Colorimetric Sensor

Xuanye Zeng

3rd Year Project Final Report

Department of Electronic &
Electrical Engineering

UCL

Supervisor: Dr. Oliver Hadeler

5 April 2025

I have read and understood UCL's and the Department's statements and guidelines concerning plagiarism.

I declare that all material described in this report is my own work except where explicitly and individually indicated in the text. This includes ideas described in the text, figures and computer programs.

I acknowledge the use of Chat-GPT-4o by OpenAI(https://chat.openai.com) to summarise my initial notes, structure LaTeX layouts, and proofread my final draft.

This report contains 32 pages (excluding this page and the appendices) and 10755 words.

Signed: Xuanye Zeng Date: 09/04/2025

(Student)

Colorimetric Sensor

Xuanye Zeng

This project presents the development of a low-cost, portable colorimetric pH sensor that automates the reading of pH test strips using a Raspberry Pi-based imaging system. The device integrates controlled white LED illumination, a Raspberry Pi Camera, and an image processing algorithm to objectively determine pH from the color of a test strip. In operation, a user places a pH test strip into the device, presses a button to capture an image under consistent lighting, and the system computes the pH by comparing the strip's color in CIE L*a*b* color space to pre-calibrated reference values. The measured pH is then displayed on an LCD screen. The sensor achieved a mean absolute error of approximately 0.28 pH units with a low mean standard deviation of 0.057 and a correlation coefficient R^2 of 0.985 across a range of test solutions. These results demonstrate that the device can reliably distinguish pH-induced color changes, reducing subjective interpretation errors and providing quantitative readings. The project's outcome is a proof-of-concept for field-portable colorimetric analysis, with potential applications in water quality monitoring and chemical analysis where quick and accurate pH assessment is required.

Contents

1 Introduction and Literature Review

Monitoring pH is crucial in fields ranging from environmental water quality assessment to chemical analysis in laboratories. In environmental monitoring, for example, the pH of freshwater ecosystems must be tracked to detect issues like acidification that can harm aquatic life[1]. Traditionally, pH test strips are a simple and cost-effective tool for such measurements: the strip's chemical indicator changes color according to the solution's pH, and the user compares this color to a printed reference chart. However, manual interpretation of test strip colors is subjective and prone to error, especially under inconsistent ambient lighting or with subtle color differences between pH levels[2]. Such inconsistencies can lead to inaccurate readings, potentially compromising decisions in public health, environmental management, or industrial processes. These limitations motivate the development of a digital colorimetric pH sensor that can provide objective, reliable readings in a variety of conditions.

Currently, the main analytical techniques for water quality detection are electrochemical, spectroscopic, and colorimetric methods. While spectroscopic and electrochemical approaches provide high accuracy, they involve high costs, lengthy processing, and complex operations, making them unsuitable for field deployment[3]. By contrast, colorimetric techniques have gained attention for their rapid detection and low reagent consumption[4]. Recent advances in low-cost electronics and computer vision have opened opportunities to automate colorimetric analyses. A common approach is to use controlled illumination and a camera to capture the color of reagent, then apply image processing to quantify the color change[5]. Several research efforts have demonstrated the feasibility of such systems. Zhang et al.(2024) developed a smartphone- and papersensor-based multitask colorimetric sensor with multiple reaction zones for water quality monitoring, employing deep learning to detect various analytes' color changes[6]. Their device achieved high precision and recall, highlighting how automation and advanced algorithms can enhance the reliability of on-site measurements. However, their system remains vulnerable to external lighting conditions, variations in smartphone camera quality, and the need for consistent sensor positioning. Similarly, Luka et al.(2017) introduced a low-cost, portable device for detecting pH and nitrite in water samples. Their design incorporated a Raspberry Pi camera and computational analysis to minimize the influence of ambient light and human error, demonstrating improved accuracy in colorimetric readings[4]. Nevertheless, the device's reliance on cuvettes poses challenges for field use, such as requiring a stable surface and risking sample spillage. In another study, Yang et al.(2022) presented a Raspberry Pi-based analyzer for urine test strips using HSV color space, which successfully determined multiple indicators by analyzing strip colors in a controlled setup[2]. These examples illustrate a trend toward integrating single-board computers and imaging for chemical sensing, achieving performance comparable to laboratory instruments but in field-friendly formats.

While prior works have addressed general colorimetric sensing and even multi-analyte detection with sophisticated methods, this project focuses specifically on pH sensing and explores a simpler algorithmic approach. The aim is to develop a dedicated colorimetric pH sensor that is highly accessible and emphasizes ease of use and reproducibility. Compared with smartphone-based system, the proposed design employs a stable internal light source and a fixed sample-to-camera distance within a controlled enclosure, thereby increasing the repeatability and reliability of colorimetric analysis. Unlike deep learning techniques that may require large training datasets and considerable processing power, our approach leverages a perceptually uniform color space (CIE L*a*b*), which is more effective than the simpler HSV approach, and a nearest-neighbor interpolation algorithm to translate color into pH. By focusing on a single parameter (pH) and optimizing the hardware for that purpose, we seek to achieve reliable accuracy with minimal complexity. This project builds on the insights from the strengths of previous approaches - such as the importance of controlled lighting and calibration – while keeping the system simple, affordable, and focused on the single task of pH sensing. The following sections detail the project goals, the theoretical background of the techniques used, the design and methods implemented, and an evaluation of the system's performance.

2 Project Goals

The project was guided by three primary goals, each with specific objectives to ensure a systematic development of the colorimetric pH sensor:

1. Design and Construct a Portable Hardware System:

   Create an enclosure and hardware setup that enables consistent, repeatable imaging of pH test strips in the field.

   - Objective 1.1: Develop a stable lighting environment using white LEDs and a closed housing to eliminate external light variability.
   - Objective 1.2: Integrate a Raspberry Pi 4, camera module, and necessary interface components (LCD display and push button) into a compact power-bank-powered enclosure.
   - Objective 1.3: Implement a simple user interface – a button to initiate/reset measurement and an LCD to display results – for ease of use by non-specialist users.

2. Implement Robust Image Processing and pH Calculation Algorithms:

   Develop a software pipeline to accurately extract the strip's color and determine pH from the captured image.

   - Objective 2.1: Write an image processing routine to isolate the pH test strip region within the photo under varying conditions, including background segmentation and noise reduction.
   - Objective 2.2: Apply appropriate color space conversion and analysis techniques (e.g. converting to L*a*b* color space and using clustering) to obtain a representative color value of the strip's region of interest (ROI).
   - Objective 2.3: Calibrate and map the detected color to a pH value. Using a pre-recorded reference table of color–pH correspondences (obtained

from known pH standards), implement an algorithm (such as nearest-neighbor interpolation in color space) to estimate the pH from the ROI's color.

3. Validate Accuracy and Reliability through Testing:

   Experimentally evaluate the device with known pH solutions to verify its performance and identify areas for improvement.

   - Objective 3.1: Calibrate the system using the reference pH color card provided by test strip manufacturer to build a reliable color-to-pH reference dataset.
   - Objective 3.2: Test the controlled lighting environment and the effect of measurement timing on pH readings using samples of known pH, to assess both the reliability of the enclosure and the optimal time window for reading the strip's pH value.
   - Objective 3.3: Test the fully integrated system on multiple samples of known pH (acidic, neutral, alkaline) to assess measurement accuracy.

These goals and objectives ensured a balanced approach covering hardware development, software algorithm design, and empirical validation. By meeting them, the project aimed to deliver a functional prototype and demonstrate its effectiveness in measuring pH.

3 Technical Background

3.1 Colorimetric pH Sensing Principles

Colorimetric pH strips rely on pH-sensitive indicator chemicals that undergo a color change in response to the hydrogen ion concentration of a solution[7]. Each test strip is typically impregnated with one or more indicators that produce a particular color at a given pH value. The resulting color is then matched to a chart to estimate the pH. The science behind this is rooted in acid-base chemistry and the equilibrium of indicators, but for the purpose of a sensing device, the key point is that each pH corresponds to a characteristic color or range of colors on the strip. However, these colors are not perfectly distinct for every integer pH; small changes in pH can lead to subtle shifts in hue or brightness, and different indicators cover different sub-ranges of the pH spectrum. Therefore, accurately reading a colorimetric strip requires distinguishing these subtle color differences[17].

In a digital color sensor, the task is to measure the strip's color as numerical data. A camera acts as a color sensor, capturing the strip's appearance under controlled lighting. The raw output of a typical camera is in the RGB color model (red, green, blue values for each pixel). However, directly using RGB values to quantify color differences can be problematic. The RGB space is not perceptually uniform – meaning a given numerical difference in RGB does not consistently correspond to the same perceived color difference across the spectrum. Additionally, RGB mixes color and brightness together, which can complicate comparisons[16].

3.2 Color Spaces and Perceptual Uniformity

To address these issues, the project uses the CIE L*a*b* color space for color analysis. CIE L*a*b* is designed to be approximately perceptually uniform, which

implies that the Euclidean distance between two colors in Lab space (ΔE) correlates better with human perceptual difference than differences in RGB[8, 15]. The conversion process consists of three main steps:

1. **Gamma Correction**

   Since RGB values are nonlinearly encoded, they must be linearized. For each channel C (where C represents R, G, or B normalized to [0,1]), we use:

$$C_{\mathbf{lin}} = \begin{cases} \frac{C}{12.92}, & C \leq 0.04045, \\ \left(\frac{C+0.055}{1.055}\right)^{2.4}, & C > 0.04045. \end{cases} \tag{1}$$

   This piecewise function corrects for gamma compression, providing linear intensity values.

2. **Linear RGB to CIE XYZ**

   The linear RGB values are then mapped to the device-independent CIE XYZ space using a matrix transformation (for D65 illuminant):

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{pmatrix} 0.4124 & 0.3576 & 0.1805 \\ 0.2126 & 0.7152 & 0.0722 \\ 0.0193 & 0.1192 & 0.9505 \end{pmatrix} \begin{pmatrix} R_{\mathbf{lin}} \\ G_{\mathbf{lin}} \\ B_{\mathbf{lin}} \end{pmatrix}. \tag{2}$$

   This step aligns the colors with the human visual response by using standardized color-matching functions.

3. **CIE XYZ to CIE Lab**

   Finally, the XYZ values are normalized by a reference white point (X_n, Y_n, Z_n) and converted to CIE Lab via:

$$L^* = 116\, f\left(\frac{Y}{Y_n}\right) - 16, \tag{3}$$

$$a^* = 500 \left[f\left(\frac{X}{X_n}\right) - f\left(\frac{Y}{Y_n}\right) \right], \tag{4}$$

$$b^* = 200 \left[f\left(\frac{Y}{Y_n}\right) - f\left(\frac{Z}{Z_n}\right) \right], \tag{5}$$

   where

$$f(t) = \begin{cases} t^{\frac{1}{3}}, & t > \delta^3, \\ \frac{t}{3\,\delta^2} + \frac{4}{29}, & t \leq \delta^3, \end{cases} \quad \textbf{with} \quad \delta = \frac{6}{29}. \tag{6}$$

   This nonlinear transformation compensates for the human visual system's nonlinear perception of brightness, yielding a more perceptually uniform space.

In the Lab model, L* represents lightness (luminance), and a* and b* represent color-opponent dimensions: roughly green–red and blue–yellow components, respectively. By separating lightness from chromaticity, we can handle varying illumination and reflection better – a critical factor since small lighting differences or shadows could change a strip's brightness but not its inherent hue. Working in Lab space allows the system to compare colors in a way that is closer to how the human eye would perceive differences, which is ideal for matching the strip color

to references[24].

In Lab space, the difference between a measured color (with coordinates L_1, a_1, b_1) and a reference color (L_2, a_2, b_2) can be quantified by the Euclidean distance:

$$\Delta E_{Lab} = \sqrt{(L_1 - L_2)^2 + (a_1 - a_2)^2 + (b_1 - b_2)^2}\,. \tag{7}$$

This ΔE metric is used in our sensor to find the closest matching reference color for the strip's reaction pad. A smaller ΔE means a more similar color. The Lab color space's relative uniformity makes this a reliable way to determine which pH reference color the unknown sample is nearest to.

3.3 Image Processing Techniques

Image preprocessing is necessary to isolate the test strip in the camera's field of view. Subsection 4.3 shows the relevant images and the detailed workflow. The first step is to convert the target image into grayscale to reduce complexity and emphases the intensity differences, making it easier to identify target area of the test strip. Grayscale conversion combines the three color channels (RGB) into a single intensity value by applying a weighted sum. The formula for grayscale conversion is[18]:

$$I = 0.2989R + 0.5870G + 0.1140B \tag{8}$$

Where I is Grayscale intensity value. R, G, and B are Red, Green, and Blue channel values respectively (0–255). And the coefficients 0.2989, 0.5870, and 0.1140 represent the relative luminance contributions of each channel.

The grayscale image can then be converted into black-and-white images through thresholding, which converts an image to a binary form in order to separate a light background from a darker strip. A global threshold may not always be adequate if lighting is uneven, so adaptive thresholding is utilized. Mathematically, in adaptive (Gaussian) thresholding $T(x, y)$, each pixel (x, y) uses a threshold computed from its local neighborhood. Specifically,

$$T(x, y) = \big(\textbf{local weighted mean}\big) - C \tag{9}$$

where the local weighted mean is often determined using a Gaussian window so that nearby pixels have a higher influence on the threshold, and C is a constant subtracted to fine-tune the threshold. By calculating thresholds locally across the image, adaptive thresholding compensates for gradients in illumination, ensuring that the white background (on which the strip is placed) can be distinguished from the colored strip itself even under non-uniform lighting conditions.

After thresholding, morphological operations (from mathematical morphology) like closing and opening are applied to the resulting binary image[11]. In mathematical terms:

- Closing is defined as:

$$\textbf{Closing}(I) = \big(I \oplus S\big) \ominus S, \tag{10}$$

  where I is the binary image, S is the structuring element (e.g., a 3×3 ellipse), $\oplus$ denotes dilation, and $\ominus$ denotes erosion. Closing helps fill small holes and gaps within the detected regions, ensuring that the strip's area is contiguous.

- **Opening is defined as:**

$$\textbf{Opening}(I) = \big(I \ominus S\big) \oplus S. \tag{11}$$

Opening removes small noise specks (isolated white pixels) by first eroding and then dilating the image. Both operations clean up the binary mask so that finding the strip's contour becomes more robust.

To accurately determine the strip's color, it's important to focus on the area of the strip containing the indicator pad (where color change occurs), avoiding edges or partially soaked regions. After finding the largest contour corresponding to the strip, the system scales that contour inward by a certain factor (e.g.,70%). This inward scaling is performed around the centroid of the contour, which is computed from image moments. Hence, the $p-q$ moments of the contour is calculated, defined as:

$$M_{pq} = \sum_{(x,y)\in Image} x^p y^q I(x,\ y) \tag{12}$$

where x and y are the horizontal and vertical pixel coordinates, $I(x,\ y)$ is the intensity of the pixel at $(x,\ y)$, and p and q are the exponents that define the specific moment being calculated. The centroid (c_x, c_y) can then be calculated as:

$$c_x = \frac{M_{10}}{M_{00}}, \quad c_y = \frac{M_{01}}{M_{00}} \tag{13}$$

where M_{00} is the total number of pixels within the area of the object, M_{10} is the the total sum of horizontal pixel positions weighted by intensity and M_{01} is the total sum of vertical pixel positions weighted by intensity. Each point (x, y) on the contour is then scaled relative to the centroid (c_x, c_y) by a factor s (where $0 < s < 1$):

$$x_{\textbf{new}} = (x - c_x) \times s + c_x, \quad y_{\textbf{new}} = (y - c_y) \times s + c_y. \tag{14}$$

This technique yields a region of interest (ROI) focused on the central portion of the strip's pad, which is more uniformly colored by the chemical reaction. By excluding the outer edges, the measurement reduces the influence of any gradients or partial coloration at the strip margins, leading to a more reliable estimation of the strip's true color.

3.4 Dominant Color Extraction via Clustering

Within the region of interest (ROI), individual pixels often exhibit slight color variations due to texture, lighting gradients, or sensor noise. Moreover, any portion of the strip held by hand does not undergo the chemical reaction and should be excluded from color determination. Rather than averaging all pixels directly, which could be skewed by any outliers, the project employs a K-means clustering algorithm to find the dominant color in the ROI[2, 10].

Mathematically, K-means clustering is an unsupervised machine learning method that partitions the pixel data $\{x_i\}$ (where each x_i is a three-dimensional Lab value) into K clusters by minimizing the within-cluster sum of squared distances:

$$\min_{\{\mu_j\}} \sum_{i=1}^{N} \min_{j} \|\mathbf{x}_i - \mu_j\|^2 \tag{15}$$

where μ_j are the centroid (mean) Lab values for each of the K clusters, and N is the total number of pixels in the ROI. The algorithm proceeds iteratively:

1. **Assignment Step: Each pixel x_i is assigned to the closest cluster center μ_j for which the Euclidean distance $\|x_i - \mu_j\|$ is smallest.**

2. **Update Step: Each cluster center μ_j is recalculated as the mean of the pixels currently assigned to it.**

This assignment-update process repeats until the cluster centers no longer change significantly or until a maximum number of iterations is reached.

We chose $K = 3$ clusters as a balance between capturing multiple color components if present and filtering out noise. Essentially, the algorithm groups pixels into three clusters in Lab space and identifies their centers (mean Lab values). Typically, in an image of a mostly uniform color area, one cluster contains the majority of pixels representing the main color of the indicator pad, while the other clusters capture slight shadows or non-reaction regions if any are included.The dominant cluster is identified as the one with the most pixels. Its centroid (the average Lab value in that cluster) is then taken as the representative color of the strip.

3.5 Calibration and Color Matching Algorithm

To translate a measured color into pH, the system needs a reference. This requires calibration with known standards. Prior to using the device for unknown samples, a reference database of Lab color values for known pH levels (1 through 14) was established. This was done by capturing images of standard pH color reference cards under the same lighting and setup as the actual measurements. For each pH point, the Lab coordinates of the indicator's color were recorded. The outcome is a set of reference color vectors $C_{pH1}, C_{pH2}, \ldots, C_{pH14}$ in Lab space that serve as the expected colors for each integer pH. For example, Figure 9 in section 5.1 shows plots of CIE L*a*b* values for the reference card.

Given a measured color C_{meas} from the strip, the algorithm computes the distance ΔE (as shown in equation 1) to each reference C_{pHi}. The two closest matches (i.e. the two smallest ΔE values) are identified[9]. Let these correspond to pH values pH_a and pH_b (for example, the color might lie between the reference for pH6 and pH7). If d_a and d_b are the distances to these two nearest reference colors, we calculate the estimated pH by an interpolation weighted by the inverse of these distances:

$$pH_{est} \;=\; \frac{d_b}{d_a + d_b} \cdot pH_a \;+\; \frac{d_a}{d_a + d_b} \cdot pH_b \,. \tag{16}$$

This formula essentially says that the final pH estimate will be closer to the pH of whichever reference color is closer in Lab space. For instance, if the strip's color is almost exactly matching the pH7 reference and far from pH6, then $d_a \ll d_b$, and pH_{est} will be very close to 7. On the other hand, if the color lies about midway between the typical colors of pH 6 and 7, the estimate will come out around 6.5. This linear interpolation in color space assumes that small variations in color between two known pH points correspond roughly to proportional mixtures of those pH effects. While not a perfect physical model, this approach is justified by the fact that a smooth change in the indicator's color is expected as pH changes incrementally.

It is important to note that the relationship between pH and color is not strictly linear or monotonic across the full range[9] – an observation from our calibration is

that no single color channel (L*, a*, or b*) varies in a strictly one-directional way from pH 1 to 14. This makes it difficult to fit a simple global equation (like a polynomial) to predict pH from color values. Thus, the nearest-neighbor interpolation strategy was chosen for its simplicity and effectiveness: it makes no assumption of global linearity and instead relies on local linearity between adjacent reference points. This is a form of piecewise calibration, which tends to be reliable as long as the reference set is dense enough to capture the major color transitions of the indicator.

3.6 Hardware and Materials

A Raspberry Pi 4 serves as the primary processing unit due to its low cost, compact size, and extensive open-source library support. It interfaces directly with the Pi Camera Module and executes the image-analysis algorithms. Illumination is provided by white Light Emitting Diodes (LEDs), chosen for their stable light output and sufficient brightness. Because direct LED light can produce glare or uneven illumination, a thin diffuser (translucent sheet) is placed between the LEDs and the test strip to ensure a uniform illumination field, thereby minimizing reflections on the strip's wet surface.

In addition, the Raspberry Pi Camera's internal parameters (e.g., exposure and white balance) are set to fixed values during both calibration and measurement to preserve consistency. For instance, automatic white balance is locked following calibration to avoid unexpected color shifts that could compromise measurement accuracy.

For user interaction and feedback, the system includes a 16×2 character LCD module connected via an I^2C interface, enabling the display of the measured pH value. A single push-button is wired to a Raspberry Pi GPIO pin through a pull-down resistor, with debouncing handled in software. When pressed, it initiates the measurement cycle. These straightforward interface elements make the device self-contained and user-friendly, qualities important for any practical field-based instrument.

All onboard electronics are powered by a 20,000mAh lithium polymer power bank, which directly connects to the Raspberry Pi board. This setup isolates the power supply from the rest of the device and allows easy swapping of batteries for extended in-field usage.

To ensure illumination uniformity and minimize external lighting interference, the device is housed in a 3D-printed enclosure with dimensions of 110×80×254 mm (length × width × height) as shown in Figure 1a. Constructed from black polylactic acid (PLA), the enclosure minimizes internal light reflections while offering ease of use and durability. The bottom interior is lined with white paper to create contrast against the test strip, thereby facilitating accurate detection during image processing. Figure 1b is the cross-sectional view of the device, which indicates where all the electronics are housed inside the enclosure. The pH test strips utilized in this project are sourced from Lansonee, as shown in Figure 1c, and the buffer solution powders (pH4.00, pH6.92, and pH9.33) are provided by BOJACK. These buffer solutions were prepared by serially dissolving the samples in 250mL of deionized water. Costs of major components are summarized in Table 1.

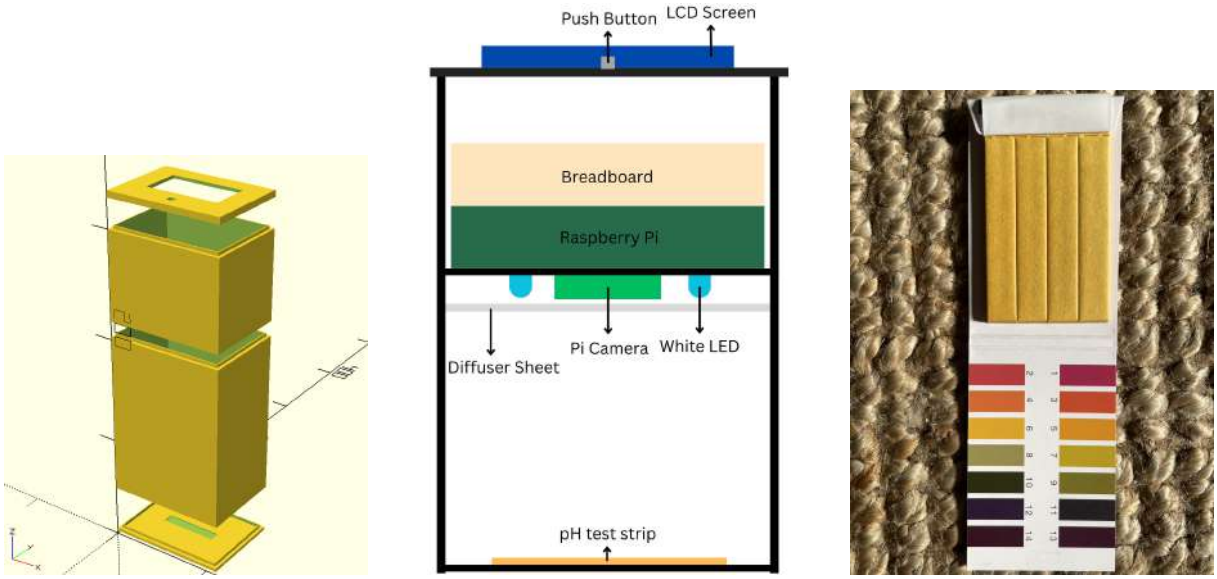


Figure 1: (a) 3D overview of the enclosure. (b) Cross-sectional view of the device. (c) pH test strips with reference card.

Table 1: Bill of Materials

Component	Quantity	Cost
Raspberry Pi 4 Model B	1	£33.30
Raspberry Pi Camera Module 2	1	£14.00
I2C 16x2 LCD Display Module	1	£11.80
2Pin Tactile Push Button	1	£0.05
White LED 5mm	8	£1.20
PLA 3D Printing Material	–	£2.20
Total		£62.55

4 Methods

In this section, we describe the practical implementation of the colorimetric pH sensor, including the hardware setup and the software methodology for image processing and analysis. We also outline the experimental procedure for calibrating and testing the device. The design choices made are justified by the technical background discussed earlier.

4.1 Hardware Design and Experimental Setup

The hardware architecture of the colorimetric pH sensor was designed for portability, consistency, and ease of use. Figure 1b and Figure 2 illustrate the device's architecture. The entire system is built into a two-layer 3D-printed rectangular box with a snap-fit lid and bottom. A 16×2 character LCD screen is mounted on the top of the lid, along with a single push-button which serves as the user input: the entire measurement process is triggered by one button press for simplicity[4]. The upper compartment of the enclosure houses the Raspberry Pi 4 along with a solderless breadboard on top for wiring the LCD, button and LEDs. The LCD is

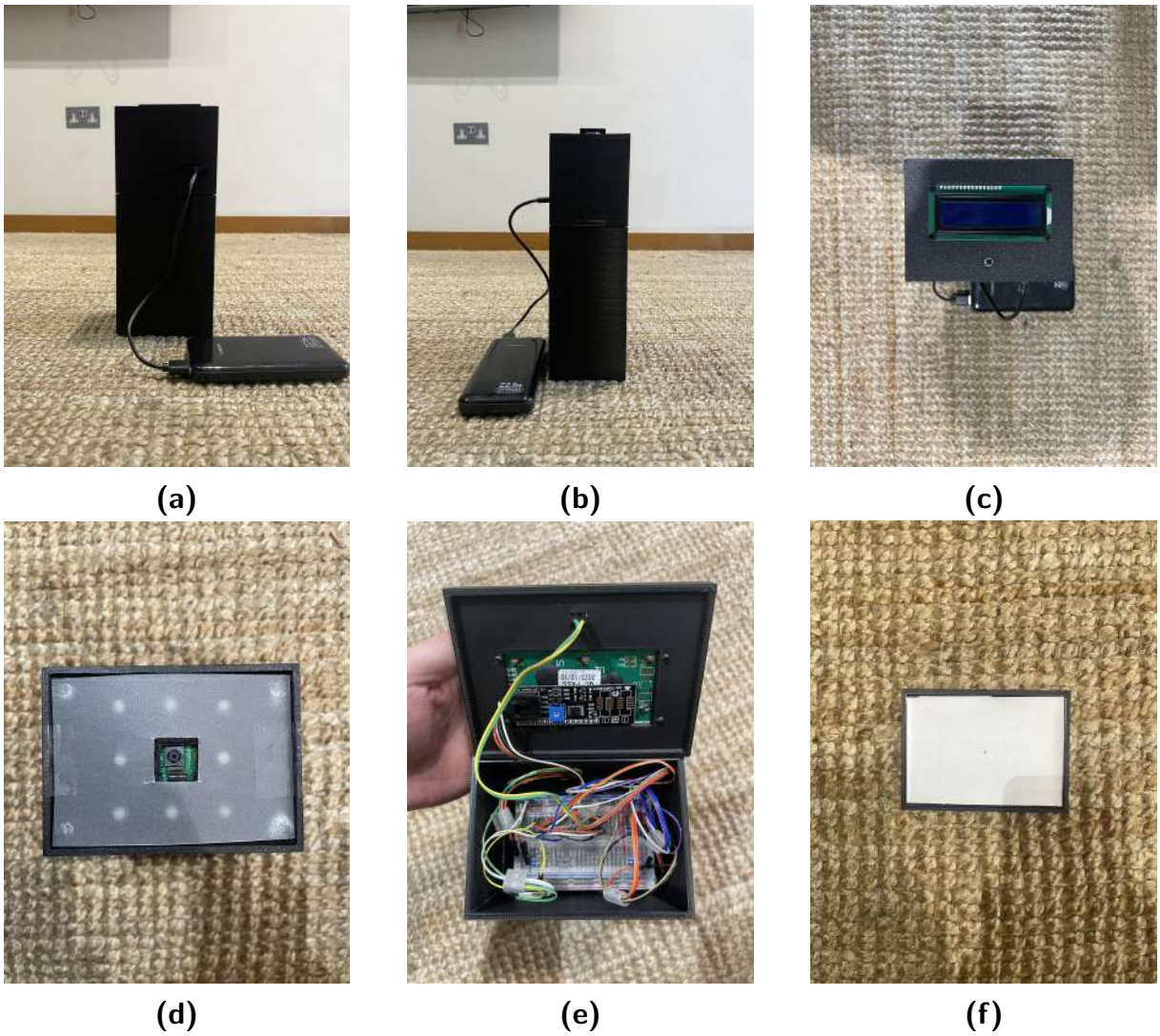


Figure 2: (a) Front view. (b) Side view. (c) Top view. (d) Bottom of the first compartment: eight white LEDs around the camera covered with diffusing sheet. (e) Breadboard and Raspberry Pi are housed inside of the first compartment. (f) Bottom of the enclosure: place to put the test strip

connected to the Pi via an I²C interface (using a PCF8574 I/O expander module), and it provides real-time status messages and the final pH reading to the user. The push-button is wired to a GPIO input pin (using the Broadcom BCM numbering, pin 22 in our setup) and configured with a pull-down resistor so that a stable LOW (0) is read when not pressed and HIGH (1) when pressed. The Raspberry Pi monitors this pin in a loop to detect button presses with software debouncing to avoid false triggers from mechanical bounce.

Beneath the Raspberry Pi board, a Raspberry Pi Camera module (the PiCamera v2 was used, with an 8 MP sensor) is mounted facing downward at the base of the compartment. Around the camera, eight white LEDs are arranged in a rectangular pattern to evenly illuminate the area where the pH strip will be placed. These LEDs are all wired in parallel to a GPIO-controlled power output (BCM

pin 18), allowing the software to turn them on or off. A thin diffusing sheet is installed below the LEDs to scatter the light and avoid direct LED hotspots on the strip[2, 12].

The lower compartment of the enclosure is simply a tunnel, the inside walls of the chamber are matte black to minimize reflections. Roughly 15cm below the camera is the snap-fit bottom where the user place the pH test strip. The strip rests on a white platform, ensuring that the background in the images is a consistent white reference. The distance from the camera to the strip was empirically chosen (around 15cm) to allow the entire strip to appear in the frame with some margin and to ensure depth of field is not an issue (everything at that distance is in focus given the camera's fixed focus setting).

During operation, the lid of the device is closed, creating a dark environment isolated from external light. The only illumination is from the device's own LEDs, which are controlled for intensity and duration. Power is supplied by a portable USB power bank connected to the Raspberry Pi, making the device mobile. In this configuration, pressing the start button initializes a measurement: the Raspberry Pi sets the LEDs to turn on, and then the Pi camera captures an image. After capturing, the LEDs are turned off to conserve power and also to avoid heating or prolonged light exposure (which could potentially bleach the strip if left on). The captured image is then processed as described in the next subsection, and finally the computed pH value is sent to the LCD for display.

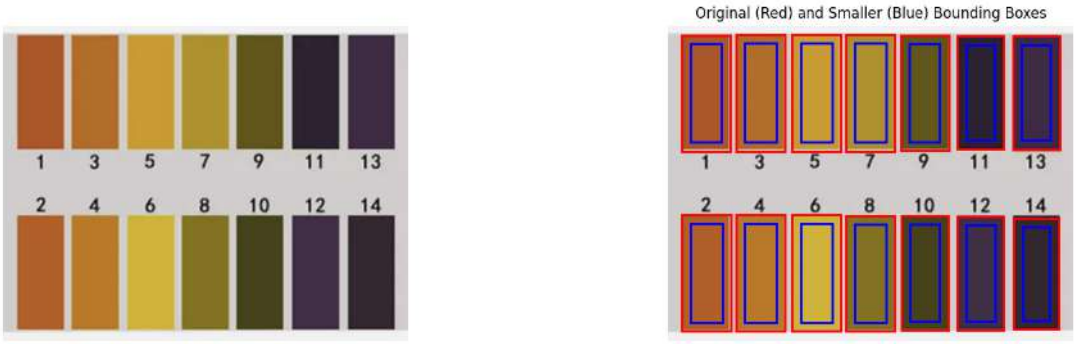
4.2 Calibration Procedure

To ensure accurate pH estimation from color, a comprehensive calibration of the sensor's color response was performed. Standard pH reference color cards (covering pH 1 through 14) were used as known benchmarks for the calibration. The reference card was inserted into the sensor under controlled illumination to simulate actual measurement conditions. An image of the reference color was captured using the sensor's camera, and the region of interest containing the color patch was extracted for analysis using the similar algorithm discussed in subsection 4.3. Figure 3 shows how the color patch of each pH value is extracted. The image region was converted from the native RGB color space to the CIE Lab color space using OpenCV, as the Lab space provides a device-independent and perceptually uniform representation of color. From this converted image, the mean L^*, a^*, and b^* values of the reference patch were computed to characterize the color of that pH sample[8, 24].

To improve reliability, the above measurement was repeated five times for each pH reference and the results were averaged. This averaging reduced the effect of any transient noise or lighting fluctuations, yielding a stable mean color value for each pH[4, 13]. The collected calibration data were stored in a lookup table, with each entry mapping a pH value (1–14) to its corresponding average Lab color coordinates $L^*, a^*, b^*_{\mathbf{ref}}$. This lookup table effectively represents the sensor's color response curve across the pH range, and it serves as the basis for interpolating intermediate pH values.

4.3 Software Workflow and Image Processing Algorithm

The software running on the Raspberry Pi is written in Python, utilizing the OpenCV library for image processing. The overall workflow can be divided into



(a) **(b)**

Figure 3: (a) Cropped Original Reference Card image. (b) The red boxes outline the contours of each color patch, and the blue boxes mark the ROIs from which the colors are extracted.

sequential stages from initialization to result display, which are executed automatically after the user presses the start button. Figure 4 outlines the measurement process step-by-step, and each step is detailed below:

1. **Initialization**

   Upon powering up the device (or restarting the program), the system performs setup routines. The GPIO pins are configured (button pin as input with pull-down, LED pin as output). The I²C LCD is initialized to operate in 4-bit mode and cleared. The Raspberry Pi Camera is opened. Auto-exposure and auto-white-balance settings are locked to the calibrated levels to maintain consistency. The LCD presents a ready message (the first line shows "Ready" and the second line instructs "Press to Start"), indicating to the user that the system is idle and awaiting input.

2. **Image Capture Trigger**

   The software enters a loop, polling the button state. When a button press is detected (and confirmed through a short debounce delay), the system transitions to capture mode. At this moment, the state changes from "ready" to "capturing/processing." The LCD is updated to "Processing..." to inform the user that a measurement is in progress. Immediately, the GPIO outputs are used to turn on the white LED array, flooding the chamber with light. After a brief pause, the camera captures an image of the test strip. For example, Figure 5 shows a captured image by the camera. This image is stored locally. As soon as the image is acquired, the LEDs are turned off to conserve power and avoid unnecessary heating[2, 4].

3. **Preprocessing and Background Isolation**

   The captured image contains the test strip on a white background inside the device. The first stage of processing is to isolate the region containing the strip. The image is converted to grayscale, producing an intensity image. An adaptive threshold is applied to create a binary mask that separates the white background from everything else[10]. In our implementation, a binary threshold was applied such that pixels brighter than a certain value (threshold 170

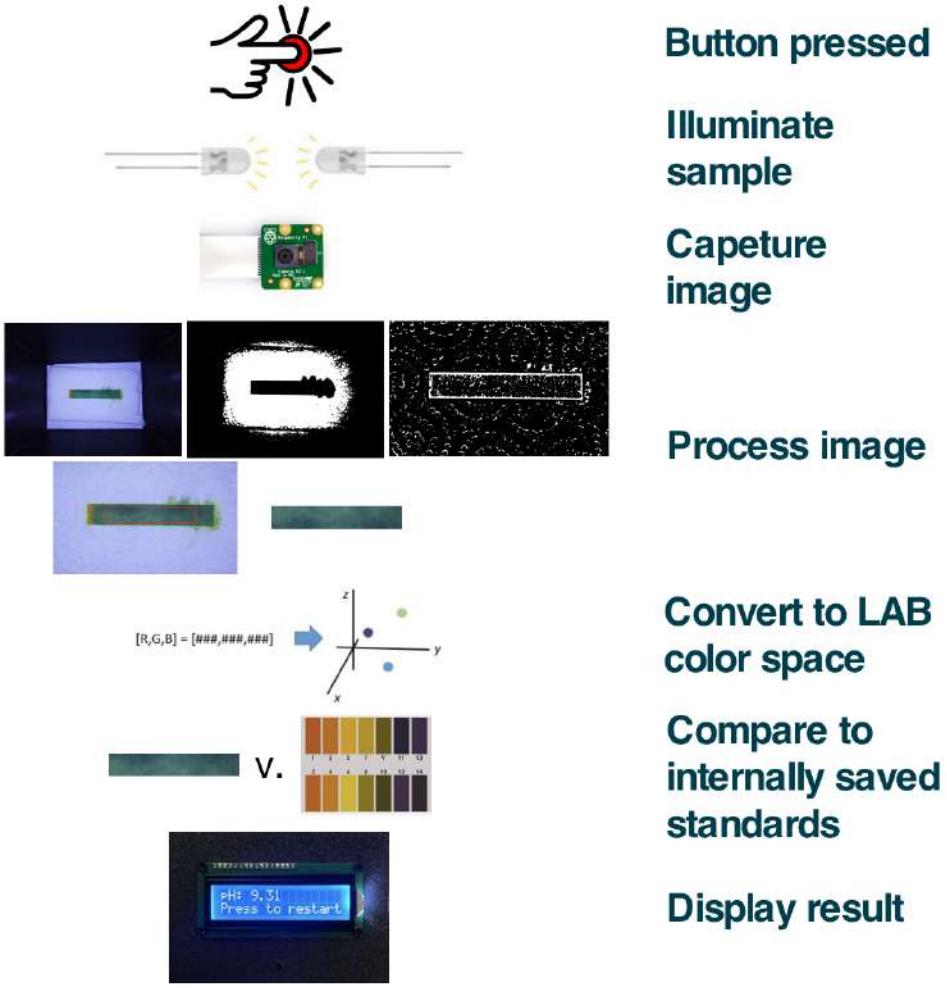


Figure 4: Pipeline of the software algorithm for pH measurement. Starting from a button press triggers image capture and processing, then leading to a pH result displayed on the LCD.

out of 255) are considered background (white) and turned to 1 (white in the mask), and others are 0. Given the controlled lighting, the background platform under the strip is uniformly white and typically much brighter than the strip, so this threshold cleanly segments the background.

After thresholding, morphological closing is applied with a small kernel (3×3 elliptical) to fill any small dark gaps within the white region (for instance, if the strip has text or markings that created dark spots). Then a morphological opening may be applied to remove any small false white patches not connected to the main background. The result is a mask where, ideally, the entire white background area is marked. The largest connected contour in this mask is assumed to be the background region. We then find the bounding box of this largest contour. To ensure we don't accidentally include the very edges of the black wall, we shrink the contour by 10% (scaling it about its centroid) before taking the bounding rectangle. This provides a margin so that the strip lying on the background will be just inside the cropped area. The original image is then cropped to this bounding box, effectively removing any black borders

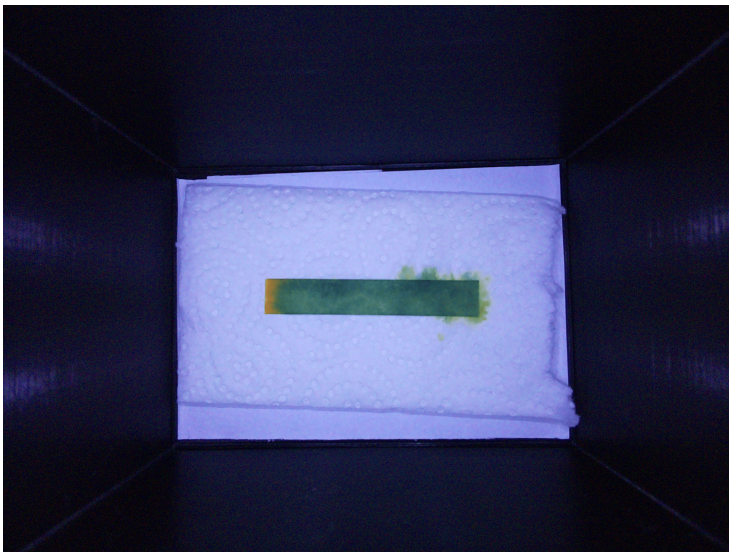
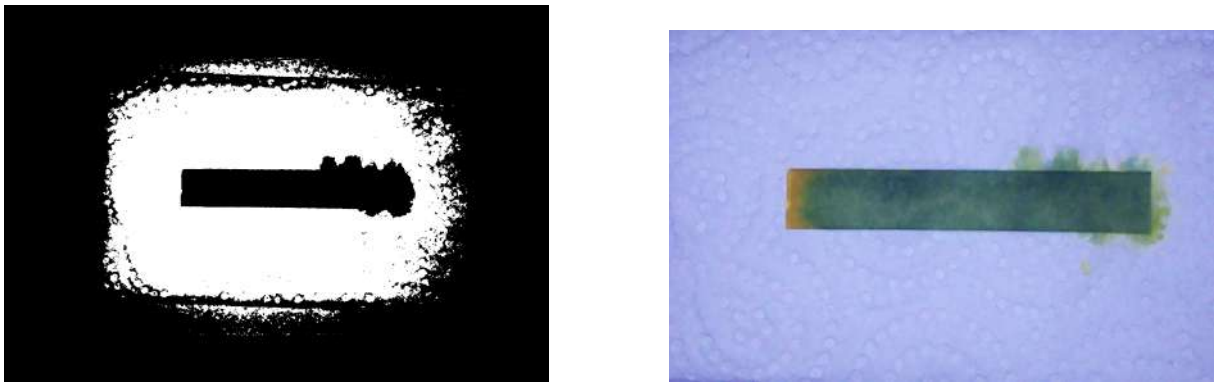


Figure 5: Original image captured

or irrelevant areas beyond the white platform. This yields a cropped image that tightly contains the white background and the strip. Figure 6 shows the original image Figure 5 after applying an adaptive threshold and a cropped image.



(a) **(b)**

Figure 6: (a) White mask of the original image. (b) Cropped image that only contains the strip and white background.

4. **Test Strip Detection**

   Now, within the cropped image, we need to find where the test strip itself is. We convert the cropped image to grayscale and apply an adaptive threshold (Gaussian) to detect the strip. We invert the binary sense here: since the strip is darker than the white background, we want a mask where the strip is white (1) and background is black (0). Adaptive thresholding is useful because even if the lighting has a slight gradient, it will still pick up the relatively darker strip. The parameters (block size and constant C) were tuned so that the entire strip region becomes one blob. Morphological operations are again used (closing with a small kernel) to ensure the strip's mask is solid. We then find

contours in this strip mask and expect the largest contour to correspond to the outer boundary of the strip. This contour outline the strip's edges precisely as shown in Figure 7a.

To focus on the indicator pad area on the strip (usually the middle part of the strip where the reagent is, excluding the very top or bottom which might be part of the white background or edges with shadows), the contour is scaled down. Specifically, we compute the contour's centroid and shrink it by 30%. This smaller contour should lie entirely within the colored pad area of the strip, as shown in Figure 7b. The bounding rectangle of this smaller contour is then taken, as shown in Figure 7c, and that defines the Region of Interest (ROI) for color analysis. We extract the ROI as a separate image region from the cropped image, as shown in Figure 7d. At this point, the ROI contains mostly the color of the pH reaction pad, filling most of the ROI with that color, with minimal background or edge.

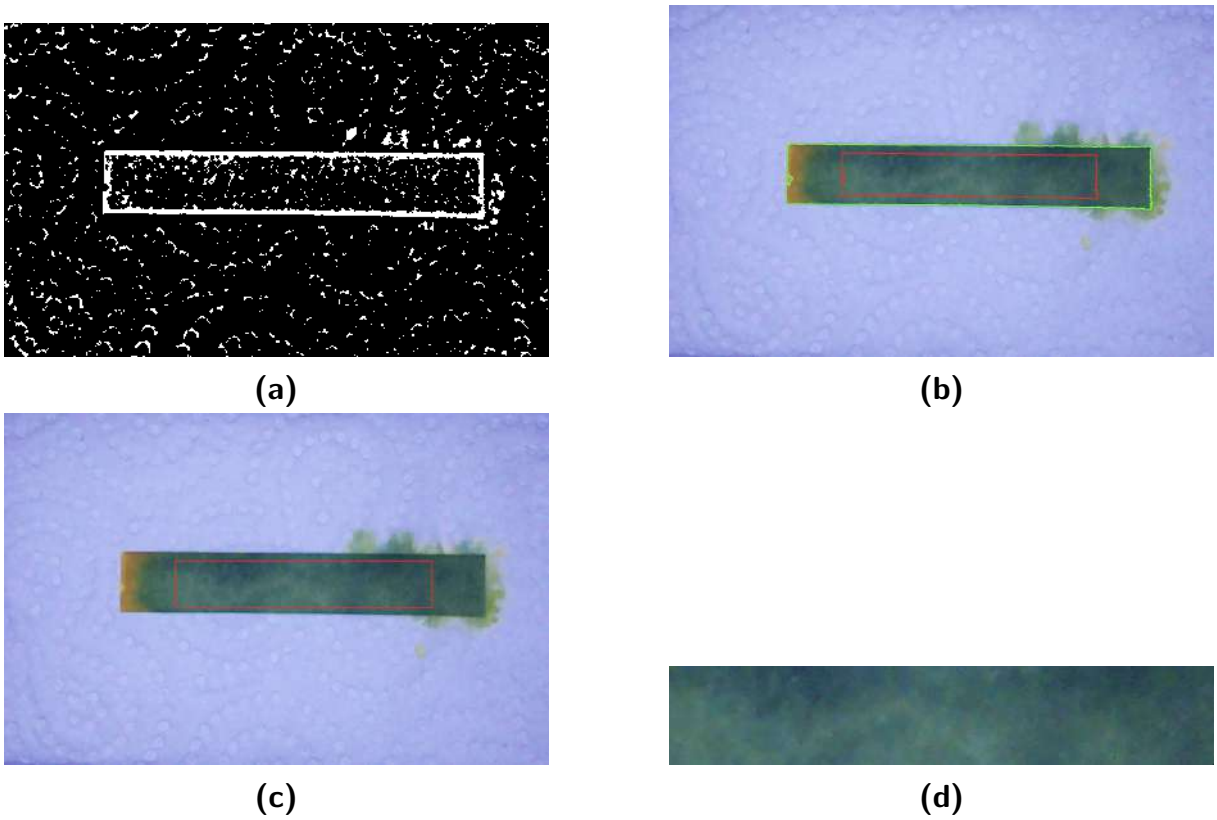


Figure 7: (a) Cropped image after applying adaptive thresholding and morphological operations. (b) The green contour outlines the strip's edges and the scaled-down red contour marks the ROI. (c) The bounding rectangle of the red contour. (d) Extracted ROI used for color analysis.

5. Color Analysis

The ROI, originally in RGB from the camera, is converted to the Lab color space for analysis. In OpenCV, the $L^*a^*b^*$ conversion produce L^* in $[0, 255]$ and a^*, b^* in $[0, 255]$ as well (with 128 as zero point). We scale these values to the

standard range: L^* is scaled to $[0, 100]$ (since Lab defines $L = 100$ as white), and 128 is subtracted from a^* and b^* to center them to the typical $[-128, 127]$ range. Now each pixel in the ROI has an (L, a, b) value. Instead of averaging all pixel values directly, we apply K-means clustering ($K = 3$) to the set of pixel Lab values. This involves treating each pixel's Lab tuple as a 3D point and grouping them into 3 clusters by iterative refinement (using the standard K-means algorithm with a random initial seeding repeated multiple times for stability)[10]. After clustering, we determine which cluster contains the most pixels (the largest cluster). We calculate the mean Lab value of that cluster (which in practice is given by the K-means output as one of the cluster centers).

This dominant cluster Lab value is taken to represent the strip's color. For example, if the strip turned an olive-green color, the cluster might have center L= fifty-something, a negative a* (greenish), and a positive b* (yellowish), corresponding to that hue.

6. pH Calculation

   With the dominant Lab color of the strip, the software now computes the pH. It does so by comparing this measured color against the stored reference Lab values for pH 1–14 obtained from calibration. The Euclidean distance (as in Equation 1) is calculated between the measured color and each reference color. The two smallest distances are identified[2]. Suppose the closest match is pH7 (distance d_7) and the second closest is pH6 (distance d_6). These indicate the strip's color lies between pH 6 and 7. The algorithm then interpolates to get a fractional pH using the formula discussed earlier (Equation 9). In this example, if the color were exactly halfway, it might report pH6.5. More generally, the output pH is a floating-point number. This value is then rounded to a convenient precision (e.g., two decimal places) for display. It's worth noting that if the color is almost exactly one of the reference colors, the second closest distance will be much larger than the closest, and the formula will yield nearly an integer (e.g., 7.00). This ability to output non-integer pH values is an advantage over simply reporting the nearest integer, as it provides more nuanced information especially for values that fall between the standard increments.

7. Result Display and Reset

   The resulting pH value is then shown to the user. In this example, the final pH value calculated is 9.31, as shown in Figure 8. The LCD, which was showing "Processing...", is cleared and updated to display the numeric pH result. It shows the result on the first line and a prompt like "Press to restart" on the second line, indicating the system is ready to run another test. Internally, the program enters a state waiting for the button to be pressed again. When pressed, it will clear the result and go back to the "Ready – Press to Start" message, rearming the cycle for a new measurement. This allows consecutive measurements to be taken with minimal interaction (the user just swaps out the strip and presses the button again).

Throughout the image processing pipeline, there are several debugging visualizations (not shown to the user in normal operation, but used during development) to

Figure 8: The resulting pH value is shown in the LCD screen.

verify each step. For instance, the intermediate binary masks and the ROI can be displayed on an attached monitor or saved, and contours can be drawn on images to ensure the correct regions are being identified. This helped in tuning parameters like the adaptive threshold window size, morphological kernel size, and contour scaling factor. The methods described above were iteratively refined during the project. For example, if the strip detection (step 4) failed for some images, we adjusted the threshold parameters or the physical placement of the strip to improve reliability. If the color clustering (step 5) occasionally picked up a wrong cluster (e.g., a reflection spot), we adjusted K or ensured the ROI truly excluded obvious glare. These adjustments were guided by repeated tests in the lab setting. Once the method proved consistent on a variety of trial images, we proceeded to formally test the device's performance, as described next.

4.4 Testing and Evaluation Procedure

1. **Controlled Lighting Environment Validation**

   A controlled lighting experiment was conducted to assess the influence of ambient light on the sensor's readings and to verify the effectiveness of the device's lighting enclosure[2, 14]. The colorimetric pH sensor was designed with its own light source and a black-box enclosure to shield the test strip from external light. Even slight changes in illumination can alter color readings in optical sensors, so controlling lighting is expected to improve consistency. To evaluate this, paired measurements were performed under two conditions: (1) with normal room ambient lighting turned on (simulating an uncontrolled environment), and (2) with ambient lighting off (the sensor operating in a darkroom with no external light, relying only on its internal illumination), while the sensor and test strip were kept in the same position and orientation. The test was done on the same sample type to allow direct comparison; for instance, a pH4.00 buffer was measured under both lighting conditions using identical procedures. Care was taken that the sensor's own LED illuminator was used in both cases (the only difference being the presence or absence of additional ambient light). After obtaining readings in both scenarios, the results were compared to see if any significant difference in the measured pH

occurred. If the controlled environment is effective, the readings with ambient light on vs. off should be very similar, indicating that external light has minimal impact. This experiment's rationale is to confirm that the device's design successfully isolates the sensing process from environmental lighting variations. A successful outcome (no significant difference between conditions) would validate that the sensor can produce reliable results in various lighting environments, whereas any large discrepancy would highlight the need for stricter lighting control or calibration adjustment.

2. **Time-Dependent pH Measurement Stability Test**

   A time-dependent stability experiment was performed to identify the optimal time window for reading the pH from the test strip. In this test, a pH test strip was dipped into a solution and the sensor's pH reading was recorded at regular intervals from 30 seconds up to 5 minutes after dipping. For example, in a representative trial using a neutral pH 6.92 buffer solution, the device's pH reading was recorded at $t = 30$ s, 60 s, 90 s, 120 s, 180s, and 300s. This procedure was repeated for multiple trials to ensure that observed trends were consistent and not an outlier[19]. The rationale behind this test is that colorimetric reagents often require a certain reaction time to fully develop a stable color, and readings taken too early or too late could be inaccurate. By monitoring the sensor output over 5 minutes, we can determine when the pH reading stabilizes and whether it remains stable or begins to drift after a certain point. Identifying this stability window is crucial for reliable sensor operation, as it informs us when to capture the reading to ensure accuracy and reproducibility.

3. **Accuracy and Precision Evaluation**

   An accuracy and precision assessment was carried out using standard pH buffer solutions of known pH values. Three buffer solutions were selected to span the acidic, neutral, and alkaline range of interest: pH4.00, pH6.92, and pH9.33. These specific values correspond to commercially available calibration buffers. For each buffer solution, five independent readings were taken using the colorimetric sensor. In each trial, a fresh test strip was immersed in the buffer and then placed in the sensor's holder for measurement. The measured pH value derived from the color of the strip was recorded. Between each trial, the system was reset and the strip was replaced to ensure independent measurements. From these repeated trials, we obtained a set of five measured pH values for each known buffer pH.

   To quantify accuracy, the sensor readings were compared to the known true pH of the buffers. We calculated the error of each reading as error $= pH_{\mathbf{measured}} - pH_{\mathbf{actual}}$, and from these we derived the mean absolute error (MAE) as a summary statistic. The MAE is given by:

$$\mathrm{MAE} = \frac{1}{N}\sum_{i=1}^{N} |pH_{\mathbf{meas},i} - pH_{\mathbf{actual}}| \tag{17}$$

   where N is the number of trials (here $N = 5$ for each buffer solution) and $pH_{\mathbf{actual}}$ is the buffer's certified pH value. The MAE (in pH units) provides an easy-to-interpret measure of accuracy (lower MAE means higher accuracy).

We also looked at the bias by computing the mean of $(pH_{\text{meas},i} - pH_{\text{actual}})$ for each buffer to see if readings tended to be consistently high or low.

To evaluate precision (repeatability), the standard deviation (σ) of the five readings at each pH level was calculated [22, 20]. A low standard deviation indicates that the sensor gives very consistent readings under the same conditions, i.e., good precision.

Additionally, we examined the overall linearity of the sensor's response across the pH range by plotting the sensor's measured pH vs. the actual pH of the buffers and performing a linear regression. The line-of-best-fit and the coefficient of determination (R^2) were obtained to assess how well a linear model describes the relationship[4, 13]. A high R^2 (close to 1) would indicate that the sensor's readings have a strong linear correlation with the true pH values, which is desirable for calibration simplicity. In summary, this accuracy and precision evaluation validates whether the colorimetric pH sensor can produce measurements that are both correct (accurate) and consistent (precise) for known reference standards.

5 Results and Discussion

5.1 Calibration Results

The calibration procedure yielded a complete dataset of color values for pH 1 through 14, which is summarized in Table 2. Each entry in the table represents the average Lab coordinates for the given pH, obtained from five repeated measurements. The consistency across those repeats was high – the standard deviation for each color component at a given pH was typically on the order of 1 unit or less in the Lab scale – indicating that the sensor's readings are stable and the calibration points are reliable. This consistency can be attributed to the controlled lighting and the averaging of multiple readings, which together minimize random noise. The distinct Lab values for each pH illustrate how the indicator's color progresses through the pH scale: for strongly acidic pH (around 1–3) the a component is relatively high (indicating a reddish hue) and b is increasing (from blueish to yellowish content), producing an overall orange-red color; near neutral pH 7, a becomes negative (indicating a shift towards green), while b remains positive (yellow-green appearance); and at alkaline pH (above 10) the b component turns negative (introducing blue tones) and a rises again towards positive (adding magenta/red content to the blue, resulting in purple hues). The L (lightness) values also change with pH, generally peaking in the mid-pH range (where the color was a bright yellow-green) and decreasing for very low or very high pH (where the colors were darker reds or purples). These trends are evident from the table and the plotted calibration curve.

Calibration curves in Figure 9 showing the variation of L, a, and b with pH (plotted from the data in Table 2). Each point represents the mean color reading at that pH. The non-linear and non-monotonic nature of these curves highlights the complexity of the color–pH relationship: for example, L (brightness) increases from pH1 to around pH6, then decreases at higher pH; a (red-green axis) starts high (red) at low pH, becomes negative (green) near neutral pH, and then rises back to positive (magenta) at high pH; b (yellow-blue axis) is positive (yellow) in mid-acidic range, drops towards zero around pH10, and becomes highly negative (blue) in the basic

Table 2: Average CIE Lab values for pH reference colors (pH 1–14). Each value is the mean of 5 measurements. L^* is in the range [0,100], and a^*, b^* are in [-128,127] (positive a^* = red, negative a^* = green; positive b^* = yellow, negative b^* = blue).

pH	L^*	a^*	b^*
1	43.03	70.05	-44.12
2	50.75	58.25	-16.03
3	46.28	46.55	39.27
4	67.86	10.82	65.47
5	67.75	7.71	63.65
6	73.13	-5.08	67.51
7	65.06	-8.89	53.07
8	57.77	-13.02	19.54
9	50.36	-14.33	36.83
10	25.26	-4.69	-4.32
11	18.32	13.64	-26.52
12	27.81	44.48	-57.78
13	27.54	36.0	-37.66
14	27.55	35.07	-44.43

range. These distinct trajectories demonstrate that each pH value corresponds to a unique position in the 3D Lab color space.

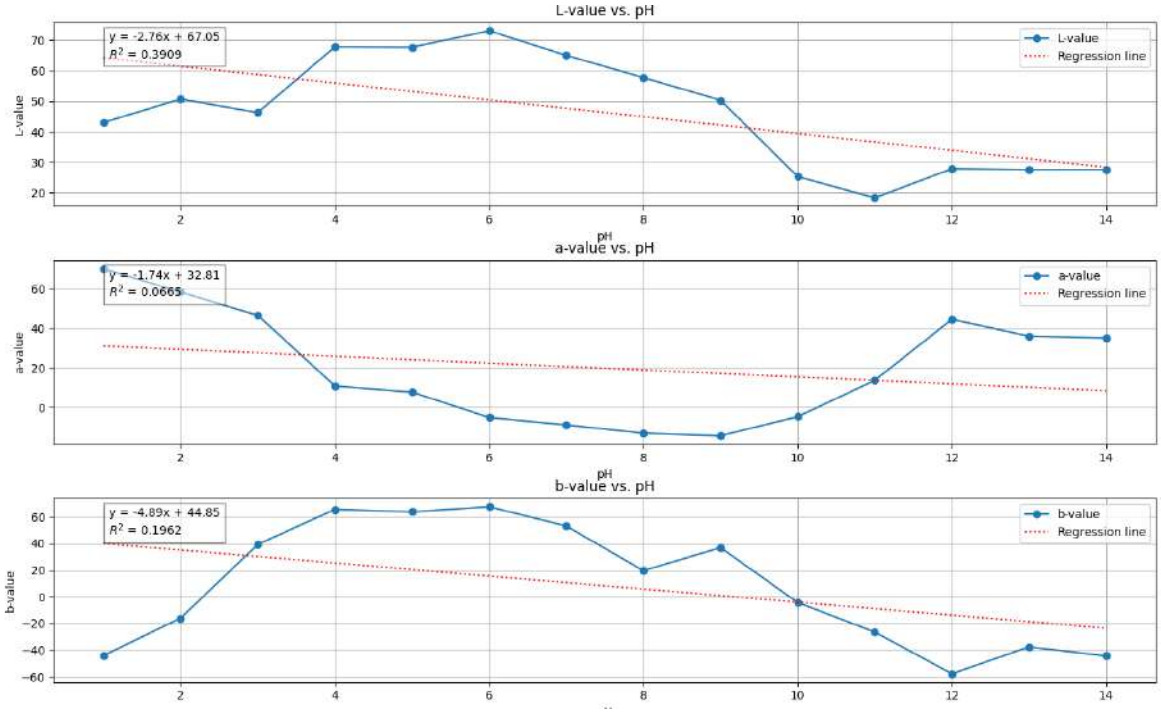


Figure 9: Calibration curves of L^*, a^*, and b^* across pH 1–14. The distinct, non-linear trajectories of the three Lab components indicate that no single component changes monotonically with pH.

The calibration results confirm that the sensor and indicator can distinguish pH across the full range: each integer pH unit produced a noticeably different color (as quantified in Lab). However, the relationship between any single color component and pH is clearly non-linear. To quantify this, a simple linear regression was attempted for each Lab component as a function of pH. Table 3 is the statistical summary of a linear regression analysis, modeling each Lab component (L, a, b) as a predictor of pH (from 1 to 14). The coefficient of determination (R^2) values were found to be very low for a and b (for example, $R^2 \approx 0.066$ for a vs pH and $R^2 \approx 0.196$ for b vs pH), and only moderate for L ($R^2 \approx 0.391$ for L vs pH). The poor R^2 indicates that a straight-line model cannot capture the variation in these components over the pH range. In practical terms, if one tried to use only a single component to predict pH (for instance, using only b as an indicator of pH), the errors would be large. Indeed, using a linear fit of b vs pH to estimate pH yields a mean absolute error (MAE) on the order of 2.82 pH units and a root mean square error (RMSE) even higher, which is unacceptable for most sensing purposes. These findings justify the need for a multi-component approach.

Table 3: Regression Performance Metrics for Lab Components

Component	R^2	MAE	RMSE
L	0.390896	2.533693	3.146098
a	0.066499	3.382984	3.894790
b	0.196239	2.828128	3.614016

One could consider a more sophisticated multivariate regression, or training a non-linear model (e.g. polynomial or neural network) for pH prediction. However, given the relatively sparse calibration points (only 14 reference samples for the entire range) and the complex shape of the color-response curve, a high-order polynomial might overfit, and a learned model might not generalize well without a significantly larger training dataset. Instead, the distance-based interpolation method (described in the Technical Background section) was chosen for its simplicity and effectiveness.

The chosen method is also inherently adaptable. If in the future additional calibration points are introduced, they can be incorporated into the lookup table, and the interpolation will automatically adjust to use them. Similarly, if the sensor's environment changes (different lighting conditions or camera properties), a new calibration can be performed and the same interpolation algorithm will accommodate the new reference values without any change in the underlying model – the Euclidean distance measure will correctly handle the new positions of colors in Lab space. This flexibility is a significant benefit over a fixed-formula approach.

5.2 Effect of Ambient Lighting on Measurement

The controlled lighting tests resulted in minimal differences between measurements taken with ambient room light on and off, indicating that the sensor's internal lighting and enclosure are effective in maintaining measurement consistency. In our paired experiments using the pH4 buffer, the measured pH values under normal lighting conditions were virtually the same as those in darkness. For example, the sensor reading for the pH4 sample was 4.38 ($\pm$0.06) with the room lights on, compared to 4.39 ($\pm$0.06) with the lights off. This tiny difference of ~ 0.06 is

well within the experimental uncertainty of the sensor. In practical terms, no statistically significant deviation due to ambient light was detected. Figure 10 and Table 4 summarizes this comparison, showing that both the mean and variance of the readings remain nearly identical between the two lighting conditions.

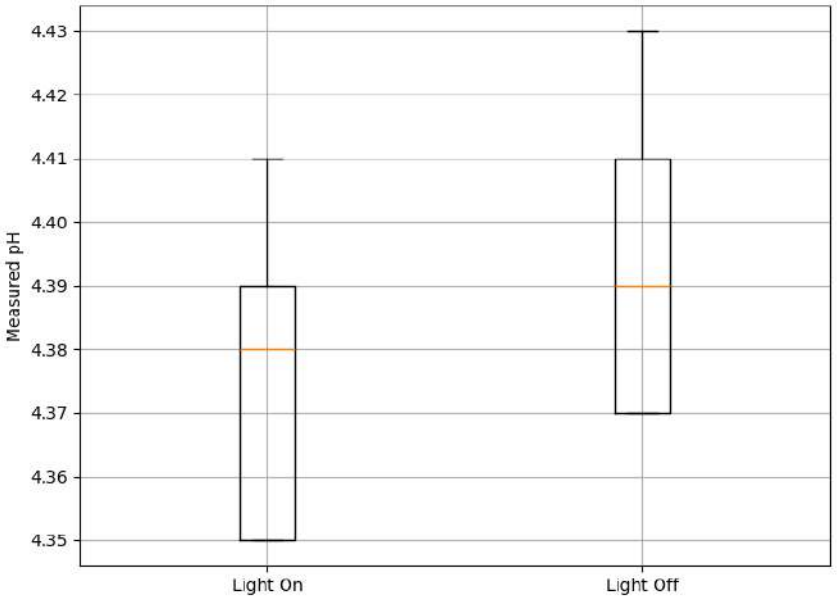


Figure 10: Boxplot of pH Measurements Under Different Lighting Conditions.

Table 4: Effect of ambient lighting on measured pH for a pH4 buffer sample. Five measurements were taken in each condition. The results show that the presence of room lighting has negligible impact on the sensor reading.

Lighting Condition	Mean Measured pH	Std. Dev.	Variance	Range
Light ON	**4.376**	**0.026**	**0.0068**	**0.06**
Light OFF	**4.394**	**0.026**	**0.0068**	**0.06**

These results suggest that the device's design successfully mitigates the influence of external lighting. The black box enclosure and internal LED illumination provide a controlled environment so that even if the surrounding light level changes, the captured image of the test strip is not significantly affected. This is an important validation of the reliability of the sensor in real-world usage: It implies that the sensor can be used in various ambient lighting environments without requiring complete darkness or recalibration for each setting. It also confirms that our initial decision to include an enclosure and consistent light source was well-founded.

It should be noted that while ambient light did not significantly skew the pH reading, this was under conditions where the sensor's internal illumination dominated the scene. In extreme cases (for instance, very bright direct sunlight or very low device lighting), results might differ. Nonetheless, for typical operating conditions,

the system is robust. Thus, the implication for design is that the current lighting control approach is sufficient, and users do not have to operate the device in a dark room for accurate results—an advantage in terms of ease of use.

5.3 Effect of Measurement Timing on pH Readings

The time-dependent measurement results revealed that the sensor's indicated pH value is rather stable in the first 3 minutes and becomes unreliable if read too late outside that window. Figure 11 illustrates a representative example of how the measured pH evolved with time after the test strip was introduced into the sample. Within the first 3 minutes window, the sensor output for each buffer remained nearly constant, varying only by a small amount. This indicates that the sensor reaches equilibrium in that time frame, yielding reliable readings. During this window, any further changes in the indicated pH were minimal. This suggests that by about 30–180 s after dipping the strip, the color reaction on the strip had fully developed and the sensor output stabilized[2, 23].

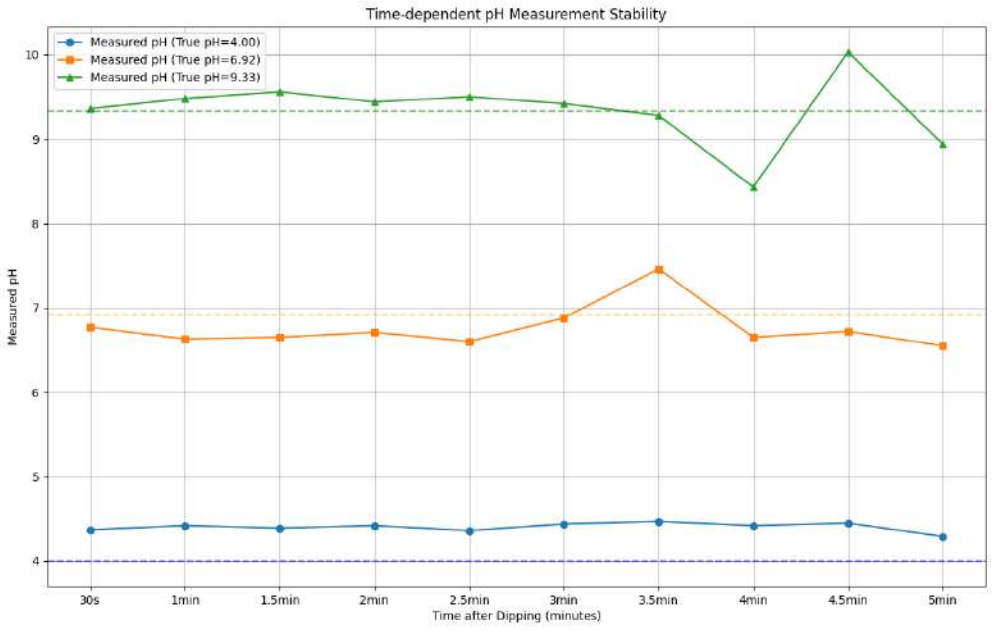


Figure 11: Time-dependent stability of the pH sensor reading. The plot shows the sensor's indicated pH value as a function of time after immersing the test strip in a sample.

However, beyond 3 minutes, the readings showed a noticeable drift compared to the stable plateau value in pH9.33 and pH6.92. In pH9.33 measurements, the pH reading at $t = 4\,\mathrm{min}$ had dropped to approximately 8.4–8.5, a noticeable deviation (roughly -1) from the value at 3 min. And in pH6.92 measurements, the pH reading rose to 7.46 around $t = 3.5\,\mathrm{min}$ compared to 6.88 at 3 min. This late-stage drift is likely due to the test strip beginning to dry out or the indicator chemistry undergoing changes after prolonged exposure (for instance, some pH indicator dyes may slowly change color as they desiccate or as equilibrium shifts). The key outcome is that there exists an optimal time window during which the pH reading is reliable

and stable. Outside of this window – too early, the reaction may be incomplete and give a low reading; too late, the reading can wander due to secondary effects. These findings underscore the importance of timing in the measurement process. Thus, the sensor's design and usage protocol will incorporate this knowledge: ensuring the measurement is taken within 3 minutes post-immersion maximizes reliability.

5.4 Accuracy, Precision, and Linearity of pH Measurements

Table 5 summarizes the measured pH values for the three standard buffer solutions (pH 4.00, 6.92, and 9.33) along with their statistical metrics. All measurements were taken within the 3-minute time window found in subsection 5.3. Each reported measured pH is the average of five independent trials, and the table includes the standard deviation as a measure of precision and the absolute error relative to the known pH.

Table 5: Accuracy and precision of the colorimetric pH sensor on standard buffer solutions. Each pH value was measured 5 times; the mean and standard deviation of the readings are given. The absolute error is the difference between the mean measured pH and the true pH of the buffer.

Buffer pH (Actual)	Mean Measured pH	Std. Dev.	Abs. Error
4.00	4.424	0.018	0.424
6.92	6.690	0.073	0.230
9.33	9.512	0.079	0.182
Overall	–	0.057	0.28

As seen in Table 5, the largest observed bias was an overestimate of 0.42 at pH4.00 (measuring 4.42 on average), while the pH9.33 buffer was read slightly high at 9.51. The overall mean absolute error (MAE) of the device was calculated to be approximately 0.28 pH units. This indicates that on average, the sensor's reading is within about ± 0.28 of the true pH. For many practical purposes in colorimetric strip testing, an error of less than half a pH unit is quite acceptable, given that human interpretation often has an uncertainty of around 0.5 to 1 pH unit especially in the mid-range colors (due to subjective color matching). This demonstrates a significant improvement in quantitative accuracy.

The precision of the sensor was also very good. The standard deviation of five measurements at each pH was on the order of 0.02–0.08 or less, resulting in an average standard deviation of 0.057. This level of consistency (roughly 1% relative uncertainty in the mid-pH range) means that the device produces repeatable results under identical conditions. There was no indication of any outlier or wild fluctuations in the readings; all five trials per buffer were tightly clustered around the mean. High precision is crucial for reliability because it means a single measurement is representative — the user can trust that repeated tests would give the same result, which is important for any practical sensor deployment.

To visualize the overall performance, Figure 12 plots the sensor's measured pH against the actual pH for the three buffer solutions. Ideally, all points would lie on

the unity line (the 45-degree line where measured = actual). In our data, the sensor's readings show a close correspondence to the true values, with small deviations observed at certain points. A linear regression fit (shown in the Figure 12) to the points yielded a slope very near 1, with a coefficient of determination $R^2 \approx 0.985$. This R^2 value confirms a strong linear correlation between the sensor readings and the true pH. In other words, over the tested range (4 to 9.33 pH), the sensor's response is linear and predictable, which simplifies calibration (a linear one-point or two-point calibration would suffice to map sensor output to actual pH). Nonetheless, an R^2 of 0.985 and low errors indicate that no significant non-linear effects are present; the sensor does not, for example, compress or expand the pH scale in any unexpected way[4, 21].

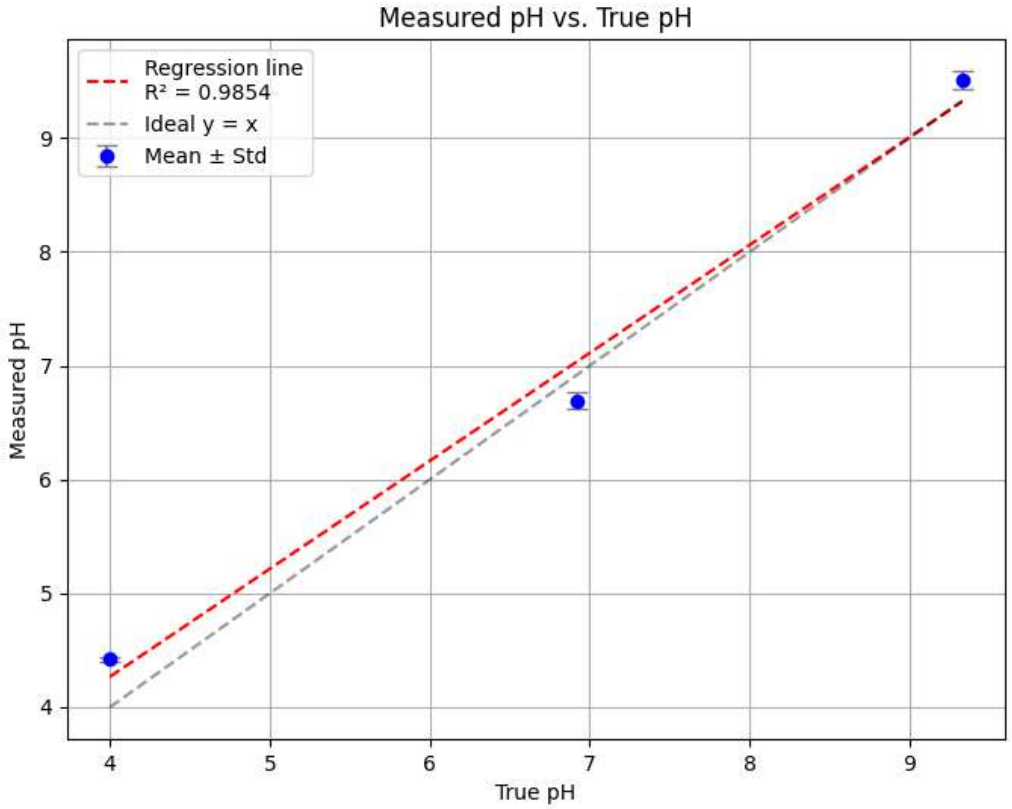


Figure 12: Correlation between sensor-measured pH and actual pH of standard buffers. Each data point represents the mean measured pH (from 5 trials) for a given buffer (pH 4.00, 6.92, 9.33), with error bars showing ± 1 standard deviation. The dashed line indicates the ideal $y = x$ relationship (measured = actual). The red dashed line is a linear fit to the data points, which has a slope near 1 and a high R^2 (approximately 0.98), demonstrating excellent linearity and accuracy of the sensor across the tested range.

Looking at specific points: in the acidic range, the device tended to slightly overestimate pH. For example, a buffer solution of true pH4.00 was consistently measured around 4.42 by the sensor. This is an error of +0.42, one of the higher errors observed in the tests. One possible reason is that the indicator's color change between

pH3 and pH4 is not very pronounced (both are shades of orange in the particular test strips used), and the device's color matching may have leaned toward the pH 5 reference color, causing an overshoot. This suggests a systematic bias at the low end of the pH spectrum for our system – the references or the interpolation might be slightly off in acidic hues.

In the neutral range around pH6.92, the sensor performed well. For a neutral pH sample, the device reported an average of pH6.69. The error here is only -0.23. Moreover, the precision at this point was good: the standard deviation of repeated measurements at pH6.92 was about 0.07 pH units. The repeatability at this pH was a bit lower than at pH 4 but still good. This low variance indicates that the device's readings are highly consistent when the same sample is measured multiple times. Such precision is a strong indicator that the imaging and analysis process is stable – the small differences in pixel-level noise or slight repositioning of the strip do not significantly affect the outcome.

Moving to the alkaline range, accuracy remained high. For instance, a test solution with true pH9.33 was measured by the sensor as 9.51 on average, yielding an error of about +0.18. This is a small error, indicating that the color difference for that range (a transition from green to blue on the strip) is captured well by the Lab distance method. A standard deviation of 0.08 was observed for the pH9.33 readings. A slightly higher variance here might be due to minor unevenness in color development at very high pH or slight differences in how the strip dries for each trial.

5.5 Analysis and Discussion of Results

The results confirm that the device meets its primary objective of providing an automated, objective measurement of pH that is generally in good agreement with the true values. The achieved accuracy ($MAE \approx 0.28$, $\sigma \approx 0.057$ and $R^2 \approx 0.985$) is quite promising for a first prototype. Considering that typical paper strips only have discrete color blocks for integer pH values, being able to interpolate and get fractional pH readings is an added benefit. For many environmental or educational applications, knowing the pH within ± 0.28 is sufficient to make decisions (e.g., differentiating between slightly acidic vs neutral water). However, certain applications (like precise titrations or regulatory water quality tests) might demand higher accuracy, so there is room for improvement.

One notable observation is the systematic bias in the lower pH range (the sensor overshooting the pH). This could be due to several factors:

- **The color reference data for low pH might not be perfectly representative. Perhaps the particular strip's color for pH4 in our calibration was a bit off (lighter) compared to typical, leading the algorithm to think a pH4 sample is closer to pH5's color. If so, recalibrating with a better representative color or using multiple strips to average the reference could help.**
- **The indicator chemistry at low pH could be such that small differences in concentration yield relatively smaller color differences, compressing the color scale. The interpolation might then overcorrect if the actual color progression is non-linear. In other words, our linear interpolation might slightly overshoot if, say, the pH3–4 transition in color space is steeper than pH4–5.**

- **Lighting or camera response at the red-orange end (acidic colors) might be less sensitive. Perhaps the camera sensor or our Lab conversion had minor inaccuracies for very saturated warm colors.**

On the other hand, the improved accuracy in neutral to alkaline range indicates that the chosen approach is quite effective there. The color transitions from green to blue (around pH 7 to 10 on universal indicator strips) seem to be well-captured by Lab distances. It is possible that in this range, the color difference per pH is larger, making it easier for the algorithm to discern fine differences. Additionally, our reference table might align particularly well in that region, or the indicator dyes behave more linearly in terms of color change.

The precision (repeatability) results, highlighted by the low standard deviations, suggest that the device's internal consistency is quite high. This validates the hardware design choices: the controlled lighting and fixed geometry effectively eliminated random errors like ambient light fluctuations or strip positioning variance. It also indicates that the image processing algorithm is stable – the ROI selection and clustering yield the same dominant color for a given strip consistently. This is important for user trust: an instrument that gives noticeably different readings on successive tries of the same sample would be unreliable, but ours does not exhibit that issue in the tests.

Another point of discussion is the response time and user experience. From button press to result display, the process currently takes on the order of a few seconds (mostly due to image capture and processing latency). This is already much faster than manual comparison in practice, but there is a slight lag due to the computational steps. On a Raspberry Pi 4, the entire pipeline (capture + processing) took roughly 6-7 seconds, which includes a deliberately added delay for LED illumination stability. This is a reasonable trade-off between speed and reliability.

In terms of power and portability, using a power bank, the device ran for several hours without issue. The LED illumination is momentary, and most of the time the system is idle waiting for input, so power consumption is low. The Raspberry Pi 4 is possibly overkill for this task – a Pi Zero or microcontroller with camera could potentially suffice – but the Pi 4 allowed easier development. This suggests future versions could be made even more compact and energy-efficient.

Comparing the device's concept to the literature examples: our approach did not employ advanced machine learning (like the deep learning model in Zhang et al.[6]), yet it achieved a respectable accuracy for a single-parameter sensor. The advantage of our simpler approach is transparency and ease of calibration – one can directly adjust the reference colors if needed. However, a learning-based model might better capture non-linear color-to-pH mappings and potentially reduce error further if trained on sufficient data. Luka et al.[4] reported a portable sensor for pH and nitrite; while their paper focused on dual-analyte detection, the pH accuracy was in a similar ballpark, suggesting that our results are on par with other non-ML, calibration-based systems. Yang et al.[2], in developing a Pi-based urine strip analyzer, likely encountered similar challenges regarding consistent lighting and color interpretation; our success in achieving low variability underscores the importance of those hardware considerations that we took into account from the start.

One limitation of the current system is that it is tailored to a specific type of pH test strip (universal indicator strips in this case). Different brands or types of strips might have different color responses. If one were to use a different indicator, the reference calibration would need to be redone. The device, however, is general-purpose enough that by updating the reference array, it could read any color-based strip (even for other analytes like chlorine or glucose tests) as long as they fit physically and appropriate color processing is implemented.

In summary, the project's outcomes validate the initial hypothesis that a portable, automated colorimetric pH sensor is feasible and can alleviate the subjectivity of manual readings. The discussion above highlights both the strengths (good accuracy, high consistency, ease of use) and the areas for improvement. These insights directly inform the conclusions and the potential future work on the system.

6 Conclusions and Future Work

In this project, we successfully developed a functional colorimetric pH sensor that integrates hardware and software to deliver quick and objective pH measurements. The device uses a Raspberry Pi with a camera and controlled LED lighting to capture the color of pH test strips, and an image processing algorithm in the CIE L*a*b* color space to determine the pH. The key findings from the experimental evaluation are: (1) The pH measurement stabilizes within about 3 minutes after the test strip is dipped, indicating the optimal time frame for reading the result. Within this window the sensor output is steady, whereas outside this window (too early or too late) the readings can be erroneous or drift over time. (2) The controlled lighting environment of the sensor (enclosed chamber with its own light source) is highly effective, as evidenced by minimal differences in readings with ambient light on or off. This suggests that the device is robust against normal variations in environmental lighting, an important factor for practical use. (3) Through calibration and testing, the system achieved a mean absolute error of around 0.28 pH units against known standards, and demonstrated high precision with minimal reading-to-reading variation. In addition, a strong linear correlation ($R^2 \approx 0.98$) between the sensor readings and actual pH. These results confirm that the approach of combining controlled illumination, clustering-based color extraction, and nearest-neighbor color matching is effective for this application. The project thus meets its primary goals of enhancing the consistency and accuracy of pH strip analysis compared to naked-eye interpretation.

Several key findings can be highlighted in conclusion: the use of a perceptually uniform color space (Lab) was validated by the sensor's performance, the interpolation technique provided sensible fractional pH outputs, and the hardware design choices (especially the lighting and enclosure) were crucial in minimizing noise and ensuring repeatable results. Moreover, the device remained portable and user-friendly, requiring just a button press to operate and displaying results clearly on an LCD. This proof-of-concept lays the groundwork for deployable field instruments for water quality testing and other colorimetric analyses, demonstrating that even with relatively simple algorithms, one can achieve reliable readings as long as the system is well-calibrated and the environment is controlled.

Despite the successes, there are clear avenues for future work and improvement:

- **Refining Calibration:**

  The observed bias at low pH suggests that the calibration could be refined. Gathering more calibration data points, possibly using high-precision prepared solutions around those problematic ranges (e.g., pH 3–5), and updating the reference color table could improve accuracy. Additionally, one could implement a dynamic calibration routine where the device, over time, learns adjustments if certain pH readings consistently error in one direction (a form of self-calibration or applying a correction curve).

- **Improving the Algorithm:**

  To push the accuracy further, especially if we want to reduce that 0.42 average error, more sophisticated algorithms could be explored. One idea is to use a form of non-linear regression or machine learning. For instance, a multilayer perceptron or random forest regressor could be trained on Lab values to predict pH, potentially capturing complex relationships beyond linear interpolation. Convolutional neural networks (CNNs) could even be used to analyze the raw image of the strip without explicit segmentation, possibly leveraging subtle patterns. However, any such approach would require a larger dataset of images for training. The benefit would be an algorithm that might correct for things like slight lighting variations or indicator non-linearities automatically. Nonetheless, the increased complexity must be justified by a significant gain in accuracy.

- **Hardware Enhancements:**

  Future prototypes might consider using a higher resolution or more sensitive camera to see if that improves color discrimination (though in our tests, resolution was not a limiting factor since the color area is large and uniform). Another hardware improvement could be an auto-sampler or multi-strip holder allowing multiple readings at once or sequentially without user intervention (useful for testing many samples in the field). Additionally, integrating wireless connectivity (through the Pi's built-in Wi-Fi or Bluetooth) could enable the device to log data to a phone or cloud server, which would be beneficial for long-term monitoring projects.

In conclusion, the project achieved its goal of demonstrating a low-cost, automated pH measurement system that addresses the subjectivity and inconsistency of manual test strip reading. The device serves as a tangible example of how principles of electronic engineering and computer vision can be applied to practical chemical sensing challenges. With further refinement, such a sensor could be a valuable tool for environmental monitoring, education, or any application where quick and reliable pH readings are needed outside of a traditional laboratory. The work done here can be extended and built upon, and it has provided a solid foundation in both the technical and practical aspects of automated colorimetric analysis.

7 References

References

[1] **B. M. Saalidong, S. A. Aram, S. Otu, and P. O. Lartey, "Examining the dynamics of the relationship between water pH and other water quality parameters in ground and surface water systems," *PLOS ONE*, vol. 17, no. 1, p. e0262117, 2022.**

[2] Z. Yang, G. Cai, J. Zhao, and S. Feng, "An optical POCT device for colorimetric detection of urine test strips based on Raspberry Pi imaging," *Photonics*, vol. 9, no. 10, Art. 784, 2022.

[3] P. B. Luppa, C. Müller, A. Schlichtiger, and H. Schlebusch, "Point-of-care testing (POCT): current techniques and future perspectives," *Trends Anal. Chem.*, vol. 30, pp. 887–898, 2011.

[4] G. S. Luka, E. Nowak, J. Kawchuk, M. Hoorfar, and H. Najjaran, "Portable device for the detection of colorimetric assays," *R. Soc. Open Sci.*, vol. 4, p. 171025, 2017.

[5] W. Di and D.-W. Sun, "Colour measurements by computer vision for food quality control—a review," *Trends Food Sci. Technol.*, vol. 29, pp. 5–20, 2013.

[6] S. Zhang, S. Wu, L. Chen, P. Guo, X. Jiang, H. Pan, and Y. Li, "Multi-task water quality colorimetric detection method based on deep learning," *Sensors*, vol. 24, no. 22, Art. 7345, 2024.

[7] L. DiCostanzo and B. Panunzi, "Visual pH Sensors: From a Chemical Perspective to New Bioengineered Materials," *Molecules*, vol. 26, no. 10, p. 2952, 2021.

[8] A. Pastore, D. Badocco, S. Bogialli, L. Cappellin, and P. Pastore, "pH Colorimetric Sensor Arrays: Role of the Color Space Adopted for the Calculation of the Prediction Error," *Sensors*, vol. 20, no. 21, p. 6036, 2020.

[9] P. Cebrián, L. Pérez-Sienes, I. Sanz-Vicente, Á. López-Molinero, S. de Marcos, and J. Galbán, "Solving Color Reproducibility between Digital Devices: A Robust Approach of Smartphones Color Management for Chemical (Bio)Sensors," *Biosensors*, vol. 12, no. 5, p. 341, 2022.

[10] S. Saifullah, R. Drezewski, A. Khaliduzzaman, L. K. Tolentino, and R. Ilyos, "K-means segmentation based on Lab color space for embryo detection in incubated egg," arXiv, arXiv:2103.02288 [cs.CV], Aug. 2022.

[11] A. Soetedjo and E. Hendriarianti, "Plant Leaf Detection and Counting in a Greenhouse during Day and Nighttime Using a Raspberry Pi NoIR Camera," *Sensors*, vol. 21, no. 19, p. 6659, 2021.

[12] A. Tonelli *et al*., "Sensing Optimum in the Raw: Leveraging the Raw-Data Imaging Capabilities of Raspberry Pi for Diagnostics Applications," *Sensors*, vol. 21, no. 10, p. 3552, 2021. (MDPI Open Access)

[13] E. Kirchner, P. Koeckhoven, and K. Sivakumar, "Improving Color Accuracy of Colorimetric Sensors," *Sensors*, vol. 18, no. 5, p. 1542, 2018. (MDPI Open Access)

[14] P. Escobedo *et al*., "Smartphone-Based Diagnosis of Parasitic Infections with Colorimetric Assays in Centrifuge Tubes," *IEEE Access*, vol. 7, pp. 160636–160645, 2019.

[15] C. Wang, J. Zheng, and Z. Li, *"Machine-free calibration of colorimetric sensors using a distance-based approach in the CIE Lab color space,"* ChemistryOpen, vol. 8, no. 1, pp. 78–86, 2019.

[16] **Luo, M. R., Cui, G., and Rigg, B., *"The development of the CIE 2000 color-difference formula"*, Color Research Application, vol. 26, no. 5, pp. 340–350, 2001.**

[17] **Steinegger, A., Wolfbeis, O. S., and Borisov, S. M., *"Optical Sensing and Imaging of pH Values: Spectroscopies, Materials, and Applications"*, Chemical Reviews, vol. 120, pp. 12357–12489, 2020.**

[18] **Gonzalez, R. C., & Woods, R. E., *Digital Image Processing*, 3rd ed., Pearson, 2008.**

[19] **Jeong, H., Jung, B. J., Kim, J. H., Choi, S.-H., Lee, Y. J., and Kim, K. S., *"Instant pH sensor based on the functionalized cellulose for detecting strong acid leaks"*, R. Soc. Open Sci., vol. 9, p. 211660, 2022.**

[20] **O. S. Wolfbeis, *"An overview of progress in chemical sensing using optical methods,"* Analytica Chimica Acta, vol. 864, pp. 1–13, 2015.**

[21] **M. Abdulrazik, M. S. Naik, R. Fernandes, J. P. C. Ward, and J. R. Miller, *"A low-cost smartphone-based colorimetric reader for test strips,"* Sensors, vol. 20, no. 16, p. 4482, 2020.**

[22] **D. Wencel, T. Abel, and C. McDonagh, *"Optical chemical pH sensors,"* Analytical Chemistry, vol. 86, no. 1, pp. 15–29, 2014.**

[23] **Q. Lin and M. Wu, *"Design of a real-time pH monitoring system based on paper strip and image analysis,"* Sensors and Actuators B: Chemical, vol. 225, pp. 637–643, 2016.**

[24] **L. Gzara, H. Gallardo, J. R. Steter, and M. A. S. Siqueira, *"Colorimetric detection using near-isogenic lines combined with an imaging approach in Lab color space,"* Biosensors, vol. 8, no. 4, p. 108, 2018.**

Appendices

A Code Appendix

1.1 Main Program (main.py)

Below is the full code listing for the Raspberry Pi-based colorimetric pH sensor, including hardware initialization, calibration, image capture, and pH calculation.

```
#!/usr/bin/env python3
from picamera2 import Picamera2
import time
import RPi.GPIO as GPIO
import smbus

# Import your pH calculation function from lab.py
import detection  # Changed from ‘import color‘ to ‘import lab‘

# --------------------------
# I2C LCD Configuration
# --------------------------
I2C_ADDR = 0x20        # <--- CHANGE to match your LCD’s I2C address
BUS = smbus.SMBus(1)   # I2C bus 1 on Raspberry Pi

LCD_WIDTH = 16         # Characters wide
LCD_LINES = 2          # 2 lines total

% For a 2×16 display, addresses for line 1 & 2 typically:
LCD_LINE_ADDR = [0x80, 0xC0]

LCD_CMD = 0
LCD_CHR = 1

ENABLE = 0b00000100
LCD_BACKLIGHT = 0x08  % 0x08 = backlight ON, 0x00 = OFF

# --------------------------
# GPIO Configuration
# --------------------------
BUTTON_PIN = 22        % BCM pin for button
LED_PIN = 18           % BCM pin for LED

# --------------------------
# LCD Helper Functions
%--------------------------
def lcd_byte(bits, mode):
    """Send a byte to the LCD via PCF8574 I²C."""
    high_bits = mode | (bits & 0xF0) | LCD_BACKLIGHT
    low_bits  = mode | ((bits << 4) & 0xF0) | LCD_BACKLIGHT

    # Write high 4 bits
```

```
    BUS.write_byte(I2C_ADDR, high_bits)
    lcd_toggle_enable(high_bits)

    # Write low 4 bits
    BUS.write_byte(I2C_ADDR, low_bits)
    lcd_toggle_enable(low_bits)

def lcd_toggle_enable(bits):
    """Toggle the enable pin."""
    time.sleep(0.0005)
    BUS.write_byte(I2C_ADDR, (bits | ENABLE))
    time.sleep(0.0005)
    BUS.write_byte(I2C_ADDR, (bits & ~ENABLE))
    time.sleep(0.0005)

def lcd_init():
    """Initialize 2x16 LCD in 4-bit mode via I²C expander."""
    lcd_byte(0x33, LCD_CMD)  % Initialization sequence
    lcd_byte(0x32, LCD_CMD)
    lcd_byte(0x06, LCD_CMD)  % Cursor move direction
    lcd_byte(0x0C, LCD_CMD)  % Display on, cursor off
    lcd_byte(0x28, LCD_CMD)  % 2-line display, 5x8 matrix
    lcd_clear()

def lcd_clear():
    """Clear display."""
    lcd_byte(0x01, LCD_CMD)
    time.sleep(0.002)

def lcd_string(message, line=0):
    """
    Send string to LCD line (0 or 1 for a 2-line display).
    Pads or truncates to 16 characters.
    """
    if line >= LCD_LINES:
        line = LCD_LINES - 1  % Clamp
    lcd_byte(LCD_LINE_ADDR[line], LCD_CMD)
    message = message.ljust(LCD_WIDTH, " ")[:LCD_WIDTH]
    for char in message:
        lcd_byte(ord(char), LCD_CHR)

# --------------------------
# Main Program
# --------------------------
def main():
    % Initialize LCD
    lcd_init()

    % Setup GPIO
```

```
GPIO.setmode(GPIO.BCM)
GPIO.setup(BUTTON_PIN, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)
GPIO.setup(LED_PIN, GPIO.OUT, initial=GPIO.LOW)

% Setup Pi Camera
picam2 = Picamera2()
still_config = picam2.create_still_configuration()  % Fixed typo
picam2.configure(still_config)
picam2.start()
time.sleep(1)

% Show "Ready to Start"
lcd_clear()
lcd_string("Ready", 0)
lcd_string("Press to Start", 1)
print("System is ready. Press button to capture and process pH.")

state = 0  % 0 = ready, 1 = showing pH

try:
    while True:
        if GPIO.input(BUTTON_PIN) == GPIO.HIGH:
            time.sleep(0.2)  % Debounce
            if GPIO.input(BUTTON_PIN) == GPIO.HIGH:
                if state == 0:
                    GPIO.output(LED_PIN, GPIO.HIGH)
                    lcd_clear()
                    lcd_string("Processing...", 0)
                    print("Processing image...")
                    time.sleep(5)  % Short delay for LED illumination

                    % Capture image as "strip.png"
                    picam2.capture_file("strip.png")
                    print("Image captured -> strip.png")

                    GPIO.output(LED_PIN, GPIO.LOW)
                    % Call lab.py for pH calculation
                    pH_value = detection.process_image()  % Must return a float

                    % Display the result
                    lcd_clear()
                    pH_str = f"pH: {pH_value:.2f}"
                    lcd_string(pH_str, 0)
                    lcd_string("Press to restart", 1)
                    print(f"Result: {pH_str}")
                    state = 1

                elif state == 1:
                    lcd_clear()
```

```
                        lcd_string("Ready", 0)
                        lcd_string("Press to Start", 1)
                        print("Clearing pH. System ready again.")
                        state = 0

                    while GPIO.input(BUTTON_PIN) == GPIO.HIGH:
                        time.sleep(0.05)
            time.sleep(0.1)
    except KeyboardInterrupt:
        pass
    finally:
        lcd_clear()
        picam2.stop()
        GPIO.cleanup()
        print("Exiting.")

if __name__ == "__main__":
    main()
```

1.2 pH Calculation and Image Processing (detection.py)

Below is the full code listing for the image processing and pH calculation module with debug monitoring.

```
import cv2
import numpy as np
import math

# ========== pH Reference Colors ==========
phColorLab = np.array([
    [43.03, 70.05, -44.13],   # pH 1
    [50.75, 58.25, -16.03],   # pH 2
    [46.28, 46.55, 39.27],    # pH 3
    [55.66, 30.33, 46.36],    # pH 4
    [67.75, 7.71, 63.65],     # pH 5
    [73.13, -5.08, 67.51],    # pH 6
    [78.31, -13.80, 57.94],   # pH 7
    [57.77, -13.02, 19.54],   # pH 8
    [50.36, -14.33, 36.83],   # pH 9
    [25.26, -4.69, -4.32],    # pH 10
    [18.32, 13.64, -26.52],   # pH 11
    [27.81, 44.48, -57.78],   # pH 12
    [27.54, 36.00, -37.66],   # pH 13
    [27.55, 35.07, -44.43]    # pH 14
], dtype=np.float32)

def getLabPhValue(color, phColorLab):
    """
    Finds a fractional pH value given a measured Lab color, comparing to
    the nearest two reference pH Lab values via Euclidean distance.
    Returns (final_ph, closest_pH, second_closest_pH).
```

```
    """
    dists = []
    for i in range(len(phColorLab)):
        dist = math.sqrt(
            (color[0] - phColorLab[i][0])**2 +
            (color[1] - phColorLab[i][1])**2 +
            (color[2] - phColorLab[i][2])**2
        )
        dists.append(dist)
    min_index1 = dists.index(min(dists))
    dist1 = dists[min_index1]
    dists[min_index1] = float("inf")
    min_index2 = dists.index(min(dists))
    dist2 = dists[min_index2]
    if min_index1 <= min_index2:
        final_ph = min_index1 + abs(min_index1 - min_index2) * (dist1 / (dist1 + dist2))
    else:
        final_ph = min_index1 - abs(min_index1 - min_index2) * (dist1 / (dist1 + dist2))
    return final_ph + 1, (min_index1 + 1), (min_index2 + 1)

def scale_contour(contour, scale):
    """
    Scale a contour around its centroid by ‘scale‘ (e.g. 0.7 = 70\%).
    """
    M = cv2.moments(contour)
    if M["m00"] == 0:
        return contour
    cx = int(M["m10"] / M["m00"])
    cy = int(M["m01"] / M["m00"])
    scaled_contour = []
    for point in contour:
        x, y = point[0]
        x_new = int((x - cx) * scale + cx)
        y_new = int((y - cy) * scale + cy)
        scaled_contour.append([[x_new, y_new]])
    return np.array(scaled_contour, dtype=np.int32)

def process_image():
    """
    Processes the image 'strip.png' and returns the fractional pH value.
    If debug mode is enabled, displays intermediate debug windows.
    """
    img = cv2.imread('strip.png')
    if img is None:
        raise FileNotFoundError("Could not load image: strip.png")
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    thresh_val = 170
    _, white_mask = cv2.threshold(gray, thresh_val, 255, cv2.THRESH_BINARY)
    kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (3,3))
```

```
white_mask = cv2.morphologyEx(white_mask, cv2.MORPH_CLOSE, kernel, iterations=1)
white_mask = cv2.morphologyEx(white_mask, cv2.MORPH_OPEN, kernel, iterations=1)
contours, _ = cv2.findContours(white_mask, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
if not contours:
    raise ValueError("No white background found!")
largest_contour = max(contours, key=cv2.contourArea)
scale_fac = 0.9
smaller_contour = scale_contour(largest_contour, scale_fac)
x_bg, y_bg, w_bg, h_bg = cv2.boundingRect(smaller_contour)
cropped_img = img[y_bg:y_bg+h_bg, x_bg:x_bg+w_bg].copy()
gray_cropped = cv2.cvtColor(cropped_img, cv2.COLOR_BGR2GRAY)
binarized_strip = cv2.adaptiveThreshold(
    gray_cropped, 255,
    cv2.ADAPTIVE_THRESH_GAUSSIAN_C,
    cv2.THRESH_BINARY_INV,
    blockSize=51,
    C=5
)
binarized_strip = cv2.morphologyEx(binarized_strip, cv2.MORPH_CLOSE, kernel, iterations=
binarized_strip = cv2.morphologyEx(binarized_strip, cv2.MORPH_OPEN, kernel, iterations=1
contours_strip, _ = cv2.findContours(binarized_strip, cv2.RETR_EXTERNAL, cv2.CHAIN_APPRO
if not contours_strip:
    raise ValueError("No strip detected with adaptive threshold. Adjust blockSize/C.")
largest_contour_strip = max(contours_strip, key=cv2.contourArea)
scale_factor = 0.7
small_contour = scale_contour(largest_contour_strip, scale_factor)
x_strip, y_strip, w_strip, h_strip = cv2.boundingRect(small_contour)
roi = cropped_img[y_strip:y_strip+h_strip, x_strip:x_strip+w_strip].copy()
roi_lab = cv2.cvtColor(roi, cv2.COLOR_BGR2Lab).astype(np.float32)
roi_lab[..., 0] = roi_lab[..., 0] * (100/255)
roi_lab[..., 1] = roi_lab[..., 1] - 128
roi_lab[..., 2] = roi_lab[..., 2] - 128
pixels_lab = roi_lab.reshape(-1, 3).astype(np.float32)
K = 3
criteria = (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER, 10, 1.0)
_, labels, centers = cv2.kmeans(
    pixels_lab, K, None, criteria, 10, cv2.KMEANS_RANDOM_CENTERS
)
unique, counts = np.unique(labels, return_counts=True)
dominant_cluster_idx = unique[np.argmax(counts)]
dominant_lab = centers[dominant_cluster_idx]
print(dominant_lab)
final_ph = getLabPhValue(dominant_lab, phColorLab)
pH_float = round(final_ph[0], 2)
cv2.rectangle(cropped_img, (x_strip, y_strip),
              (x_strip + w_strip, y_strip + h_strip),
              (0, 0, 255), 2)
cv2.putText(cropped_img,
            f"pH={pH_float}",
```

```
            (x_strip, y_strip - 10),
            cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)
    output_image = cropped_img.copy()
    cv2.drawContours(output_image, [largest_contour_strip], -1, (0, 255, 0), 2)
    cv2.drawContours(output_image, [small_contour], -1, (0, 0, 255), 2)
    cv2.imshow("White Mask", white_mask)
    cv2.imshow("Cropped Image", cropped_img)
    cv2.imshow("Adaptive Binarized (Test Strip)", binarized_strip)
    cv2.imshow("Original and Smaller Contours", output_image)
    cv2.imshow("ROI", roi)
    cv2.waitKey(0)
    cv2.destroyAllWindows()
    return pH_float
```