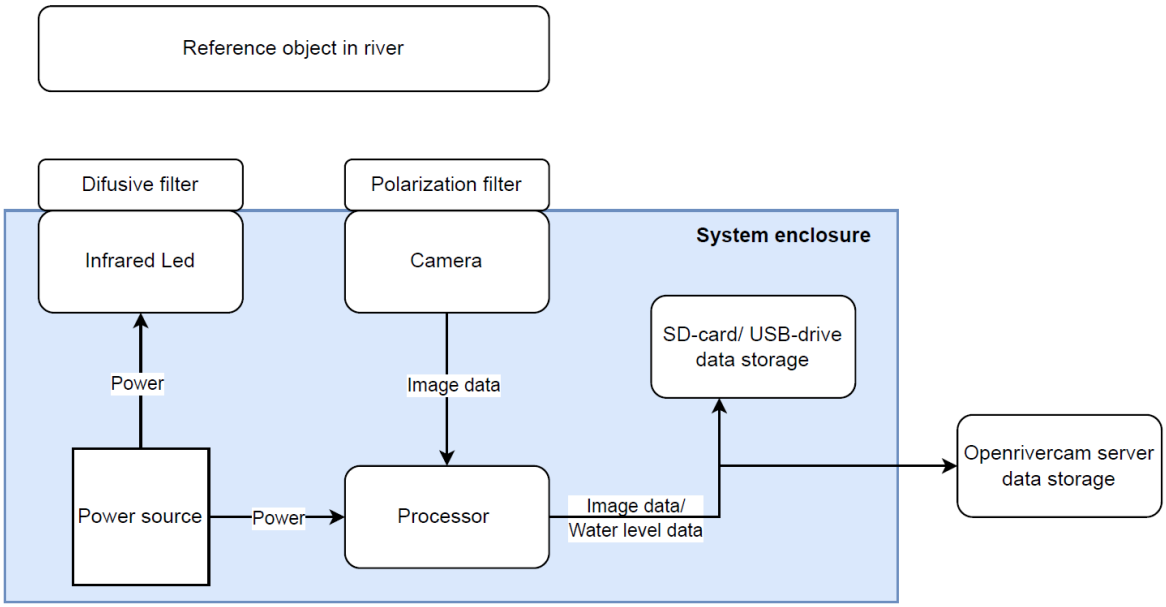
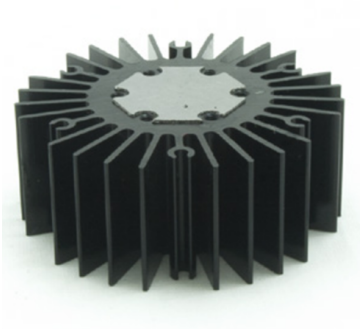


Figure 2.1: Conceptual system architecture of the image-based water-level monitoring prototype.

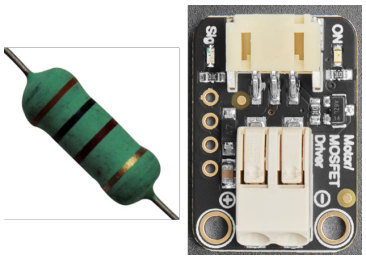
(a) PRI NoIR camera module [28]

(b) IR emitter module [14]



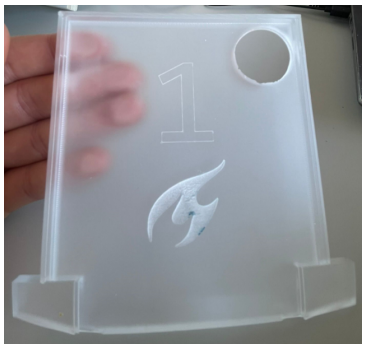
(c) LED heatsink [16]

(d) Raspberry Pi Zero 2 W [29]



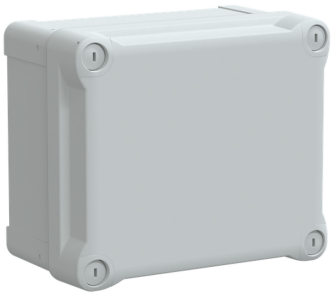
(e) Resistor [24] & MOSFET driver [9]

(f) V50 battery pack [44]



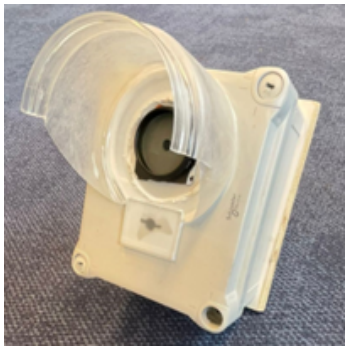
(g) Diffusive filter

(h) Circular polarisation filter [37]

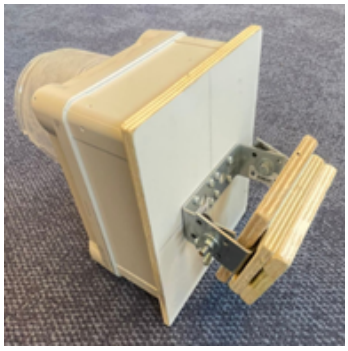


(i) Schneider electric enclosure [32]

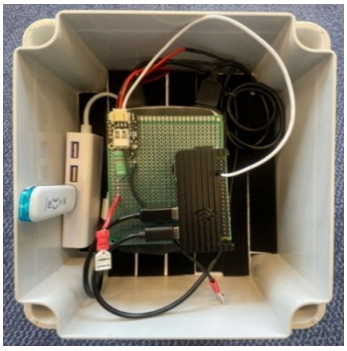
Figure 2.2: Hardware components used in the water-level monitoring system. For more detailed information on hardware components, see section 4.1



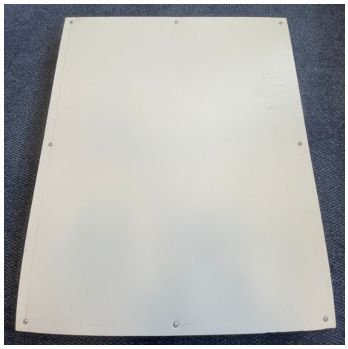
(a) Front view final assembly



(b) Rear view final assembly



(c) Inside view final assembly



(d) Front view reference object

(e) Rear view reference object

(f) Deployment of reference object in river

Figure 2.3: Final system assembly and reference object deployment.

3

Design files summary

All files can be found in the online repository: *https://github.com/AEC-Rens/Water-level-monitoring-system-repository*, as well as in the annex. A hardware schematic file (PDF) is included, showing how the camera module, power supply, and illumination module connect to the Raspberry Pi as shown in Figure 5.1. All components are ready-to-use, and build instructions are provided in Chapter 5.

A system setup manual offers instructions for initial use of the Raspberry Pi environment (annex D). The remote access manual guides the user through accessing the operating system and transferring data between systems (annex E). Instructions on how to calibrate and operate the system during deployment are provided in the operation and calibration manual (annex F).

Finally, a complete image file of the operating system is included, containing all required packages, software installations, calibration procedures, and the Python script, organized within the correct directory structure. Additionally, the experimental data, including captured images and algorithm outputs, are also available in the repository through a publicly accessible Google Drive link.

Table 3.1: Inventory of design files and their locations

Design filename	File type	Description	Location of file
Hardware_schematic	PDF	Schematic showing connection of electrical components	GitHub link
System_setup_manual	PDF	Instructions on how to setup the operating system for first use	GitHub link
Remote_access_manual	PDF	Instructions on how to communicate with the system	GitHub link
Operation_and_calibration_manual	PDF	Instructions on how to perform calibration and operate the system	GitHub link
SD_card_backup_10042025	`.img`	Backup of complete operating system and installed packages/scripts	GitHub link
Data_and_resources	`.md`	Link to experimental data, captured images, and algorithm outputs	GitHub link
Scripts	Folder	All Python scripts used in water-level detection algorithm	GitHub link

The file structure of the water level detection algorithm (which can be found in the GitHub repository) is clarified in Figure 3.1

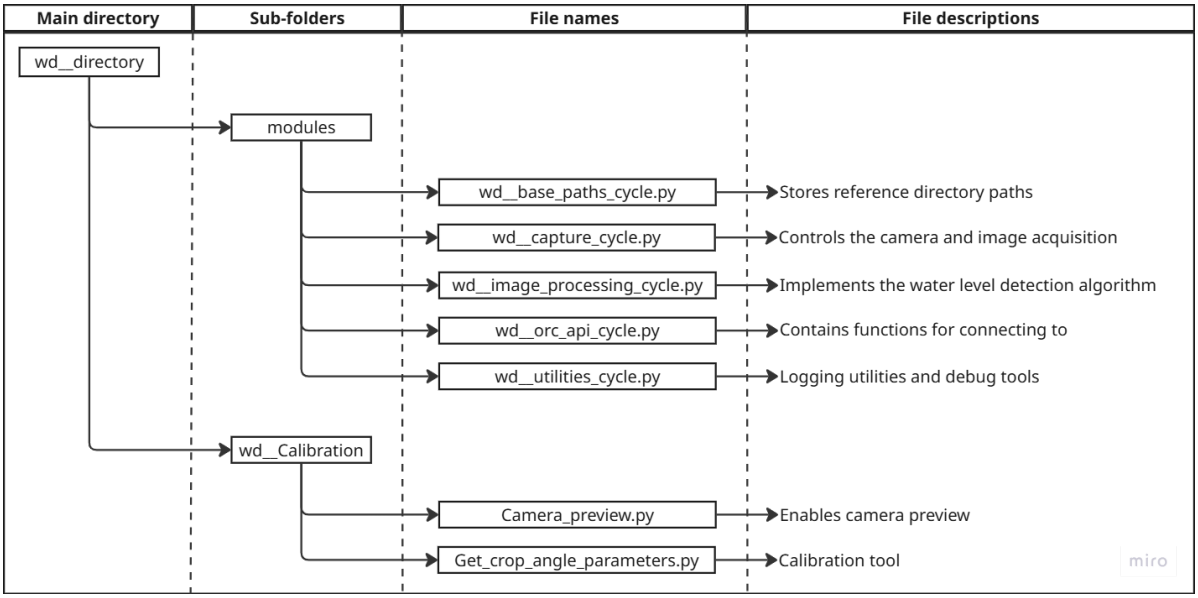


Figure 3.1: Overview of Python file structure with descriptions.

4

Bill of materials summary

The bill of materials provides a detailed list of all required components. From the Raspberry Pi Zero 2 W starter kit and Pi Camera Module V2 NoIR to the ILH-IN01 IR emitter, custom PCB, and Thalassa TBS enclosure. Each annotated with its function, supplier, quantity, total cost, and source. In total, 21 line items sum to €289.20, giving an exact financial overview for budgeting and reproducibility.

Table 4.1: Bill of materials for the prototype

#	Component	Function	Supplier	Cost (€)	Source
A	Raspberry Pi Zero 2 W – starter kit	Central processing and control unit	RaspberryPi	€44.95	raspberrystore.nl
B	Raspberry Pi camera V2 NoIR	Image capture module	RaspberryPi	€17.95	raspberrystore.nl
C	Camera Cable – 200mm	CSI interface cable	RaspberryPi	€1.15	kiwi-electronics.com
D	Camera housing V3/Arducam	Camera protection	Arducam	€5.99	raspberrystore.nl
E	Aluminium case for RPi Zero	RPi protection	Waveshare	€5.92	kiwi-electronics.com
F	MOSFET Driver JST-PH	Power control	Adafruit	€4.59	kiwi-electronics.com
G	JST PH Cable – 200mm	IR LED cable	Adafruit	€1.44	kiwi-electronics.com
H	V50 USB Battery Pack	Portable power	Voltaic Systems	€83.48	kiwi-electronics.com
I	USB Micro B to USB-A cable	Power/data cable	Onlinekabelshop	€2.49	onlinekabelshop.nl
J	1.2Ω Resistor, 5 W	Current limiter	Multicomp Pro	€0.59	farnell.com
K	IR Emitter Module, 850 nm	Night illumination	ILS	€9.78	farnell.com
L	LED Star Heatsink	LED cooling	ILS	€12.48	farnell.com
M	USB-A (m) open wire	Power wiring	Onlinekabelshop	€2.99	onlinekabelshop.nl
N	Prototyping Board – 4x6cm	Circuit platform	Kiwi Electronics	€1.68	kiwi-electronics.com
O	Velcro tape 20mm	Mounting tool	Onlinekabelshop	€5.99	onlinekabelshop.nl
P	Thalassa TBS Casing	Waterproof case	Schneider Electric	€19.70	se.com
Q	Silicone Wire – 2m 26AWG	Low-voltage wire	Kiwi Electronics	€1.20	kiwi-electronics.com
R	USB 2.0 Flash Drive 16GB	Backup device	Philips	€11.50	123accu.nl
S	Mounting hardware (misc.)	System install	–	€20.00	Hardware Store
T	Circular polarized filter	Optical filter	UwCamera	€19.95	uwcamera.nl
U	Foamed PVC board (4mm)	Waterline reference	kunststofplatenshop	€15.38	kunststofplatenshop.nl
			Total	**€289.20**	

5

Build instructions

This chapter details the end-to-end assembly of the water-level monitoring system. In brief, first, the hardware components; power and control PCB, processing unit, camera module, IR illuminator and filters are mounted and interconnected within the weather-proof enclosure (Section 5.1). Next, the software environment is prepared: the operating system is imaged onto the SD card, system settings and packages are configured, the detection scripts are deployed, remote-access is enabled, and the main cycle is registered as a boot-up service (Section 5.2). These instructions ensure that both the physical build and the software setup can be reproduced. Finally, the water level detection algorithm is elaborated (Section 5.3). We provide these steps in more detail below.

5.1. Hardware

The following subsections describe the step-by-step assembly and wiring of all hardware components within the weather-proof enclosure. A hardware schematic, showing all electrical components and connections is displayed in Figure 5.1.

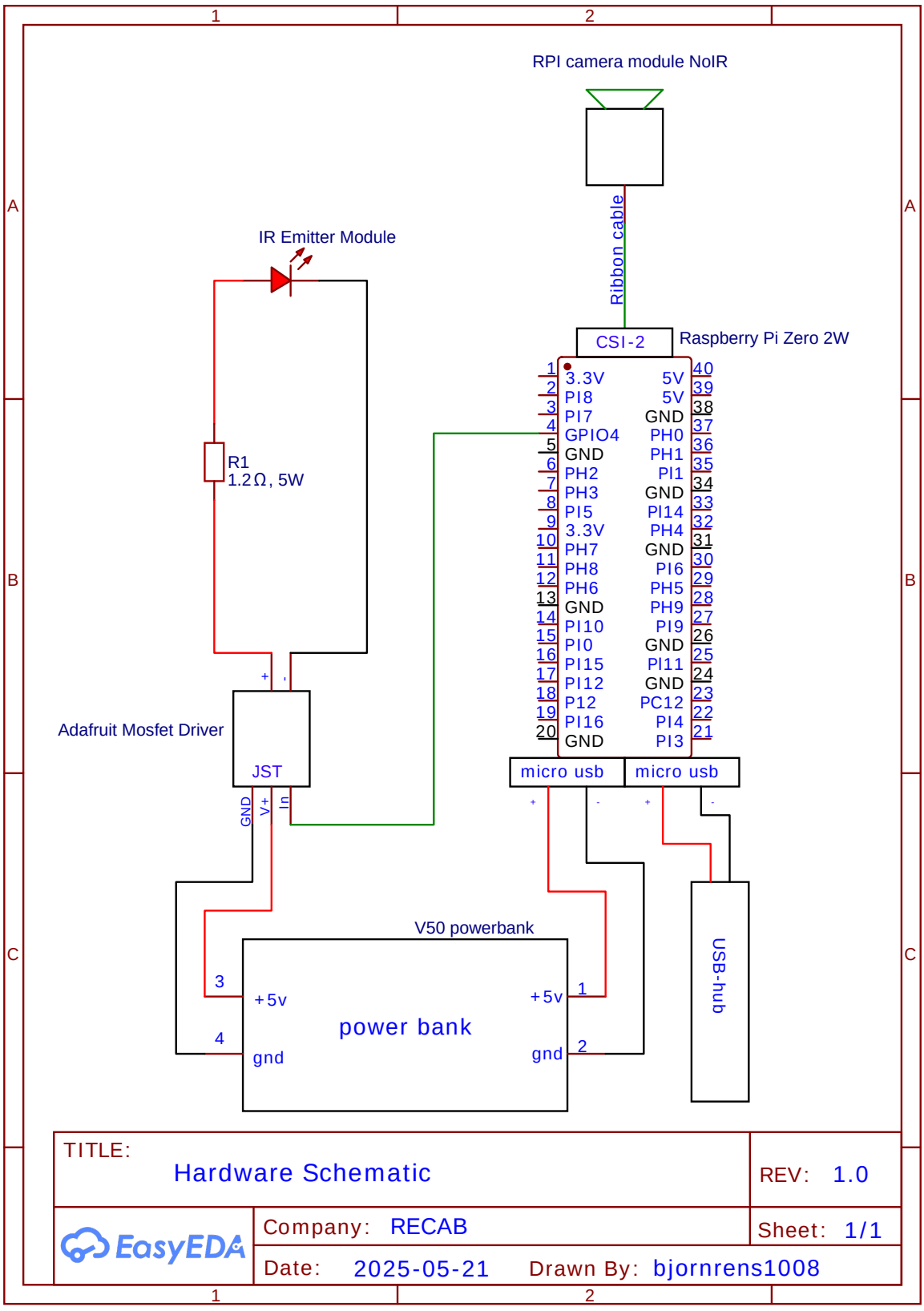


Figure 5.1: Hardware schematic of the camera-based water-level monitoring system.

5.1.1. Preparing power supply and PCB

To enable GPIO-controlled IR illumination and modular power connections, a MOSFET driver and a 1.2 Ω, 5 W resistor were soldered to a 60 × 90 mm prototyping PCB, together with power (+) and ground (GND) leads. Cable lugs were crimped onto the external wire ends using a crimping tool to allow secure, removable connection of the IR LED. The JST connector cable was modified by stripping and soldering its power and ground wires to the corresponding conductors of a USB-A power cable, permitting powering via any 5 V USB source. Heat-shrink tubing was applied over each joint for electrical insulation (Figure 5.2).

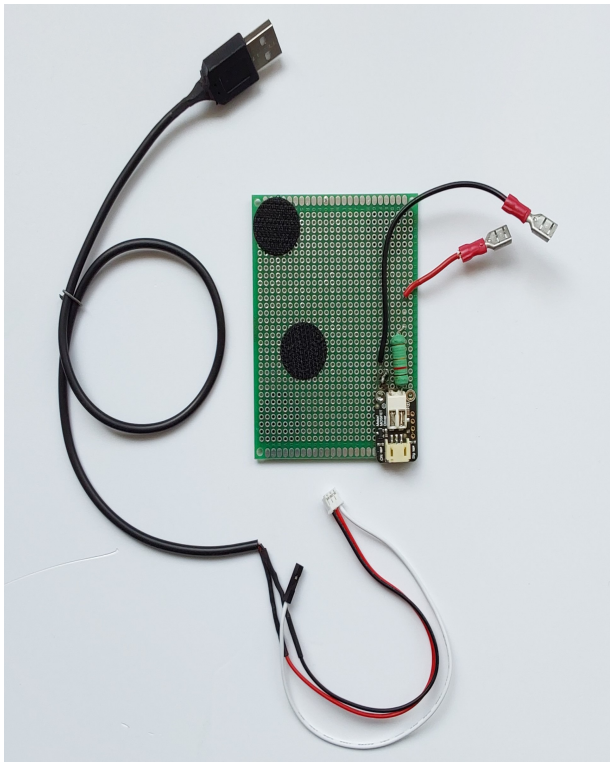


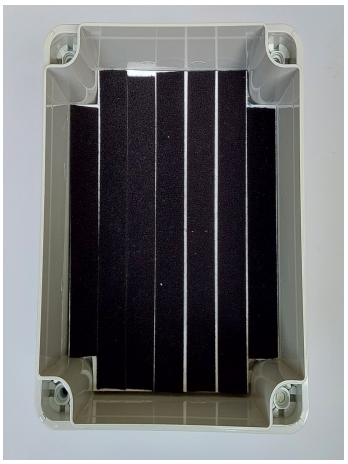
Figure 5.2: Overview of the soldered components, including a 1.2Ω, 5W resistor and MOSFET driver integrated on a 60×90mm PCB, with output leads for IR LED connection.

5.1.2. Mounting mechanism & modular design

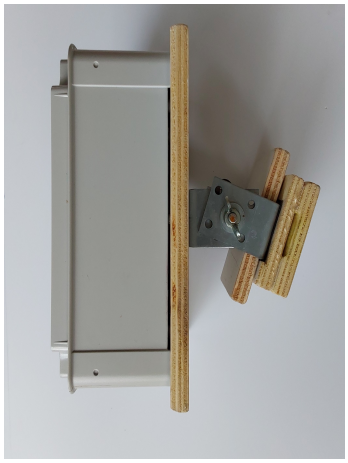
To support rapid assembly and component exchange, a 4 mm PVC platform (cut with a utility knife) was press-fitted into the enclosure and lined with 20 mm Velcro tape (Figure 5.3a-5.3b-5.3c). Steel L-profile brackets were assembled into an adjustable hinge mechanism on the enclosure's back panel; a wingnut and rubber washer lock the hinge's position, enabling stable vertical alignment (Figure 5.3d). The enclosure was then attached to 12 mm plywood via its factory corner-mount screw holes located outside the sealed volume to preserve weather resistance. Opposite the hinge, a wooden block with milled slits served as an anchor for tension straps, allowing the entire assembly to be quickly secured in the field (Figure 5.3e). Plywood and steel were chosen for their low cost and ease of fabrication for short term deployments; for long-term use, other materials should be investigated.

(a) Interior of the Schneider Electric Thalassa TBS enclosure used to house all system components.

(b) Custom-cut 4mm PVC platform designed for press-fit installation into the enclosure base.



(c) Final platform installation with applied Velcro strips to allow modular placement and replacement of components.



(d) Side view of the assembled hinge mechanism using steel L-profiles and 12mm plywood for vertical alignment and structural support.

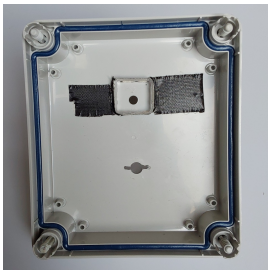
(e) Close-up of the adjustable hinge secured with a wingnut and rubber washer for fine-tuned positioning and quick deployment.

Figure 5.3: Assembly and mounting of the system enclosure.

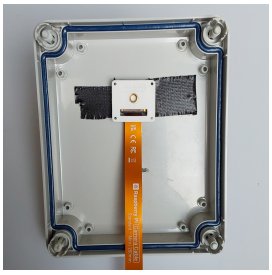
5.1.3. Installing camera module

To integrate the camera and maintain environmental sealing, two apertures (9 mm for the lens, 5 mm for the IR LED) were drilled in the enclosure lid using a step drill (Figure 5.4a). To minimise optical interference in the captured images, the lens and IR emitter should not be positioned too closely together; therefore, the apertures were spaced sufficiently apart during drilling. An Arducam lens housing was shortened by 2 mm at the front so that its transparent cover sits flush with the lid (Figure 5.4f-5.4e); it was then affixed inside the lid with adhesive and black tape to block stray light and LED indicators (Figure 5.4b). A polarizing filter was screwed into its circular adapter and was cut to size and adhered over the camera aperture to seal against dust and moisture while reducing water-surface glare. A transparent rain shield, fabricated from a plastic container and roughened with sandpaper was attached externally to diffuse incident light during daytime operation (Figure 5.4d). Finally, the CSI-2 camera connector ribbon cable is attached to the camera module (Figure 5.4c).

(a) Drilled apertures in the enclosure lid for the camera lens (9mm) and IR LED (5mm), made using a step drill to ensure precise alignment.

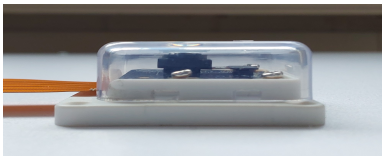


(b) Interior view of the shortened Arducam camera housing mounted flush against the lid and sealed with tape to block stray light.

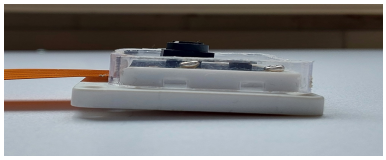


(c) Installation of the camera module inside the lid, with the ribbon cable routed for connection to the main board.

(d) External view showing the rain shield fabricated from a translucent plastic container, affixed to diffuse direct sunlight.



(e) Side views of the modified camera housing: original.

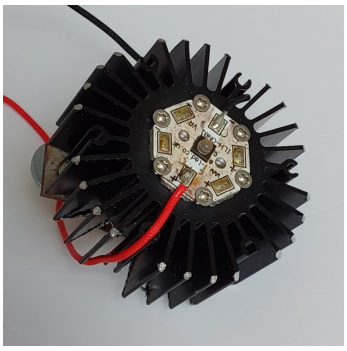


(f) Side views of the modified camera housing, shortened by 2mm to align the lens flush with the outer surface for a watertight fit.

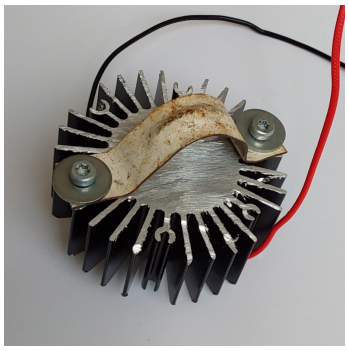
Figure 5.4: Integration of the camera module into the system enclosure.

5.1.4. Installing IR-illumination module

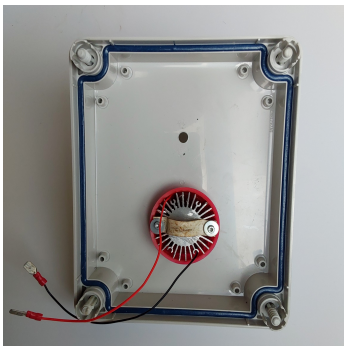
The ILH-IN01-85SL-SC211-WIR200 IR LED emitter was secured to an LED-star heatsink with supplied mounting screws and thermal interface fluid (Figure 5.5a). A saddle bracket was fixed to the heatsink's rear so the assembly can be withdrawn for maintenance without stressing the LED (Figure 5.5b). The heatsink assembly was enclosed in a protective plastic cap and bonded to the interior of the enclosure lid (Figure 5.5c-5.5d-5.5e). The MOSFET/resistor PCB was mounted adjacent to the heatsink and connected via shrink lugs (Figure 5.5f). A diffusive plastic filter was attached over the IR LED to seal the opening and scatter 850 nm illumination 5.4d, reducing hot-spot glare and reflections.



(a) Top view of the ILH-IN01-85SL-SC211-WIR200 infrared LED emitter mounted to a star-type aluminium heatsink using thermal paste and screws to ensure efficient heat transfer.



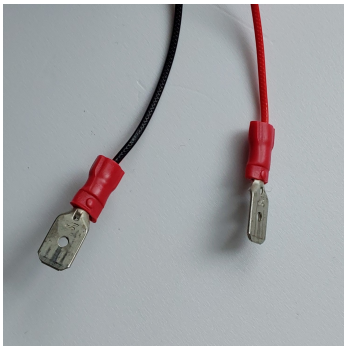
(b) Rear view showing the saddle bracket used to attach the heatsink securely while allowing for maintenance removal without stressing the LED.



(c) Internal view of the enclosure lid with the LED-heatsink assembly installed and wired.

(d) Image of the 3D-printed plastic cap used to enclose the LED module, featuring a central aperture for light transmission.

(e) Fully assembled unit with the diffuser glued in place to the cap, scattering emitted IR light and sealing the enclosure.

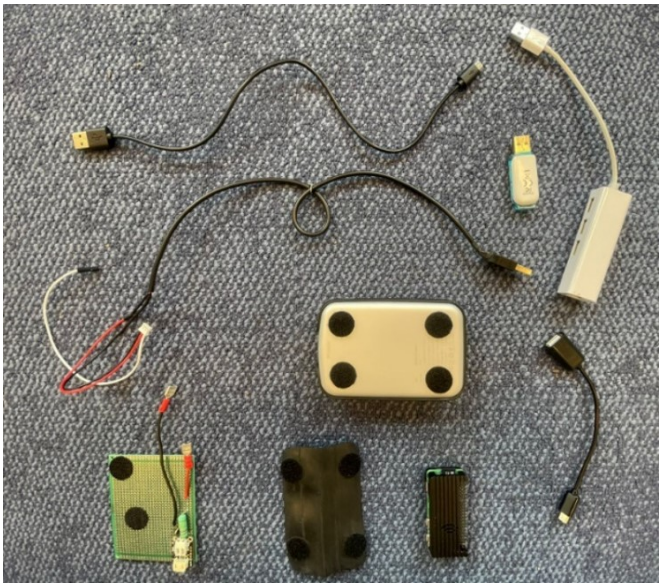


(f) Crimped shrink-lug connectors on the LED leads for modular connection to the control PCB.

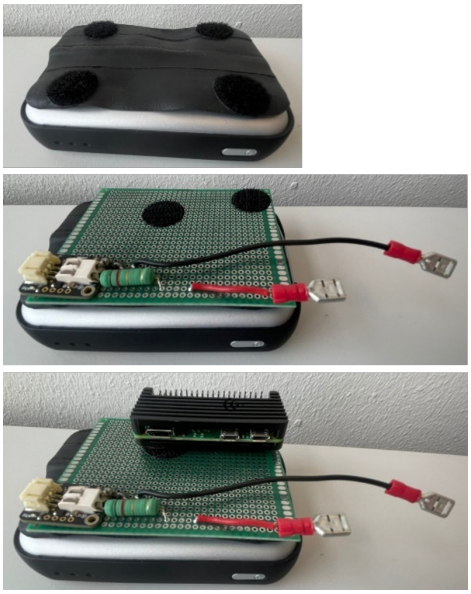
Figure 5.5: Integration of the IR LED module with heatsink, enclosure, and diffuser.

5.1.5. Device assembly

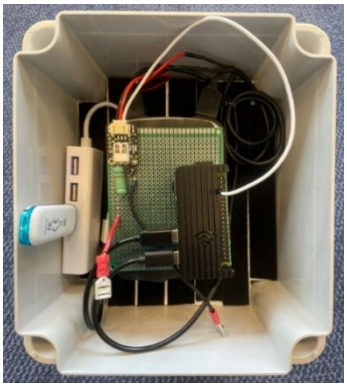
The Voltaic V50 battery pack was secured to the enclosure floor with Velcro tape; a rubber pad between the pack and PCB prevents short circuits (Figure 5.6b). The Raspberry Pi Zero 2 W is enclosed in its aluminium case, which also functions as a passive heatsink. It is secured via Velcro to an available area of the 60 × 90 mm PCB immediately adjacent to the MOSFET driver and the 1.2 Ω resistor that regulate power to the IR LED (Figure 5.6c-5.6d). A USB hub was mounted alongside the battery pack and connected to the Pi with a USB-to-micro-USB adapter cable. This enables the user to connect both a USB flash drive and utilities like a keyboard and mouse to the system in case remote access software fails. Power lines from the battery were routed to both the MOSFET board and the Pi (Figure 5.6c). Finally, the IR-LED cable and the CSI-2 ribbon cable were inserted into their respective ports. Once connections were confirmed, the battery pack was activated, and the lid was re-installed and screwed down to restore weather resistance (Figure 5.6e).



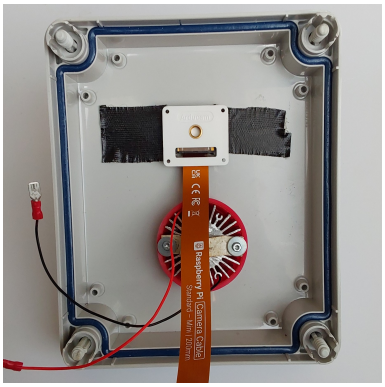
(a) Overview of all hardware components prior to installation, including battery pack, Raspberry Pi, IR control board, USB hub, storage devices, and cables.



(b) Velcro-mounted components on the battery pack include the insulating pad, IR LED control PCB, and Raspberry Pi Zero 2 W with heatsink, enabling a compact and removable integration.



(c) Top-down view of the full internal configuration, showing the Pi, power routing, USB hub, and flash drive.



(d) Inside lid with IR LED, camera module, and corresponding power and ribbon cable connections in place.

(e) Exterior view with reinstalled lid and transparent rain shield covering the camera aperture to protect against glare and weather exposure.

Figure 5.6: Final assembly and integration of power, processing, and connectivity components.

5.1.6. Reference object

A 4 mm matte-white foam-PVC board (600 × 500 mm) was fastened to 15 mm plywood using stainless-steel screws (Figure 5.7a). Saddle brackets were then attached to the plywood's rear, avoiding penetration of the PVC face and holding two 25 mm-diameter rods (Figure 5.7b-5.7c). In combination with adjustable tension straps, this arrangement provides a modular mounting system for field deployment (Figure 5.7d).

(a) Front view of the 4mm matte-white foam-PVC board (600×500mm), offering high contrast for reliable waterline detection.

(b) Rear view showing the 15mm plywood backing, to which the PVC is attached using stainless-steel screws for structural stability.

(c) Side view highlighting the layered construction and saddle brackets that hold two 25mm mounting rods for depth and height adjustability.

(d) Field deployment of the assembled reference board in a natural river channel, stabilized using tension straps and anchored between adjacent wooden branches.

Figure 5.7: Construction and installation of the modular reference board for waterline detection.

5.2. Software

To enable image capture, illumination control, image processing, and the storage and posting of water level detection results, a specific software configuration must be implemented. This section describes the software required to successfully run the water level detection algorithm. First, the acquisition of the operating system for the Raspberry Pi is discussed, followed by a list of the required software packages. Finally, methods for remotely accessing the system are covered. While this chapter introduces the general concepts of these software components, detailed instructions can be found in Annex D: System Setup Manual and Annex E: Remote Access Manual.

5.2.1. System setup summary

To prepare the Raspberry Pi environment, the Raspberry Pi OS (64-bit, Bookworm) is flashed onto a microSD card using the official Raspberry Pi Imager. During this process, the system can be preconfigured with login credentials, Wi-Fi settings, SSH/VNC access, and locale preferences. This results in a ready-to-use software environment, suitable for both direct and remote operation.

Direct operation can be achieved by connecting a keyboard and mouse via the USB hub and attaching a monitor or portable display using a mini-HDMI to HDMI cable. Remote operation is enabled through software tools, as described in Section 5.2.2 (see Annex D.1 for system setup manual).

For stability during image processing tasks, the default swapfile size is increased from 100 MB to 1024 MB via the terminal. This is necessary because the Raspberry Pi Zero 2 W has limited physical RAM (512 MB), which can lead to system crashes or unresponsiveness when handling memory-intensive operations such as image analysis. Increasing the swapfile size provides additional virtual memory, allowing the system to offload some of the processing load to disk and operate more reliably under heavier workloads (see Annex D.2 for instructions).

All necessary software dependencies are installed through a single terminal command. These include system-level camera tools and drivers:

- `libcamera-apps`
- `libcamera-dev`
- `v4l-utils`

Alongside these, the following core Python libraries are required for image processing, data handling, and hardware control:

- `python3-picamera2 0.3.27` (Picamera2 API & libcamera bindings)
- `python3-pillow 11.2.1` (PIL image handling)
- `python3-numpy 2.2.6` (array operations)
- `python3-matplotlib 10.3.1` (plotting backend)
- `python3-scipy 1.15.3` (signal processing & statistics)
- `python3-requests 2.30.0` (HTTP for ORC API integration)
- `python3-psutil 7.0.0` (system resource monitoring)
- `python3-rpi.gpio 0.7.1` (GPIO pin control)

For detailed instructions and troubleshooting, refer to Annex D.3.

To ensure autonomous operation, the main Python script is configured as a `systemd` service, allowing it to start automatically on boot and restart upon crashes. The service file is placed in `/etc/systemd/system/`, and activated via `systemctl`. This setup enables reliable, self-recovering operation in off-grid deployments (See annex D.4 for detailed instructions).

Full instructions, configuration files, and command listings are provided in:
Annex D: System Setup Manual
Online repository: GitHub link

5.2.2. Remote access and data retrieval

To enable remote control and file access, both graphical (VNC) and terminal-based (SSH) remote access methods are supported. Graphical access is particularly useful during debugging or calibration, as it provides an interactive visual interface. Terminal-based access, on the other hand, allows users to connect to the system while the water level detection algorithm is running, enabling monitoring without interrupting its operation. Additionally, file transfers are managed via SFTP using WinSCP.

Remote desktop access is provided through RealVNC. After enabling the SSH and VNC interfaces on the Raspberry Pi, a connection can be established using RealVNC Viewer and the device's IP address. This allows full graphical user interface (GUI) access from a workstation on the same network (see Annex E.1 for detailed instructions).

Headless operation is supported via SSH, using PuTTY. The SSH server is enabled through the raspi-config tool, and Wi-Fi credentials are entered to connect the Pi to a mobile hotspot or local network. After identifying the Pi's IP address, users can log in via PuTTY for full terminal access without a display (see Annex E.2 for detaild instructions).

File transfers are handled using WinSCP, an SFTP client. Once configured with the Pi's IP address and login credentials, WinSCP allows users to upload or download images, logs, and CSV files. The full project directory, including scripts and configuration files, can also be transferred to the Pi in one step (see Annex E.3 for detaild instructions).

Full instructions, configuration files, and command listings are provided in:
Annex E: Remote Access Manual
Online repository: GitHub link

5.3. Water level detection algorithm

With the hardware components described in Section 5.1 and the software environment configured in Section 5.2, this section focuses on the core of the system: the implementation of the water-level detection algorithm. In Chapter 1, the basic working principle of the algorithm was introduced. This chapter builds on that foundation by presenting the full implementation and describing the underlying Python code. It begins with an explanation of the image pre-processing steps, followed by a detailed overview of the two algorithmic approaches used for water-level detection: the mean-difference method and the Kolmogorov–Smirnov (KS) test method.

5.3.1. Image pre-processing

The water-level detection algorithm runs autonomously on the Raspberry Pi Zero 2 W, analysing vertical pixel-intensity patterns in each captured image. Each input image, originally in RGB format with pixel values ranging from 0 to 255, is first normalized to floating-point values between 0 and 1. Depending on the selected detection mode, the algorithm then extracts a specific channel to generate a single detection array. This can be either the normalized RGB values (original mode), the brightness component (Value channel), the colour tone (Hue), or the colour intensity (Saturation) from the HSV colour space. This preprocessing procedure is visualised in Figure 5.8.

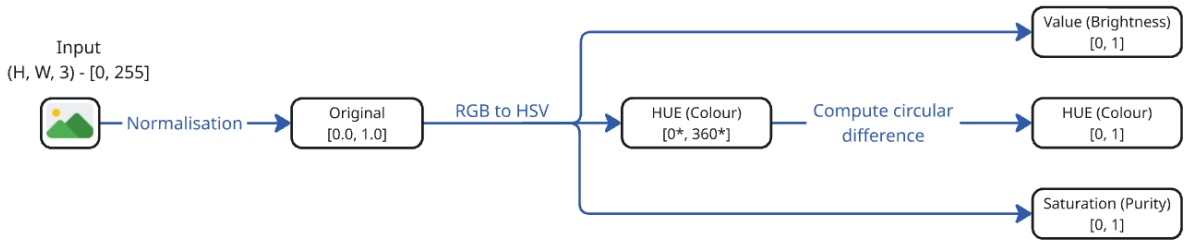


Figure 5.8: Preprocessing workflow for image-based waterline detection.

When using the Hue channel, a special transformation is required to account for its circular scale, which ranges from 0° to 360°. Without this correction, colours that are perceptually similar (such as red (0°)

and purple (360°) as shown in figure 5.9) can appear artificially distant in numerical terms, leading to false high contrasts. To correct for this, the algorithm computes the minimal circular difference, ensuring that hue differences reflect actual colour shifts rather than numeric discontinuities. This improves the reliability of waterline detection in scenes with subtle colour gradients.

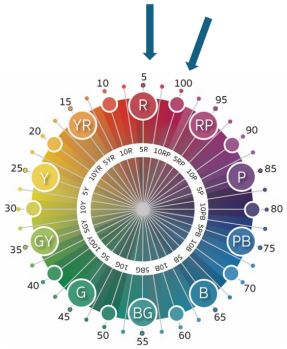
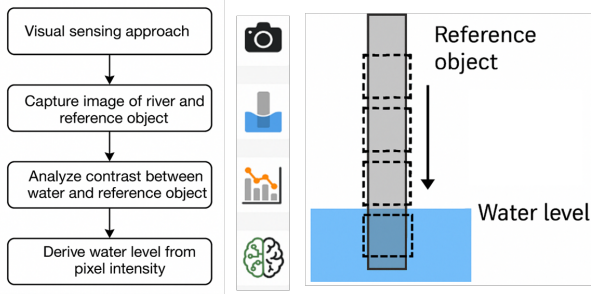


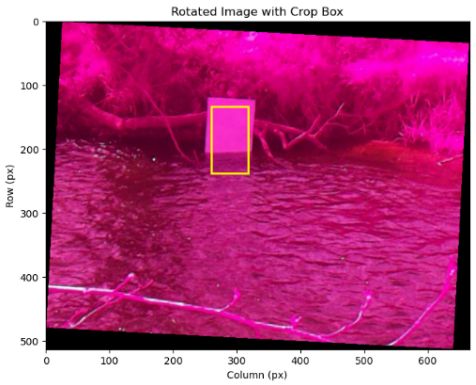
Figure 5.9: Illustration of the circular nature of the Hue channel [47]

5.3.2. Mean-difference method

To identify the water level, an algorithm was developed. To locate the waterline, the algorithm identifies shifts in pixel intensity along the vertical axis of the reference object, where the waterline typically causes a distinct visual transition. To detect this, the detection array is divided into a series of small horizontal ”boxes” as visualized in Figure 5.10a. For each adjacent pair of boxes, the algorithm calculates the absolute difference in their mean intensity. For the hue channel, a circular difference is used to account for the wraparound nature of hue values. These raw differences are then passed through a one-dimensional Gaussian filter to reduce random noise and highlight structured intensity changes. The resulting smoothed signal is normalized into a probability distribution over the vertical axis, emphasizing the most likely positions of the waterline.



(a) Conceptual illustration showing image processing principle; contrast between wet and dry areas allows the waterline to be inferred from image data.



(b) Example image from field testing showing a cropped region (yellow box) containing the reference object, used for waterline detection.

Figure 5.10: Schematic overview of the visual waterline detection process.

The peak of this distribution represents the most probable vertical pixel coordinate of the waterline within that image channel. In cases where multiple peaks exist, the highest (i.e., most prominent) peak in the distribution is selected as the candidate waterline for that channel. To improve robustness under varying lighting and background conditions, this process is repeated across all four image channels (original RGB intensity, value, hue, and saturation) as shown in Figure 5.11. For each channel, the maximum value of the smoothed difference signal (before normalization) is recorded as an indicator of detection strength. The channel with the highest detection strength is selected, and its corresponding peak is used as the final water-level measurement.

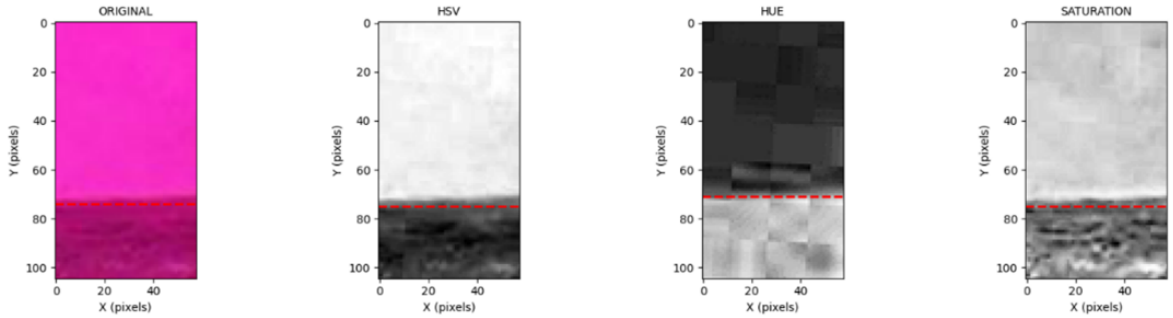


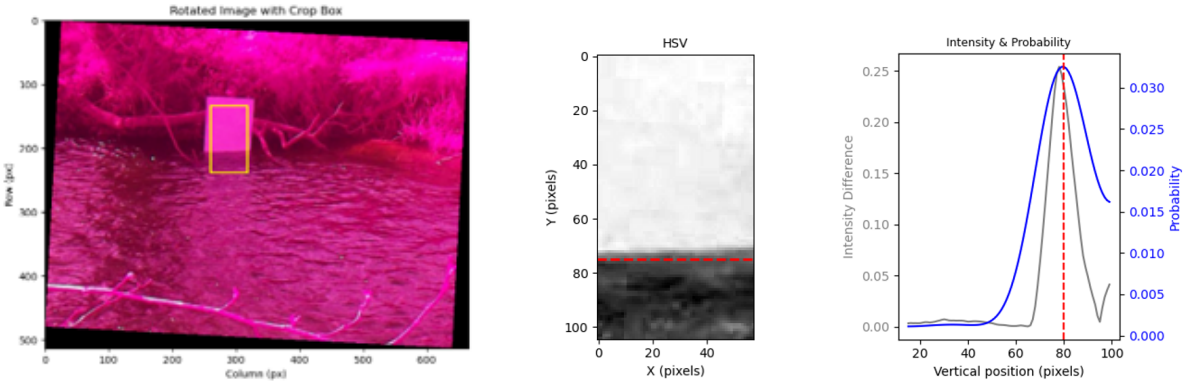
Figure 5.11: Algorithm results displaying the most probable waterline location (red dashed line) for each image channel.

Each cycle's results (including the per-channel waterline positions, the selected detection mode, and its corresponding score) are appended to a CSV file for later analysis as shown in Table 5.1. In addition, visual outputs are saved to support quality control, including the cropped image (Figure 5.12a), the detected waterline overlay (Figure 5.12b), and the intensity difference and probability curves (Figure 5.12c).

Table 5.1 displays the algorithm output. The WL_<channel> columns (column 2-5) reflect the detected y-coordinate representing the waterline in image space. The best_mode column (column 6) reflects the image channel with the highest detection strength. The best_score value (column 7) reflects the maximum intensity difference between adjacent boxes of that channel, prior to normalization. Since all intensities are scaled to the [0–1] range, best_score also falls within this range, where 1 indicates maximum contrast between two comparison boxes, and 0 indicates no difference.

image_name	WL_original	WL_hsv	WL_hue	WL_saturation	best_mode	best_score
capture_140235.jpg	207	208	204	208	hsv	0.25

Table 5.1: Logged detection output per image cycle.



(a) Captured image overlaid with the detection crop box (yellow), isolating the reference object region used in the analysis.

(b) Extracted crop visualized in the HSV brightness channel, with the detected waterline marked in red.

(c) Plot showing the absolute intensity difference curve (grey) and its corresponding normalized probability distribution (blue). The vertical red dashed line indicates the selected waterline position corresponding to the peak detection score.

Figure 5.12: Visual outputs for quality control and verification of the waterline detection process.

5.3.3. KS-test method

As an alternative to the mean-difference approach, a Kolmogorov–Smirnov (KS)-based method has also been implemented. Instead of comparing average pixel intensities between adjacent boxes, this method evaluates the entire distribution of pixel values. Specifically, it calculates the KS statistic, which

quantifies the maximum difference between the cumulative distribution functions (CDFs) of two adjacent regions. An illustration of this concept is shown in Figure 5.13, where the vertical arrow indicates the KS statistic, the largest vertical gap between the two CDFs (one shown in red, the other in blue). This single value reflects how dissimilar the pixel distributions are between the two regions.

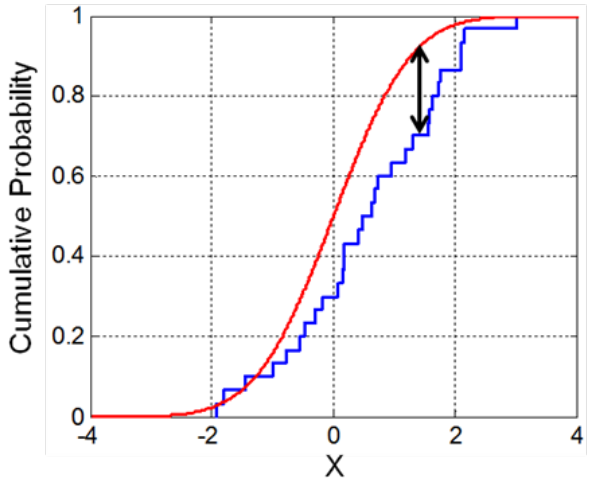
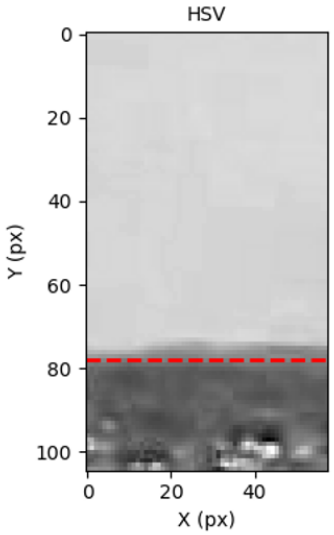


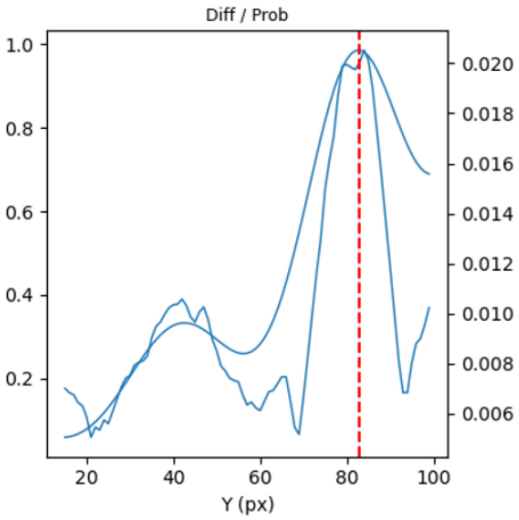
Figure 5.13: Illustration of the Kolmogorov–Smirnov (KS) statistic.[7]

Because the KS test is sensitive to distribution shape rather than just average intensity, it can detect subtle shifts in intensity that the mean-difference method might miss. This is especially useful in noisy or low-contrast environments. For hue values, a sine transform is applied to preserve circular continuity before computing the CDFs.

Once the KS statistic is computed for each vertical step, the resulting values are smoothed using a one-dimensional Gaussian filter, normalized, and passed through the same peak-detection process used in the mean-difference method. The strongest peak across all channels is selected as the final water-level estimate. The output of the KS-test method are shown in Figure 5.14.



(a) Cropped and normalized image showing the vertical structure of the reference object with a dashed red line indicating the detected waterline.



(b) Corresponding KS statistics (blue curve) computed along the vertical axis, and their smoothed probability distribution. The red dashed line marks the vertical position of the highest peak in the normalized signal, which is selected as the most probable waterline location.

Figure 5.14: KS-test detection process in the HSV channel.

6

Operation instructions

This chapter provides instructions for deploying and operating the water-level monitoring system in the field. Section 6.1 covers initial setup and orientation, including verifying service status and installing the reference board. Section 6.2 describes the calibration and verification procedure to align the optical and algorithm parameters (in GUI mode). Section 6.3 explains how to run the device in command line interface mode (CLI) for unattended, periodic measurements and how to retrieve data on demand.

6.1. Setup and orientation

Before sealing the enclosure, establish a graphical access using RealVNC as described in Section 5.2.5. Verify that the water-level detection service is active with the appropriate systemctl commands. Once the service is confirmed to be running correctly, the enclosure may be sealed and prepared for deployment.

Next, install the reference object in the river so that its face is positioned perpendicular to the camera lens. This perpendicular orientation is essential in the current POC because each pixel is directly mapped to real-world dimensions. If the reference board is tilted relative to the camera, parts of the board that are closer to the lens will appear larger (each pixel represents a smaller real-world area) compared to parts that are farther away, where each pixel represents a larger area. This perspective distortion would lead to inaccuracies in water level detection. To allow more flexible camera angles in future versions, georeferencing techniques would be required (see Section 8.3: Limitations).

Once positioned, secure the reference object firmly and ensure the camera's line of sight is unobstructed by overhanging branches or floating debris (Figure 6.1). While mounting methods may vary depending on the site, the adjustable hinge and tension-strap system provides sufficient flexibility to attach the enclosure to either vertical or horizontal supports.

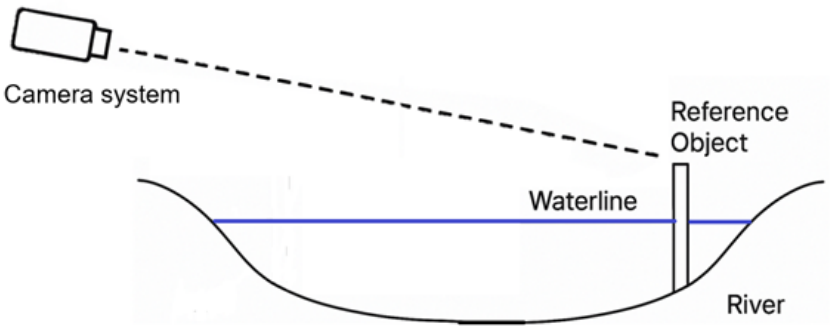


Figure 6.1: Schematic overview of the deployment setup

6.2. Calibration and verification

To ensure accurate water-level detection, the system must be calibrated after installation. This involves aligning the camera with the reference board and extracting key processing parameters. General overview of this process is described below. A more detailed description and checklist can be found in annex F.

First, physical alignment is required to avoid perspective distortion. The camera must be positioned perpendicular to the reference board so that pixel dimensions accurately reflect real-world distances. A live preview is used to fine-tune this orientation before locking the camera mount.

Next, image processing parameters such as rotation angle and cropping boundaries are configured. These parameters ensure the detection algorithm focuses on the correct region of interest and compensates for any slight misalignments in the camera setup. The system's measurement interval is also reviewed and adjusted to suit the intended deployment frequency.

Once these settings are defined, the setup is verified by running a test detection cycle. This confirms that the camera, algorithm, and storage pipeline are functioning as intended. Visual and numerical outputs are inspected to detect any misalignment, image noise, or processing errors.

During desk tests, GUI mode required more energy compared to operating in command line interface (CLI) mode. That is why, after calibration is complete, the system is switched to CLI mode to reduce energy consumption. Remote access is maintained via SSH, allowing users to monitor performance, inspect logs, or restart services without requiring a graphical display. If configured for remote uploading, successful data transmission to the OpenRiverCam server is verified as part of final validation.

Detailed calibration and verification instructions are provided in Annex F: Operation and calibration Manual.

6.3. Operation

Once calibration is complete and the Raspberry Pi is configured to boot in headless CLI mode, the system begins automatic measurement cycles at fixed intervals, as defined in the configuration file. During each cycle, the system illuminates the reference board, captures an image, runs the detection algorithm, optionally uploads the result, and then enters standby until the next cycle, requiring no user interaction.

Manual control remains possible: a cycle can be triggered on demand, or the service can be restarted or paused as needed. Output files, including raw images, annotated results, and CSV logs, are stored locally and can be retrieved via remote access (see annex F.2 for commands). For data access or debugging, the system can temporarily be switched back to GUI mode and accessed via VNC and SFTP tools like WinSCP. After retrieval, CLI mode is restored to maintain low-power operation.

Periodic maintenance involves checking battery levels, ensuring the reference board remains aligned and free of obstructions, and inspecting the enclosure and mounting hardware. With this setup, the system delivers reliable, unattended water-level measurements, while still allowing straightforward manual access when required.

Detailed operation and maintenance procedures are provided in Annex F: Operation and calibration Manual.

7

Validation and characterization

This chapter evaluates the prototype's performance through field validation and detailed characterization. Section 7.1 describes the experimental deployment including location, mounting, environmental conditions, and manual reference measurements, while Section 7.2 outlines the data-acquisition protocol, output formats, and reference-level definition. Section 7.3 presents the detection errors of both the mean-difference and KS-test methods, including outlier analysis, diurnal and precipitation effects, and overall statistics. Finally, Section 7.4 assesses the system's energy consumption per cycle and over extended duty cycles to gauge its suitability for off-grid operation.

7.1. Experimental setup

The deployment took place on the Kleine Geul River near Wittem (Gulpen-Wittem, Limburg, the Netherlands). The site features narrow, loess- and silt-lined banks and a steep gradient that produces fast, highly dynamic flows in response to rainfall [46]. The location of the experimental setup is shown in Figure 7.1, including coordinates.

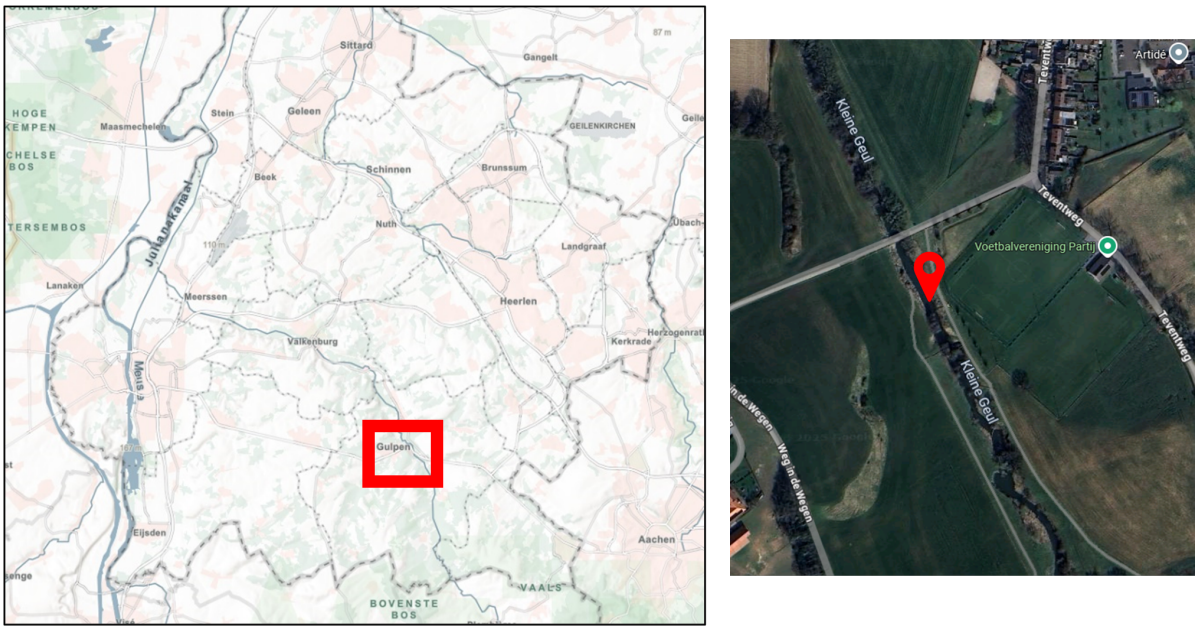


Figure 7.1: Field deployment location (50.80762°N, 5.91350°E).

The system operated continuously for 42 h, with a single manual check and service restart at the 20 h mark to confirm output quality and battery status.

Figure 7.2: Field deployment of the water-level monitoring system at the Kleine Geul.

The weather-resistant enclosure was affixed to a sturdy, elevated support above anticipated flood levels, facing the opposite bank. The reference object was mounted partly submerged, perpendicular to the camera's optical axis and parallel to the river flow to ensure consistent framing. All components were secured with the tension-strap mounting system outlined in Section 5.1.2.

Days 1 and 2 were predominantly sunny with intermittent clouds. Light rain fell during the night between Day 2 and the morning of Day 3, briefly altering illumination and surface reflectivity [42].

7.2. Data acquisition

This section summarizes the timing, data formats, and reference measurements collected during the 42 h field deployment.

During field testing, the device was configured as follows:

- Capture interval: After each detection cycle, the system rested for 540 s (9 min). Given processing times of 20–40 s, this delivered an effective measurement frequency of just under 10 min, meeting the availability requirement.
- Infrared LED usage: The IR emitter remained active during day and night (only when capturing) to confirm system operation visually and to minimize illumination-related variability.
- Network configuration: Real-time posting to the OpenRiverCam API was disabled due to lack of Wi-Fi coverage on-site.
- Operating mode: CLI mode was used to reduce power consumption; USB-backup functionality was disabled, as it was not yet compatible with CLI operation.

A total of 270 measurement cycles were recorded over two continuous deployment segments (approximately 20 h followed by 22 h) for a combined duration of 42 h.

Each cycle produced three primary outputs under `/home/bjorn/wd__directory/results/`:

- Raw images (`raw_images/`): 640 $\times$ 480 px JPEG frames captured by the camera.
- Rotated & cropped views: full-frame JPEGs showing the applied rotation and crop window overlay as shown in Figure 5.12a
- Mode-specific overlays: figures (all in one PNG file) combining the detected waterline overlay on the cropped view with its corresponding intensity-difference or KS-statistic graph as shown in figures 5.11 & 5.12c.
- Summary log (`algorithm_results.csv`): CSV entries listing the filename, waterline pixel position for each mode, selected best mode, and score as shown in Table 5.1.

A resolution of 640 $\times$ 480 pixels was selected for image capture, as the algorithm demonstrated robust performance at this scale. Preliminary tests using higher resolutions (e.g., 1024×768) showed only marginal gains in detection precision while significantly increasing runtime.

An acoustic sensor system located approximately 3 m downstream of the prototype recorded water levels at 15 min intervals throughout the 42 h trial. Because the sensor's readings exhibited very low variability (std = 4 mm), the reference water level was treated as effectively constant for validation purposes . All camera-based detections were then compared against this constant in image-space.

To define this constant in image space, a custom Python script was used to manually annotate the waterline in a set of 42 images (the first image of each hour). The mean of these 42 pixel coordinates defined the image-space validation level. Algorithm outputs were compared to this baseline, and errors were converted to millimetres using an site-specific conversion rate (1px = 6.7mm).

To express detection errors in physical units (millimetres), a pixel-to-length conversion factor was estimated using the known width of the foam-PVC reference board. With a real-world width of 500 mm and an image width of 75 pixels, a scale of approximately 1 px = 6.7 mm was derived. This scale assumes that the reference board remained perpendicular to the camera's optical axis during the deployment.

7.3. Results

This section presents the error characteristics of the two detection methods. Both the mean-difference method and the KS-test method produced comparable results in proximity to the actual water level, with an average mean absolute error (MAE) of 1.8 cm for both approaches. While the KS-test method demonstrates greater robustness under varying conditions (particularly in response to diurnal lighting changes) the mean-difference method achieves slightly higher precision overall.

The remainder of this section elaborates on the field testing results in more detail. First, the raw output data is presented, along with a discussion of outliers for each method. This is followed by an analysis of diurnal and precipitation-related influences on detection performance. Finally, key performance statistics are reported.

7.3.1. Raw data and outliers

Figure 7.3 presents the time series of detection errors (defined as the difference between the detected waterline and the manually annotated reference in cm) for both the mean-difference method and the KS-test method over the 42-hour field deployment. All captured images and corresponding analysis outputs, including overlays, are available in the online repository (GitHub link).

For the mean-difference method, four notable outliers beyond ±10 cm are visible. The first two occurred around 16:00–17:00, coinciding with low-angle sunlight that produced glare on the water surface. Although a circular polarization filter was installed to suppress such reflections, it had been reattached without verifying its proper rotational alignment. As a result, the filter was ineffective, allowing strong reflections to degrade image contrast and impair the algorithm's performance (see repository for captured images).

The latter two outliers for this method appeared around dusk (20:00–21:00). During this transition from daylight to darkness, the camera's extended shutter times captured residual ambient light, introducing blur and reducing contrast. Both of which negatively affected detection accuracy.

For the KS-test method, three significant outliers with errors near –40 cm appear around 09:00 on the second day. Upon reviewing the algorithm output, these errors appear to stem from incorrect scoring of the colour channel outputs. In these cases, the mode selection mechanism incorrectly prioritized a misaligned or low-contrast channel, resulting in a detection failure.

Despite these specific cases, both methods show consistent performance across most of the deployment period, with error values clustered closely around the reference level. The outliers are isolated and traceable to identifiable environmental or algorithmic issues, indicating overall system reliability.

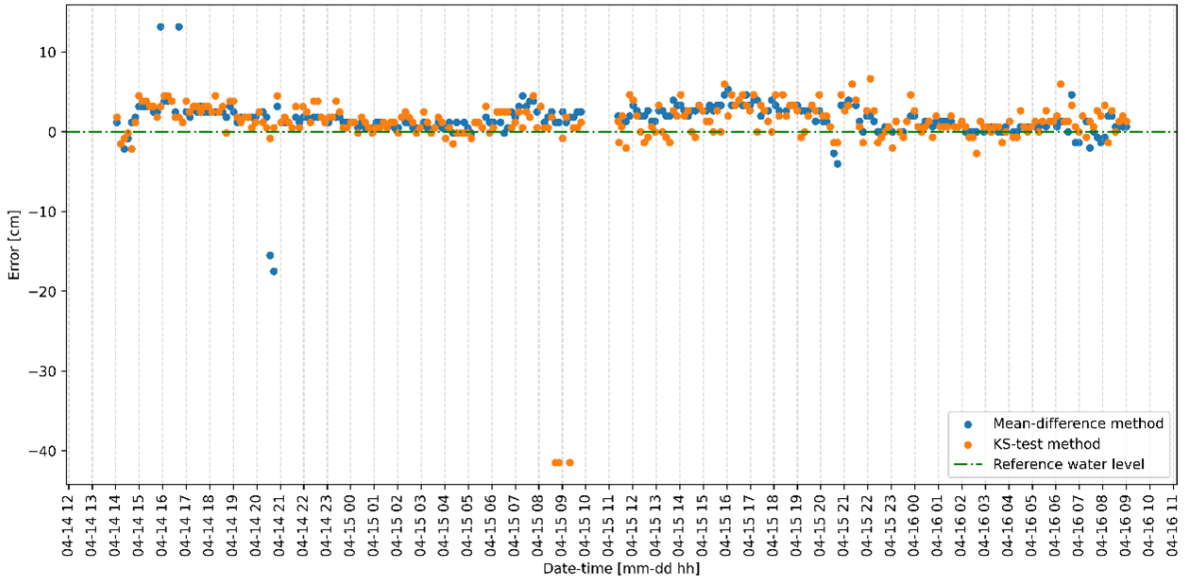


Figure 7.3: Time series of detection error (cm) for both the mean-difference and KS-test methods across the 42-hour field deployment.

7.3.2. Diurnal patterns

Introducing day (white background) and night (grey background) shading in the error plots (Figures 7.4-7.5) reveals a clear diurnal pattern in the mean-difference method. Errors tend to cluster higher above the zero-error line during daylight hours, indicating a systematic positive bias. This is confirmed by the fitted 24-hour sinusoidal trend (red line), which captures the regular daily fluctuation in detection error. The orange dashed line represents the bias, computed as the mean of all error values across the deployment period: 1.57 cm for the mean-difference method (Figure 7.4) and 1.02 cm for the KS-test method (Figure 7.5). These values reflect the average overestimation of water depth relative to the fixed reference level.

To improve visibility of the diurnal trend, the vertical axis range in both plots has been limited. As a result, the extreme outliers discussed in the previous section are not shown here. However, they are still included in the bias calculation and thus contribute to the mean error line.

The KS-test method exhibits more stable error distribution across the 24-hour cycle. Although the plot includes a 24-hour sinusoidal fit for consistency, the trend line does not clearly align with the actual day–night cycle, suggesting that no significant or consistent diurnal pattern is present. This indicates that the KS-test method is less sensitive to illumination changes, even if it produces slightly more scatter overall. The lower average bias and absence of systematic error shift further reinforce its robustness to environmental variability.

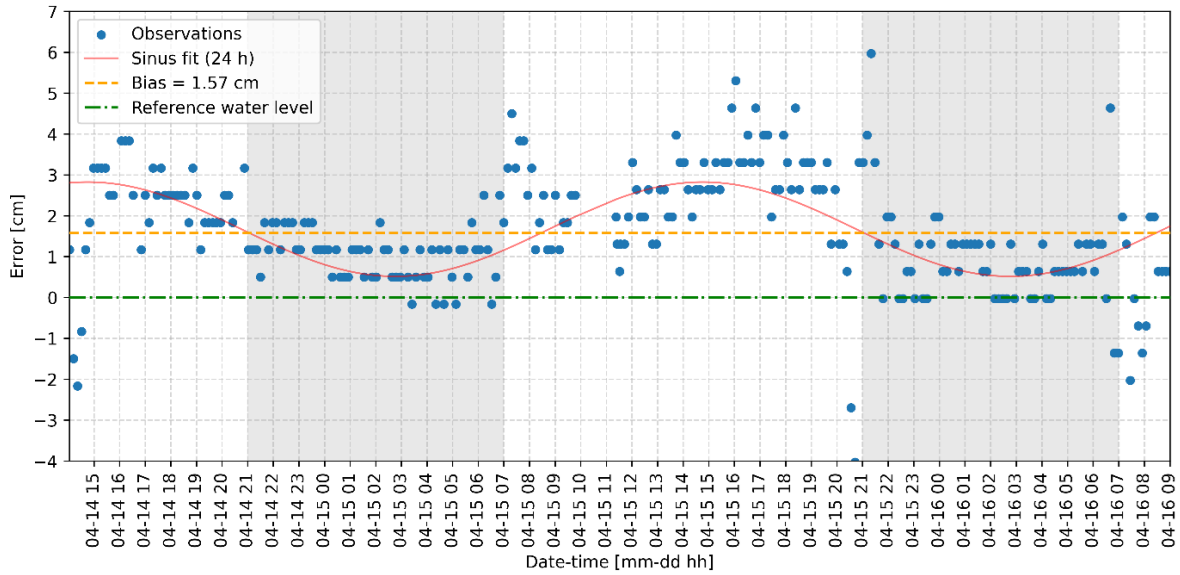


Figure 7.4: Detection error over time for the mean-difference method. A 24-hour sinusoidal curve (red line) highlights the diurnal bias, with a clear positive shift during daylight hours. The mean bias across the deployment was 1.57cm (orange dashed line). The green dashed line represents the zero-error reference level. Day (white) and night (grey) periods are shaded accordingly.

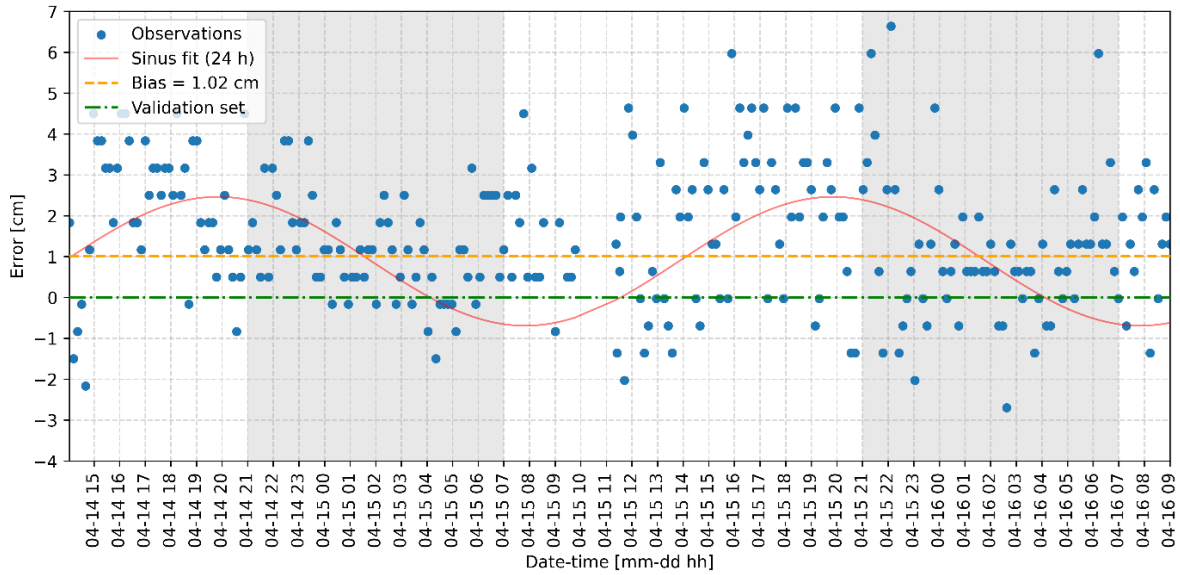


Figure 7.5: Detection error over time for the KS-test method. A sinusoidal fit (red line) is shown for comparison, but no clear diurnal pattern is observed. The average bias was 1.02 cm (orange dashed line), with relatively stable error across day and night intervals. Shaded regions distinguish day (white) and night (grey) conditions.

7.3.3. Influence of precipitation

Figures 7.6 and 7.7 show the detection errors over time for both methods, with precipitation periods highlighted in blue. Visually, errors appear slightly higher during rainfall events. However, this effect cannot be conclusively attributed to precipitation, as all rainfall occurred during daytime hours, when detection errors were already elevated due to diurnal lighting influences (Section 7.3.2).

It is important to note that only 51 out of 270 measurement cycles occurred during rain, limiting the strength of any conclusions. To fully assess the system's robustness in wet conditions, further testing is needed across a wider range of precipitation intensities, durations, and environmental settings.

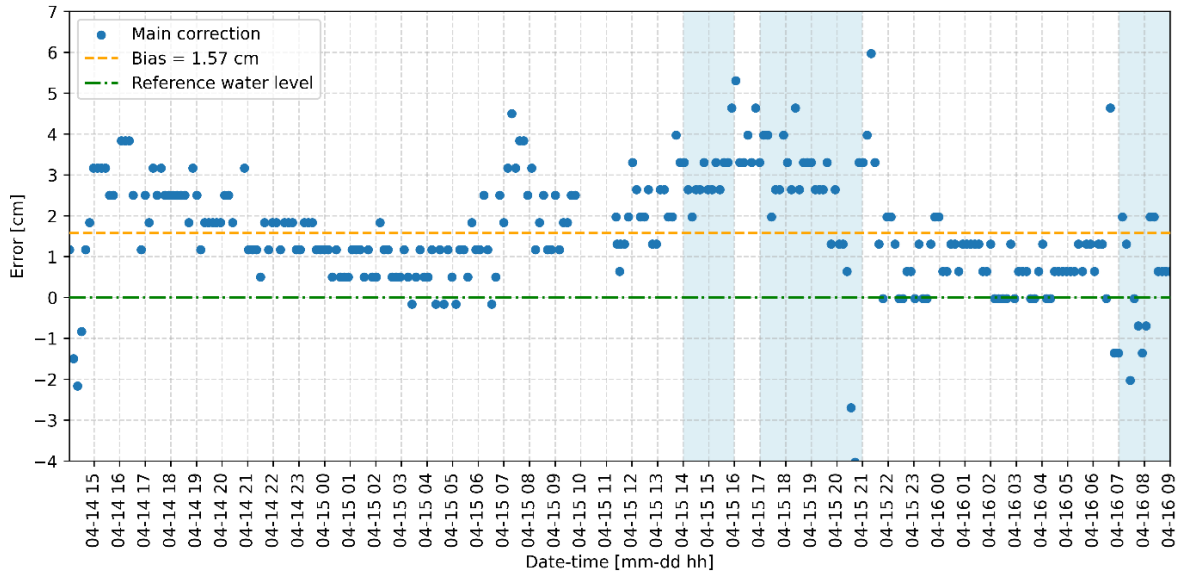


Figure 7.6: Detection errors over time using the mean-difference method, with precipitation periods shaded in blue.

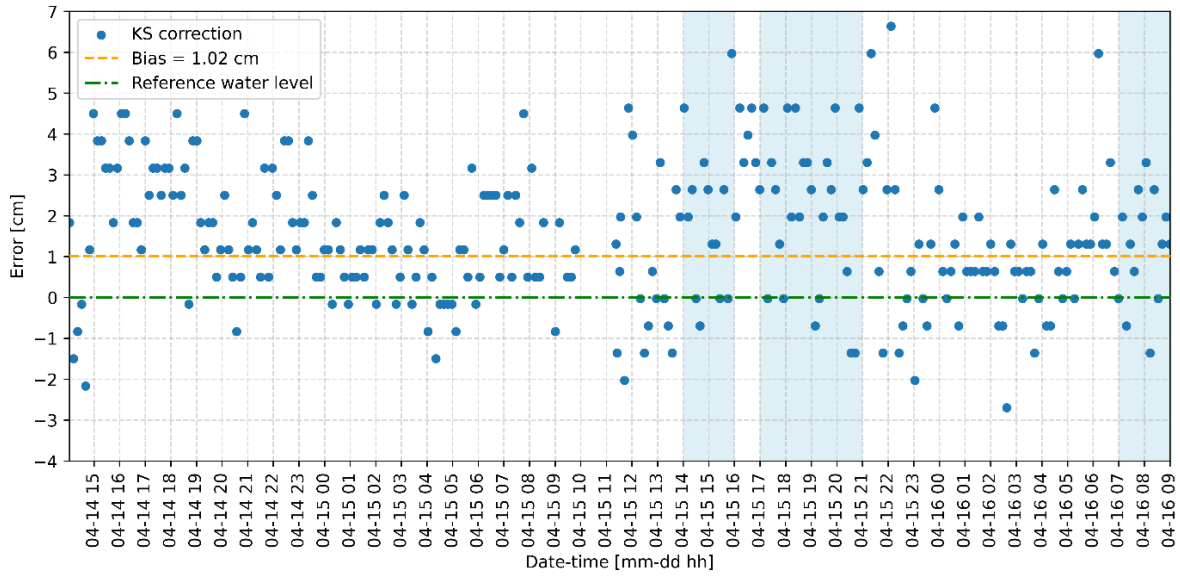


Figure 7.7: Detection errors over time using the KS-test method, with rain periods indicated in blue.

7.3.4. Performance statistics

Table 7.1 summarizes the overall accuracy and precision of both detection methods across all 270 measurement cycles. The mean-difference method shows a slightly higher bias of 1.57 cm compared to 1.02 cm for the KS-test method, but achieves better precision, with a standard deviation of 2.30 cm versus 4.82 cm for the KS-test. This reflects a more consistent performance with fewer extreme deviations.

In terms of absolute error, the mean-difference method records a lower mean absolute error (MAE) of 1.97 cm, and a lower root mean square error (RMSE) of 2.79 cm, compared to 2.23 cm and 4.92 cm respectively for the KS-test method. Notably, while both methods reach similar median accuracy (1.30 cm), the KS-test exhibits greater variability, including a minimum error of –41.50 cm, whereas the mean-difference method's minimum error is –17.50 cm.

The proportion of measurement cycles falling within specific error thresholds also highlights these differences. While the KS-test performs slightly better at the ±1 cm level (33.7% vs. 25.2%), the mean-difference method outperforms at higher thresholds: 95.6% of its errors fall within ±4 cm, compared to 91.5% for the KS-test.

To provide a common benchmark for comparison with conventional water-level sensors, accuracy is also expressed as the half-width of the 95% error interval. That is, the maximum deviation that includes 95% of all errors. Both methods yield the same 95% accuracy bound of ±4.64 cm, despite their different error distributions.

Table 7.1: Summary of accuracy and precision metrics for both detection methods over 270 field measurements.

Statistic	Formula	Mean-Difference	KS-Test
Sample Size	$N = \text{len}(y)$	270	270
Bias (cm)	$\text{Bias} = \frac{1}{N}\sum y_i$	1.57	1.02
Median (cm)	$\text{Median}(y)$	1.30	1.30
Precision (cm)	$\sigma = \sqrt{\frac{1}{N-1}\sum(y_i - \bar{y})^2}$	2.30	4.82
MAE (cm)	$\text{MAE} = \frac{1}{N}\sum\lvert y_i\rvert$	1.97	2.23
RMSE (cm)	$\text{RMSE} = \sqrt{\frac{1}{N}\sum y_i^2}$	2.79	4.92
Min Error (cm)	$\min(y_i)$	-17.50	-41.50
Max Error (cm)	$\max(y_i)$	13.17	6.64
95% Accuracy (cm)	$\max(\lvert P_{2.5}\rvert,\ \lvert P_{97.5}\rvert)$	4.64	4.64
% $\leq$ 1 cm	$\left(\frac{\text{count}(\lvert y_i\rvert \leq 1)}{N}\right) \times 100\%$	25.19%	33.70%
% $\leq$ 2 cm	$\left(\frac{\text{count}(\lvert y_i\rvert \leq 2)}{N}\right) \times 100\%$	64.81%	65.56%
% $\leq$ 3 cm	$\left(\frac{\text{count}(\lvert y_i\rvert \leq 3)}{N}\right) \times 100\%$	80.37%	78.52%
% $\leq$ 4 cm	$\left(\frac{\text{count}(\lvert y_i\rvert \leq 4)}{N}\right) \times 100\%$	95.56%	91.48%
% $\leq$ 5 cm	$\left(\frac{\text{count}(\lvert y_i\rvert \leq 5)}{N}\right) \times 100\%$	97.78%	97.41%

Table 7.2 compares detection performance between daytime and nighttime conditions. For the mean-difference method, a clear improvement is observed at night: bias drops from 2.01 cm to 1.00 cm, and precision improves significantly from 2.89 cm to 0.86 cm. The MAE and RMSE follow the same trend, indicating that this method is particularly sensitive to daytime illumination effects, such as increased water transparency and surface reflections.

In contrast, the KS-test method shows nearly identical bias values for day and night (both ≈1.02 cm), supporting its robustness to changing lighting conditions. However, its error spread is far greater during the day, with precision dropping from 6.26 cm to 1.49 cm at night. This suggests that while the KS method is less prone to systematic over- or underestimation across the day–night cycle, it is still susceptible to larger fluctuations in certain conditions, particularly under direct sunlight.

Table 7.2: Comparison of water-level detection performance during daytime and nighttime conditions for both the mean-difference and KS-test method.

Method	Period	N	Bias (cm)	Median (cm)	Precision (cm)	MAE (cm)	RMSE (cm)
Mean-Difference	Day	154	2.01	2.50	2.89	2.68	3.51
Mean-Difference	Night	116	1.00	1.17	0.86	1.02	1.32
KS-Test	Day	154	1.02	1.97	6.26	2.90	6.32
KS-Test	Night	116	1.02	0.64	1.49	1.34	1.80

Table 7.3 compares performance between dry and precipitation periods. For both methods, there is a modest increase in bias and MAE during rain events. In the mean-difference method, bias increases from 1.48 cm to 1.98 cm, while MAE rises from 1.86 cm to 2.44 cm. For the KS-test method, the

increase is more pronounced: bias rises from 0.82 cm to 1.87 cm, while MAE increases from 2.26 cm to 2.12 cm (note the reverse trend in MAE here).

However, these shifts remain within the range of one standard deviation from overall precision and should therefore be interpreted with caution. Additionally, only 51 of the 270 cycles occurred during rainfall, limiting statistical confidence. More field data under diverse and heavier precipitation would be required to reliably quantify performance impacts.

Table 7.3: Summary of detection performance under dry and rainy conditions for both the mean-difference and KS-test method.

Method	Condition	N	Bias (cm)	Median (cm)	Precision (cm)	MAE (cm)	RMSE (cm)
Mean-Difference	No Precip	219	1.48	1.30	2.40	1.86	2.81
Mean-Difference	Precip	51	1.98	2.64	1.82	2.44	2.68
KS-Test	No Precip	219	0.82	1.17	5.27	2.26	5.32
KS-Test	Precip	51	1.87	1.97	1.80	2.12	2.58

7.4. Power assessment

Energy consumption per processing cycle was first characterized under headless operating conditions. A Keweisi USB power monitor *(Otronic.nl)* was connected between the power bank and the Raspberry Pi Zero 2 W to log supply current and voltage, while runtime and CPU active time were recorded using the psutil library.

Over 100 single-cycle runs, the mean-difference method showed an average current draw of approximately 0.79 A at 4.95 V over 20.81 seconds of runtime, resulting in an average power consumption of $\sim$3.91 W and an energy usage of approximately 0.023 Wh per cycle. The KS-test variant drew around 0.84 A at the same voltage for 21.23 seconds of runtime, consuming $\sim$4.14 W on average and 0.024 Wh per cycle (Table 7.4). These results indicate that although the KS-test method consumes about 4–5% more energy and slightly more compute time, both pipelines remain efficient enough for continuous low-power operation.

A second set of tests simulated the expected field duty cycle: one measurement every nine minutes (six cycles per hour). Over one hour, including idle periods, the mean-difference method averaged of 0.552 A at 4.95 V, resulting in a total hourly consumption of approximately 2.73 Wh. The KS-test method consumed slightly more at 0.585 A (2.90 W), as shown in Table 7.5. These values suggest that both algorithms maintain stable and predictable power demand under representative deployment conditions.

Table 7.4: Average processing characteristics per cycle.

Mode	Voltage	Avg. Current	Avg. Duration per Cycle	Avg. Power	Avg. Energy per Cycle
Mean-difference	4.95 V	$\sim$0.79 A	20.81 s	$\sim$3.91 W	$\sim$0.023 Wh
KS-test	4.95 V	$\sim$0.84 A	21.23 s	$\sim$4.14 W	$\sim$0.024 Wh

Table 7.5: Estimated average power consumption over 1 hour of operation.

Mode	Voltage	Avg. Current	Avg. Power
Mean-difference	4.95 V	$\sim$0.552 A	$\sim$2.73 W
KS-test	4.95 V	$\sim$0.585 A	$\sim$2.90 W

The infrared LED was evaluated separately. During its 5-second activation per cycle ($\sim$30 seconds per hour), it drew a peak current of 1.24 A at 4.95 V, corresponding to $\sim$6.13 W during operation. Over one hour, this yields an additional 0.051 Wh, a negligible increment relative to the baseline power draw. Thus, intermittent use of the LED introduces only minor overhead in the total energy budget.

Taken together, these desk-based assessments and simulated field duty cycles demonstrate that the current hardware and software setup is well-suited for short-term autonomous deployment. To meet the six-month energy-independence goal, however, integration of a solar charging system will be essential.

Preliminary estimations indicate that duty-cycling the Raspberry Pi Zero 2 W between measurement cycles could reduce average power consumption from 2.73 W to 0.138 W. A more detailed evaluation of this approach, including limitations and practical considerations, is provided in the discussion (Chapter 8).

8

Discussion

This chapter reflects critically on the performance and design of the developed camera-based water level monitoring system by interpreting field results, comparing them with expectations and existing literature, and identifying key limitations. Section 8.1 explores the observed detection behaviour in the field, highlighting the influences of wind, sunlight, image compression, and spatial resolution on measurement accuracy. Section 8.2 positions the system in relation to conventional hydrological sensors, including radar, ultrasonic, and pressure-based devices, and examines its robustness under diurnal lighting variations and rainfall events. Finally, Section 8.3 discusses design and deployment constraints, including hardware durability, calibration complexity, power autonomy, and data handling, offering insights into areas for improvement and recommendations for future iterations of the system.

8.1. Interpretation of field data

Wind-driven ripples and advective waves on the reference board (observed on the order of ±3 cm) likely drive the observed accuracy shifts, since this noise matches the system's precision limits and therefore defines a practical lower bound on measurement resolution. To mitigate these disturbances, one approach could be burst-image averaging, in which a rapid sequence of frames is captured, and the detected waterline positions are averaged to smooth out peaks caused by individual waves. Alternatively, a rolling-average filter applied to consecutive detections may offer a similar smoothing effect. This however would increase the system response time as it takes averages of multiple detections.

The mean-difference method exhibits a daytime bias of approximately 2.1cm compared with a nighttime bias of around 1.0cm. In contrast, the KS-test method shows smaller diurnal variation, with biases of 1.86cm by day and 1.07cm by night, demonstrating greater resilience to changing illumination. This systematic overestimation during daytime is likely caused by sunlight penetrating the water surface and increasing the apparent depth in the captured image. As a result, a lighter intensity band becomes visible just below the actual waterline on the reference board, which may cause the algorithm to detect the transition too deep. One mitigation technique is to use an infrared-bandpass filter (*uwcamera.nl*) that restricts incoming light to a narrow IR wavelength range. Although such a filter was evaluated during desk tests, it was excluded from field trials to minimize setup complexity; nevertheless, these preliminary results indicate that an IR-bandpass filter could reduce daytime overestimation.

In addition to these optical effects, JPEG compression may also contribute to the misidentification of the waterline. Captured frames are stored in the JPEG format, which introduces compression artifacts by smoothing fine image details and altering local gradients. Since the algorithm detects the waterline based on intensity or colour differences between adjacent pixel regions, even subtle distortion introduced by JPEG compression can shift the perceived transition zone. This is particularly problematic in scenes where the transition is already gradual, such as a sunlit, semi-transparent water surface. These compression artifacts may lead to consistent underestimation of intensity gradients, biasing detection toward lower water levels.

An important consideration in the interpretation of the detection results is the spatial resolution imposed by the camera setup and reference object geometry. In the current field deployment, the pixel-to-length conversion was based on the projected image height of a 500mm wide foam-PVC reference board, positioned at a fixed distance from the camera lens. This setup yielded an estimated vertical resolution of approximately 6.7mm per pixel, defining the smallest detectable step in the waterline position. While this resolution proved sufficient for short-term deployment, it is tied to the physical distance between the camera and the reference surface. Any variation in the effective range (for example, due to mounting constraints at the field site) would alter the pixel scale and affect measurement precision. In general, objects located farther from the lens occupy fewer pixels in the frame, leading to coarser spatial resolution and reduced accuracy. To improve precision, future designs could incorporate either a higher-resolution imaging sensor or a closer mounting position. However, such adjustments may be limited by site-specific installation constraints and trade-offs in field robustness.

Another constraint encountered during testing relates to the limited dynamic range of the current setup. The vertical crop box, as shown in Figure 5.10b, was aligned to the height of the reference board ($\sim$60 cm), meaning that detection was restricted to this narrow vertical window. In operational settings, water levels can vary by several meters, introducing a much broader range of environmental conditions, lighting artifacts, and noise. As this dynamic range increases, so does the likelihood of encountering non-uniform backgrounds and temporary surface disturbances. These factors may raise the system's error floor. Scaling the approach to real-world hydrological dynamics will require rethinking both the spatial extent of the reference structure and the algorithm's capacity to remain robust across a wider vertical detection range.

8.2. Comparison to expectations & literature

8.2.1. Comparison to traditional sensors

The developed camera-based water level monitoring system was designed as a low-cost, low-power alternative to traditional sensor technologies. Compared to radar sensors, which offer sub-centimetre accuracy (typically ±2mm) but come at a cost of $\sim$€1000 [41] and require prominent installation above the water surface, the camera setup sacrifices some precision for significant gains in affordability and resilience. Radar devices, due to their exposed positioning on bridges or elevated structures, are particularly vulnerable to vandalism and direct flood impact [22].

Ultrasonic sensors present a more affordable non-contact solution ($\sim$€100), generally offering accuracies around ±1 cm [23]. However, they are sensitive to surface turbulence [5], temperature variations [6], and wind interference [22], which can degrade performance under natural conditions. More importantly, both ultrasonic and radar sensors require mounting above the water surface—typically on bridges or other overhead structures—where they are exposed to risks such as vandalism and damage during high flood events.

Pressure sensors, commonly installed in submerged housings, offer good accuracy (±1cm) at moderate cost (€300) [30], but are susceptible to damage form extreme events [27], corrosion and flood-related debris [45]. Floater sensors are less expensive ($\sim$€100) [31], but provide only coarse-level or threshold-based detection and are typically unsuitable for continuous environmental monitoring.

In contrast, the system developed in this study achieved 95% accuracy within ±4.64cm, with mean absolute errors of 1.97–2.23 cm depending on the method. While this level of precision does not match that of radar or pressure-based solutions, the trade-off is justified by several advantages: the system is low-cost ($\sim$290€), easily mountable outside flood zones and less visible (less susceptible to vandalism). These characteristics make it especially suitable for decentralized or vandalism-prone settings, where cost and robustness outweigh millimetre-scale accuracy.

8.2.2. Expected diurnal influence versus observed performance

Given the reliance on optical intensity differences, it was anticipated that changing ambient light conditions would influence system accuracy. A previous study [4] observed a decline in accuracy at night, reporting an increase in mean error from 1.8cm during the day to 2.8cm at night, attributed to poor contrast under passive lighting.

In contrast, the present study found that accuracy improved at night. With the use of active infrared illumination, the system maintained stable lighting conditions after sunset. The mean-difference method showed a diurnal bias shift from 2.01cm during the day to 1.00cm at night, while the KS-test method bias values remained 1.02 during both day and night. These findings contradict earlier reports and suggest that active IR lighting can significantly improve nighttime performance, making it more robust than under fluctuating natural daylight conditions. This indicates a design advantage of the current system in handling illumination variability.

8.2.3. Expected influence of precipitation versus observed performance

It was expected that precipitation would reduce detection accuracy, primarily due to occlusion of the reference board by water droplets, splash effects, and increased surface disturbances. This expectation is supported by previous work, which found that heavy rainfall degraded detection accuracy from 1.8cm to 2.6cm during the day, and from 2.8cm to 3.4cm at night [4].

In the present study, performance under precipitation was evaluated for 51 rainy cycles out of a total of 270. Across both the mean-difference and KS-test methods, no consistent increase in bias or MAE was observed. However, visual inspection revealed greater variability in the error signal during rainfall events. Because all rain events occurred during daylight hours (when systematic overestimation was already present due to sunlight penetration) it is not possible to isolate the independent effect of precipitation. Additionally, the limited sample size under rainy conditions constrains the statistical reliability of these findings.

Interestingly, light to moderate rain may actually improve detection under some conditions. Rainfall can increase the diffusivity of the water surface, enhancing the contrast between submerged and dry areas of the reference board, provided the lens remains free from droplets. This may increase the detectability of the waterline. In contrast, during intense rainfall, visibility can deteriorate rapidly, especially when the camera is mounted at a distance. Heavy rain may cause haze, reduce scene contrast, and obscure the reference object. These observations suggest that the effect of precipitation on detection performance is non-linear: while light rain may offer some beneficial contrast effects, heavier rainfall is more likely to degrade image quality.

8.3. Limitations

Prototype modularity versus long-term deployment

The prototype was built for rapid iteration and modular testing rather than extended field service. Components were attached with velcro, adhesive tape, improvised hinges, and plastics and plywood were used without consideration for UV resistance or mechanical fatigue. Although this approach facilitated development and short-term experimentation, it would not withstand the weathering, vibration, and material aging expected in long term deployment.

CLI mode USB-backup incompatibility

During GUI mode testing, a USB flash drive mounted via the Pi's USB port provided reliable data backup. After switching to CLI operation however, mount-point permissions prevented the drive from being accessed automatically. Efforts to resolve the issue were deprioritized in favour of core functionality, leaving the system without a fully autonomous redundancy mechanism when run in its headless mode.

Polarization filter alignment

A circular polarizer was reattached in the field after camera lens calibration to reduce glare, but it was not realigned or calibrated for the new mounting orientation. As a result, specular reflections persisted under low-angle sunlight, causing bright spots that degraded detection accuracy. Future deployments should include a polarization-filter calibration procedure.

Dependence on a reference object

The current algorithm relies on a high-contrast reference board to distinguish the waterline. This dependence increases the risk of vandalism, displacement by high flows, and site-specific installation challenges where a board cannot be mounted. A mitigation strategy could be applying waterproof paint to stable structures like bridge piers that can act as a reference frame. Alternatively, research into AI-based, reference-free detection methods is needed to eliminate this constraint [12][17][40].

Power-system limitations
With the existing 13,400 mAh battery and current software strategy, the system can sustain approximately 26 hours of continuous, autonomous operation. Although a solar-panel-compatible power bank was selected to enable off-grid charging, no panel was deployed during testing. A nano-power timer board *(kiwi-electronics.com)* was prototyped to cut power between measurements and reduce average draw, but it proved unable to reliably re-enable the supply at the end of each rest period. Minor hardware and firmware modifications will be required before dependable power-cycling can be achieved.

The potential benefit of such a system is significant: powering down the Raspberry Pi Zero 2 W during idle periods could reduce hourly energy consumption from 2.73 Wh (current continuous operation) to 0.138 Wh, based on the energy used by six active processing cycles per hour (6 × 0.023 Wh). This yields a theoretical maximum power saving of:

$$\frac{2.73 - 0.138}{2.73} \times 100\% \approx 94.95\%$$

This estimate assumes ideal conditions with instantaneous startup and shutdown, and does not account for boot-up energy peaks or delays. Further empirical testing is needed to assess these factors and determine the practical efficiency gains achievable with a power-cycling strategy.

While solar charging and improved power management remain essential for long-term autonomous deployment, they also introduce new challenges, including risks related to hardware visibility, vandalism, and circuit reliability in variable environmental conditions.

Calibration limitations
Calibration must be repeated whenever the enclosure or camera mount shifts even slightly. At present, realigning the view and obtaining new rotation and crop parameters requires establishing a local connection via a mobile hotspot, which in turn demands physical access to the field site. Two strategies could remove this requirement, though each would result in additional power or computational costs.

First, a secure remote-access interface could be provided by running a lightweight server on the Raspberry Pi. Whenever the detection results begin to drift or on a scheduled basis, an operator could log in remotely, launch the calibration scripts, and update the parameters without visiting the site. However, maintaining an active network service and handling incoming connections would increase both processor usage and standby power draw.

Second, an automatic real-world coordinate feedback loop could be implemented. By calibrating the camera intrinsics and extrinsic, the system could use known scene landmarks to detect any change in orientation. Image processing would estimate the camera's new pose relative to the reference object, compute the corresponding rotation angle and crop box in pixels, and then update the configuration automatically. This approach would require periodic execution of pose-estimation algorithms and matrix transformations, adding to the computational load and further drawing on the battery during each recalibration cycle.

Georeferencing limitations
The developed system reports only pixel changes that reflect the relative position of the water level. To yield absolute z-coordinates, camera calibration would be required. During development, some effort was spent attempting to integrate this capability, but because these techniques are well established, the decision was made not to prioritize them. However, to achieve the system's full potential (supporting flood-early-warning applications) this conversion is necessary and should be explored in future work.

Related to this, the effective centimetre-level accuracy degrades as the distance between the camera and the reference object increases, since fewer pixels span the same physical height. This spatial resolution limitation could be mitigated in part by using a higher-resolution camera module. Additionally, the viewing angle at which the waterline is observed may influence the accuracy, particularly if the line of sight introduces geometric distortion or foreshortening. These factors should be considered in future designs, especially when flexibility in installation height or orientation is required in the field.

Device storage, image posting and connectivity
The prototype currently retains every raw image, rotated and cropped view, and analysis figure per cycle on local flash storage. This was a strategy that suited development but would rapidly exhaust storage space in long-term deployment. Because reliable internet was unavailable at the field site, raw-image uploads were not exercised. Future off-site monitoring will demand a local router or module integrated into the enclosure to push both water-level readings and raw frames to a server. To balance storage and connectivity, only the numerical measurements (including all colour-channel variants) and their corresponding raw images (for model training) should be kept locally, with all intermediate files discarded. A rolling-buffer mechanism on the Pi can enforce a fixed capacity by deleting the oldest files as new data arrive, ensuring backup during temporary network outages while preventing storage overflow.

Automatic outlier rejection
An automated outlier-rejection mechanism could substantially improve detection robustness by filtering questionable waterline estimates. For example, simple boundary-condition checks (such as disallowing changes in water level of more than a few decimetres within a single cycle) would automatically flag and discard implausible readings. More sophisticated rules could e.g. incorporate recent trend information to reject physical impossible detections. Implementing such real-time quality control on the Raspberry Pi would help ensure that only credible measurements are stored and transmitted.

Dynamic measurement frequency
An adaptive sampling strategy could further optimise both data relevance and power consumption by varying the measurement frequency according to observed water-level dynamics. During periods with little change in water level, the system could lengthen the rest interval (thereby conserving battery power) while upon detecting a rapid rise or fall in stage it could temporarily increase the capture rate to provide finer-resolution monitoring. This “event-driven” scheduling could be implemented by comparing successive readings against a threshold or by integrating external rainfall forecasts. Such an approach would align energy expenditure with actual hydrological risk, extending deployment duration without sacrificing critical temporal resolution during flood events.

9

Conclusion

This chapter distils the primary outcomes of the POC, reflecting on its measured accuracy, operational robustness, and cost advantages. It then considers practical lessons learned for real-world deployments and sets the stage for longer-term enhancements and wider adoption.

9.1. Key findings

The prototype water-level monitoring system demonstrated that low-cost, image-based sensing can yield centimetre-level accuracy under real-world conditions. Using a Raspberry Pi Zero 2 W and infrared-assisted imaging, both the mean-difference and KS-test algorithms achieved reliable performance across a multi-day field deployment.

The mean-difference method reached a mean absolute error (MAE) of 1.97cm and a root-mean-square error (RMSE) of 2.79cm, while the KS-test method recorded a slightly higher MAE of 2.23cm and RMSE of 4.92cm. Despite this, the KS-test algorithm demonstrated lower systematic bias (1.02cm vs. 1.57cm) and greater robustness to varying illumination, particularly during daytime. Notably, 95% of errors for both methods fell within ±4.64cm, confirming that the majority of detections stayed within practical error bounds for hydrological monitoring.

Nighttime performance, supported by consistent IR illumination, proved more accurate and stable, with lower median errors and reduced variance. Daytime overestimation was attributed to increased water transparency under direct sunlight, and possibly amplified by JPEG compression artifacts.

In total, 270 detections were analysed. For the mean-difference method, 64.81% of measurements fell within ±2cm and 95.56% within ±4cm. The KS-test method showed similar performance: 65.56% within ±2cm and 91.48% within ±4cm.

The system enclosure withstood light rainfall and moderate field conditions. Power testing indicated that a 13,400mAh battery supported $\sim$26 hours of continuous operation. Potential integration of solar charging and a nano-timer board for hardware-level power management positions the system well for long-term autonomous deployments.

At an estimated total component cost of $\sim$€290, this camera-based solution offers a promising alternative to traditional radar, ultrasonic, pressure, or floater sensors. While it trades some precision (notably under adverse lighting) for lower cost and improved resilience to vandalism and flood damage. The modular, open-source design makes it particularly well suited for budget-constrained or hard-to-access monitoring locations.

9.2. Practical implications

This study demonstrates that compact, camera-based systems offer a viable, low-cost alternative to conventional radar and ultrasonic water-level gauges, particularly in resource-constrained or remote environments. At an estimated cost of ∼€290, the system achieves centimetre-scale accuracy while remaining robust, modular, and energy-efficient. Its discrete mounting position (well above flood-prone areas and out of easy reach) reduces exposure to vandalism and damage, common concerns for traditional sensors installed on bridges or in open water.

Such systems can meaningfully extend the reach of flood early warning and hydrological monitoring networks, especially in areas where conventional infrastructure is impractical or cost prohibitive. Their adaptability also makes them suitable for temporary installations during high-risk periods.

Energy management is a key enabler of long-term autonomous deployment. With a solar-compatible battery pack and plans for a nano-power timer, multi-week or even multi month operation without intervention appears feasible. Optimizing measurement frequency based on hydrological conditions (lengthening sampling intervals during stable flow and shortening them during rapid changes) can further conserve energy while preserving critical data resolution.

To enhance resilience in unattended deployments, the integration of on-board (or possibly centralised) outlier rejection and a circular data buffer is recommended. This would allow the system to discard implausible detections (e.g., sudden unrealistic water level shifts) and manage limited flash storage by automatically overwriting the oldest data. These additions, along with dynamic sampling and solar integration, would move the system closer to a fully autonomous, low-maintenance sensing solution suitable for real-world deployment.

10

Future work

While this proof-of-concept has demonstrated the viability of a low-cost, camera-based water-level gauge, several key areas remain to be addressed before the system can support long-term, autonomous deployments effectively. In particular, the following efforts should be pursued:

1. **Field-scale validation and durability**
    - Conduct multi-month deployments across contrasting sites and seasons to assess environmental robustness.
    - Monitor weathering, UV-degradation and mechanical fatigue of enclosure and mounting hardware.
2. **Camera calibration for absolute elevations**
    - Implement camera intrinsic/extrinsic calibration and use known reference points to convert pixel heights into real-world z-coordinates.
    - Automate pose-estimation routines to update rotation and crop parameters without manual intervention.
3. **Reference-free and ML-based detection**
    - Build a labeled image library (batch of raw frames under varied illumination, turbidity, vegetation cover) for training deep-learning models to detect waterlines without a physical board [11].
    - Evaluate state-of-the-art segmentation or edge detection networks for waterline detection.
4. **Advanced power management & solar integration**
    - Finalize integration of the nano-power-timer to reliably power-cycle the Pi between measurements.
    - Pair the optimized timer with a small solar panel, then field-test energy self-sufficiency over long term deployment.
5. **Storage, connectivity & data handling**
    - Develop an on-device rolling-buffer scheme to retain only raw images and numeric measurements, deleting intermediate files when capacity is reached.
    - Integrate an internet modem to upload both water-level data and selected raw frames in near-real time.
6. **Dynamic sampling & automatic outlier rejection**
    - Implement adaptive sampling intervals (e.g. lower frequency during low risk periods, higher when rapid changes detected).
    - Embed statistical or rule-based filters (e.g. maximum plausible water-level change per interval) to automatically discard questionable detections.

References

[1] Agile Business Consortium. *MoSCoW Prioritisation*. Accessed: 2025-06-10. 2025. URL: https://www.agilebusiness.org/dsdm-project-framework/moscow-prioririsation.html.

[2] AmebaIoT. *PowerMode – Deep Sleep Mode*. Accessed: 2025-06-02. 2021. URL: https://www.amebaiot.com/en/amebapro2-arduino-deepsleep/.

[3] Jean-Luc Aufranc. *A deep dive into Raspberry Pi Zero 2 W's power consumption*. Accessed: 2025-06-02. 2021. URL: https://www.cnx-software.com/2021/12/09/raspberry-pi-zero-2-w-power-consumption/.

[4] Joaquim Amândio Azevedo and João André Brás. "Measurement of Water Level in Urban Streams under Bad Weather Conditions". In: *Sensors* (2021). ISSN: 1424-8220. DOI: 10.3390/s21217157. URL: https://www.mdpi.com/1424-8220/21/21/7157.

[5] Inhyeok Bae and Un Ji. "Outlier detection and smoothing process for water level data measured by ultrasonic sensor in stream flows". In: *Water (Switzerland)* 11.5 (2019). ISSN: 20734441. DOI: 10.3390/w11050951.

[6] Ganggang Bai et al. "An intelligent water level monitoring method based on SSD algorithm". In: *Measurement: Journal of the International Measurement Confederation* 185 (2021). ISSN: 02632241. DOI: 10.1016/j.measurement.2021.110047.

[7] Bscan. *KS Example*. https://commons.wikimedia.org/wiki/File:KS_Example.png. CC0 license, via Wikimedia Commons. Accessed: 2025-06-02. 2013.

[8] Chen Chen et al. "An Integrated Method for River Water Level Recognition from Surveillance Images Using Convolution Neural Networks". In: *Remote Sensing* 14.23 (2022). ISSN: 20724292. DOI: 10.3390/rs14236023.

[9] Liz Clark. *Adafruit MOSFET Driver*. https://cdn-learn.adafruit.com/downloads/pdf/adafruit-mosfet-driver.pdf. Accessed: 2025-06-02. This guide details the Adafruit STEMMA MOSFET Driver, featuring an AO3406 N-Channel MOSFET rated for 30Vds and 3.6A peak, suitable for driving motors, solenoids, and LEDs. 2024.

[10] Elektor. *Raspberry Pi NoIR Camera Module v2*. https://www.elektor.com/products/raspberry-pi-noir-camera-module-v2. Accessed: 2025-06-02. 2025.

[11] Anette Eltner et al. "Automatic Image-Based Water Stage Measurement for Long-Term Observations in Ungauged Catchments". In: *Water Resources Research* 54.12 (2018). ISSN: 19447973. DOI: 10.1029/2018WR023913.

[12] Anette Eltner et al. "Using Deep Learning for Automatic Water Stage Measurements". In: *Water Resources Research* 57.3 (2021). ISSN: 19447973. DOI: 10.1029/2020WR027608.

[13] Simon Etter et al. "Quality and timing of crowd-based water level class observations". In: *Hydrological Processes* 34.22 (2020). ISSN: 10991085. DOI: 10.1002/hyp.13864.

[14] Farnell. *IR Emitter Module Datasheet*. https://www.farnell.com/datasheets/3164632.pdf. Accessed: 2025-06-02. 2023.

[15] Francisco E. Fernandes, Luis Gustavo Nonato, and Jó Ueyama. "A river flooding detection system based on deep learning and computer vision". In: *Multimedia Tools and Applications* 81.28 (2022). ISSN: 15737721. DOI: 10.1007/s11042-022-12813-3.

[16] Intelligent LED Solutions. *ILS Aluminium Alloy Heatsinks for PowerStars and PowerClusters*. https://www.farnell.com/datasheets/3161532.pdf. Accessed: 2025-06-02. Includes models such as ILA-HSINK-STAR-50X20MM, ILA-HSINK-STAR-50X40MM, ILA-HSINK-STAR-50X60MM, ILA-HSINK-STAR-50X80MM, ILA-HSINK-70X70X55MM, and ILA-HSINK-78X46X25MM. Supplied with fixing screws and pre-applied Thermal Interface Material (TIM). 2019.

[17] Navid H. Jafari et al. “Real-time water level monitoring using live cameras and computer vision techniques”. In: *Computers and Geosciences* 147 (2021). ISSN: 00983004. DOI: 10.1016/j.cageo.2020.104642.

[18] Frans Klijn et al. “Adaptive flood risk management planning based on a comprehensive flood risk conceptualisation”. In: *Mitigation and Adaptation Strategies for Global Change* 20.6 (2015). ISSN: 15731596. DOI: 10.1007/s11027-015-9638-z.

[19] C. J. Koblinsky et al. “Measurement of river level variations with satellite altimetry”. In: *Water Resources Research* 29.6 (1993). ISSN: 19447973. DOI: 10.1029/93WR00542.

[20] Lung Chih Kuo and Cheng Chi Tai. “Automatic water-level measurement system for confined-space applications”. In: *Review of Scientific Instruments* 92.8 (2021). ISSN: 10897623. DOI: 10.1063/5.0046804.

[21] Yan Ting Lin, Yi Chun Lin, and Jen Yu Han. “Automatic water-level detection using single-camera images with varied poses”. In: *Measurement: Journal of the International Measurement Confederation* 127 (2018). ISSN: 02632241. DOI: 10.1016/j.measurement.2018.05.100.

[22] Wen Cheng Liu and Wei Che Huang. “Evaluation of deep learning computer vision for water level measurements in rivers”. In: *Heliyon* 10.4 (2024). ISSN: 24058440. DOI: 10.1016/j.heliyon.2024.e25989.

[23] MaxBotix Inc. *MB7052 XL-MaxSonar-WRMAT Ultrasonic Sensor*. Accessed: 2025-06-01. 2024. URL: https://maxbotix.com/products/mb7052.

[24] Multicomp Pro. *Wire Wound Fixed Resistors Datasheet*. https://www.farnell.com/datasheets/3916833.pdf. Accessed: 2025-06-02. The datasheet covers various models including MCKNP series resistors with power ratings from 0.5W to 10W, resistance values ranging from 0.1Ω to 50kΩ, and tolerances of ±1% and ±5%. Features include flame-resistant coating, non-inductive options, and compliance with E24 and E96 series standards. 2023.

[25] OpenRiverCam. *OpenRiverCam Platform*. Accessed: 2025-06-02. 2025. URL: https://openrivercam.com/.

[26] Jinqiu Pan et al. “Deep learning-based unmanned surveillance systems for observing water levels”. In: *IEEE Access* 6 (2018). ISSN: 21693536. DOI: 10.1109/ACCESS.2018.2883702.

[27] Jonathan D. Paul, Wouter Buytaert, and Neeraj Sah. “A Technical Evaluation of Lidar-Based Measurement of River Water Levels”. In: *Water Resources Research* 56.4 (2020). ISSN: 19447973. DOI: 10.1029/2019WR026810.

[28] Raspberry Pi Foundation. *Raspberry Pi NoIR Camera Module v2*. https://www.raspberrypi.com/products/pi-noir-camera-v2/. Part number: RPI NOIR CAMERA BOARD. 8 MP Sony IMX219 image sensor. Supports 1080p30, 720p60, and 640x480p90 video. 2023.

[29] Raspberry Pi Ltd. *Raspberry Pi Zero 2 W Product Brief*. https://datasheets.raspberrypi.com/rpizero2/raspberry-pi-zero-2-w-product-brief.pdf. Accessed: 2025-06-02. 2024.

[30] RS Components. *Capacitive Liquid Level Sensor for Non-metallic Containers, RS Stock No. 2726967*. Accessed: 2025-06-01. 2024. URL: https://nl.rs-online.com/web/p/level-sensors/2726967.

[31] RS Online. *Float Level Sensor, Horizontal, Plastic Stem*. Accessed: 2025-06-01. 2024. URL: https://nl.rs-online.com/web/p/level-sensors/2726967.

[32] Schneider Electric. *Thalassa TBS ABS Enclosure NSYTBS191610H*. https://www.se.com/nl/nl/product/NSYTBS191610H/tbs-tbp-doos-abs-+-deksel-193x164x105mm-ip66-ik07-h40-ral7035/. Accessed: 2025-06-02. ABS enclosure with dimensions 193x164x105 mm, IP66, IK07, RAL7035, suitable for outdoor use. 2025.

[33] Gerhard Schoener. “Time-Lapse Photography: Low-Cost, Low-Tech Alternative for Monitoring Flow Depth”. In: *Journal of Hydrologic Engineering* 23.2 (2018). ISSN: 1084-0699. DOI: 10.1061/(asce)he.1943-5584.0001616.

[34] Lorenzo Steccanella et al. “Deep learning waterline detection for low-cost autonomous boats”. In: *Advances in Intelligent Systems and Computing*. Vol. 867. 2019. DOI: 10.1007/978-3-030-01370-7{_}48.

[35] Wenchao Sun, Hiroshi Ishidaira, and Satish Bastola. “Calibration of hydrological models in ungauged basins based on satellite radar altimetry observations of river water level”. In: *Hydrological Processes* 26.23 (2012). ISSN: 08856087. DOI: 10.1002/hyp.8429.

[36] M. J. Tourian et al. “Spatiotemporal densification of river water level time series by multimission satellite altimetry”. In: *Water Resources Research* 52.2 (2016). ISSN: 19447973. DOI: 10.1002/2015WR017654.

[37] UwCamera.nl. *37mm MRC CPL Filter (16 Lagen) / Waterafstotend Multi-Coated CPL Lensfilter*. https://www.uwcamera.nl/37mm-MRC-CPL-Filter-(16-lagen)-/-Waterafstotend-Multi-Coated-CPL-Lensfilter. Accessed: 2025-06-02. Features a 16-layer multi-resistant coating that reduces reflections and enhances color saturation. The hydrophobic coating repels water, oil, and dust, making it suitable for outdoor photography. The filter is rotatable to adjust polarization effects and is compatible with lenses having a 37mm thread. 2025.

[38] Annemiek van Boeijen et al. *Delft Design Guide: Design strategies and methods*. English. BIS Publishers, 2013.

[39] Remy Vandaele, Sarah L. Dance, and Varun Ojha. “Deep learning for automated river-level monitoring through river-camera images: An approach based on water segmentation and transfer learning”. In: *Hydrology and Earth System Sciences* 25.8 (2021). ISSN: 16077938. DOI: 10.5194/hess-25-4435-2021.

[40] Ryan L. Vanden Boomen, Zeyun Yu, and Qian Liao. “Application of Deep Learning for Image-Based Stream Gaging”. In: *Water Resources Research* 57.11 (2021). ISSN: 19447973. DOI: 10.1029/2021WR029980.

[41] VEGA Grieshaber KG. *VEGAPULS C 23 Radar Level Sensor*. Accessed: 2025-06-01. 2024. URL: https://www.vega.com/en-us/products/product-catalog/level/radar/vegapuls-c-23.

[42] Visual Crossing Corporation. *Weather Query Builder*. https://www.visualcrossing.com/weather-query-builder/. Accessed: 2025-06-02. 2025.

[43] Voltaic Systems. *6 Watt 6 Volt Solar Panel (P106)*. https://voltaicsystems.com/6-watt-panel/. Accessed: 2025-06-02. Features: IP67 waterproof rating, UV-resistant urethane coating, 19% efficient monocrystalline cells, 6.0W peak power at 6.5V, 930mA peak current, dimensions: 17.5 x 22.1 x 0.5 cm, weight: 255g. 2025.

[44] Voltaic Systems. *V50 USB Battery Pack*. https://voltaicsystems.com/v50/. Accessed: 2025-06-02. 13,400mAh capacity, 48Wh Li-Ion battery with Always On mode, dual USB-A outputs, USB-C PD input/output, optimized for solar charging. 2025.

[45] Sheng Wei Wang et al. “A continuous water-level sensor based on load cell and floating pipe”. In: *Proceedings of 4th IEEE International Conference on Applied System Innovation 2018, ICASI 2018*. 2018. DOI: 10.1109/ICASI.2018.8394554.

[46] W. van de Westeringh. *Soils and their Geology in the Geul Valley*. Tech. rep. 80-8. Accessed: 2025-06-02. Landbouwhogeschool Wageningen, 1980. URL: https://edepot.wur.nl/467976.

[47] Fan Wu, Shih-Wen Hsiao, and Peng Lu. “An AIGC-empowered methodology to product color matching design”. In: *Displays* 81 (2024), p. 102623. ISSN: 0141-9382. DOI: https://doi.org/10.1016/j.displa.2023.102623. URL: https://www.sciencedirect.com/science/article/pii/S0141938223002573.

[48] Di Zhang and Junyan Tong. “Robust water level measurement method based on computer vision”. In: *Journal of Hydrology* 620 (2023). ISSN: 00221694. DOI: 10.1016/j.jhydrol.2023.129456.