WATER QUALITY TESTING DEVICE BY USING SENSORS AND IMAGE PROCESSING TECHNIQUES

Bachelor of Engineering (Electrical and Electronics) (Honours)

Swinburne University of Technology Sarawak Campus

LOH JIA CHIEW

2021

Declaration

I hereby declare that this report entitled "Water quality testing device by using sensors and image processing techniques" is the result of my own project work except for quotations and citations which have been duly acknowledged. I also declare that it has not been previously or concurrently submitted for any other degree at Swinburne University of Technology Sarawak Campus.

Name: Loh Jia Chiew

Date: 28/5/2021

Acknowledgement

I would like to thank my parents, Loh Yun Huat and Nan Lah @ Ah Feng, for being supportive and take good care of my health by cooking healthy food as well as bringing me to exercise. Besides, I would like to thank Mr. Anderson Kho and Dr. Then Yi Lung for the guidance throughout the project.

Abstract

Water quality testing is essential to ensure that the water is always safe to use. Most people are not aware of the water quality that is consumed and that is used. It is important to have a simple water quality testing device for people to detect the water quality easily. This is to ensure that the water is safe to consume and use. There are some powerful water quality testing device such as RANA Water Quality Kit and remote sensing water quality kit. These powerful water quality testing devices might be hard for average people to use and the results take some times. Hence, a simple water quality testing device is proposed, which is a water quality testing device using image processing techniques and sensors. The main components that are used to process the data are microcontroller and microcomputer. The microcontroller used is Arduino Uno, and the microcomputer used is Raspberry Pi Zero. The proposed water quality testing device has 2 mode, which are instant mode and continuous mode. Instant mode takes an image of test strip, and the microcomputer will run it through some image processing methods, such as colour conversion and perspective transform. Then, the test strip data is compared with the colour chart to obtain the water quality information. The water quality information is then saved to the device and a cloud database. For continuous mode, the water quality data is obtain from the sensors, and process in a microcontroller, then send to the microcomputer. The microcomputer is connected to a display, and the display are able to show the water quality information. The instant mode of the device is tested with 4 types of water, and the continuous mode is tested with 2 types of water. From the results, it is shown that both mode has successfully obtained the water quality information and the information is satisfactory. Other than that, it is shown that the instant mode data has successfully been saved into the device and cloud database, which is Google Sheet.

Table of Contents

1 INTRODUCTION 1.1

1.1 PROBLEM STATEMENT 1.1
1.2 RESEARCH QUESTIONS 1.1
1.3 OBJECTIVES 1.1
1.4 PROJECT SCOPE 1.1
1.4.1 Project outline 1.1
1.4.2 Project deliverable 1.1

2 LITERATURE REVIEW 2.1

2.1 EXISTING PORTABLE WATER QUALITY TESTING DEVICE 2.1
2.1.1 RANA Water Quality Kit 2.1
2.1.2 Low Cost Water Quality Testing Device 2.2
2.1.3 Remote Sensing Water Quality Kit 2.3
2.2 COMMUNICATION PROTOCOLS 2.4
2.2.1 SPI 2.4
2.2.2 UART 2.4
2.3 IMAGE PROCESSING TECHNIQUES 2.5
2.3.1 Perspective transform 2.5
2.3.2 Colour space conversion 2.6
2.3.3 Colour difference 2.10

3 METHODOLOGY 3.1

3.1 OVERVIEW ... 3.1
3.2 INSTANT DATA ACQUISITION PROCESS ... 3.3
3.3 CONTINUOUS DATA ACQUISITION PROCESS ... 3.5
3.4 EXPERIMENT ... 3.5
3.4.1 Instant data acquisition ... 3.5
3.4.2 Continuous data acquisition ... 3.7

4 RESULTS ... 4.1

4.1 INSTANT DATA ACQUISITION ... 4.1
4.2 CONTINUOUS DATA ACQUISITION ... 4.3

5 DISCUSSION ... 5.1

5.1 INSTANT DATA ACQUISITION ... 5.1
5.2 CONTINUOUS DATA ACQUISITION ... 5.1

6 CONCLUSION ... 6.1

7 REFERENCES ... 7.1

8 APPENDICES ... 1

8.1 PYTHON CODE FOR INSTANT MODE ... 1
8.2 ARDUINO CODE FOR CONTINUOUS MODE ... 6
8.3 PYTHON CODE OF OVERALL COMBINATION ... 8

List of Tables

Table 1 RGB to Grayscale equations ... 2.8
Table 2 Advantages and Disadvantages of RGB to Grayscale equations ... 2.9
Table 3 Colour difference equations developed by CIE ... 2.10
Table 4 Pin connections of water quality testing device ... 3.2
Table 5 Comparison of instant data acquisition results ... 5.1
Table 6 Comparison of continuous data acquisition results ... 5.2

List of Figures

Figure 1 Single master with several slaves ... 2.4
Figure 2 UART data packet ... 2.5
Figure 3 Example of perspective transform ... 2.6
Figure 4 Equation for perspective transform ... 2.6
Figure 5 Results from the low light image enhancement ... 2.7
Figure 6 Equation for RGB to CIELAB conversion ... 2.8
Figure 7 Block diagram of the water quality testing device ... 3.1
Figure 8 Prototype of the water quality testing device ... 3.3
Figure 9 Filtered water test strip ... 3.6
Figure 10 Lab solution test strip ... 3.6
Figure 11 Carbonated water test strip ... 3.7
Figure 12 Tap water test strip ... 3.7
Figure 13 Carbonated drink and filtered water ... 3.8
Figure 14 Filtered water output ... 4.1

Figure 15 Lab solution 4.1
Figure 16 Carbonated water output 4.2
Figure 17 Tap water output 4.2
Figure 18 Water quality information in the device 4.2
Figure 19 Water quality testing device on the cloud 4.3
Figure 20 Carbonated drink output 4.3
Figure 21 Filtered water output 4.4

List of Abbreviations/ Notations/Glossary of Terms

RANA – Rapid Adaptive Needs Assessment

DoD – Department of Defence

PEAK – Pre-positioned Expeditionary Assistance Kit

PC – Personal Computer

ADC – Analog-to-Digital Converter

IoT – Internet of Things

WiFi – Wireless Fidelity

SPI – Serial Peripheral Interface

SCLK – Serial Clock

MOSI – Master Out Slave In

MISO – Master In Slave Out

SS – Slave Select

UART – Universal Asynchronous Receiver/Transmitter

TX – Transmitter

RX – Receiver

RGB – Red, Green, Blue

CIE – International Commission on Illumination

Ppm – parts per million

1 Introduction

1.1 Problem Statement

In most rural areas, the water quality is bad due to poor regulation of agriculture waste. The use of water and consumption of water with bad water quality can cause some serious issues. For example, consumption of contaminated water can cause disease such as diarrhoea and dysentery (World Health Organisation 2019). Another example is that usage of hard water can cause dry skin on human (Roland 2019).

1.2 Research Questions

What are the existing portable water quality testing kits or devices? What are the image processing methods can be used to determine the water quality?

1.3 Objectives

The objective of this project is to identify if the water quality is safe for usage and consumption. Other than that, it is to design a portable water quality testing device that is easy-to-use and reliable. Besides, the data of the water quality testing device can be stored in a cloud database.

1.4 Project Scope

1.4.1 Project outline

This project includes the design process of a water quality testing device, such as identify the communication protocol used by the water quality testing device, identify the image processing techniques will be used by the water quality testing device, and identify the hardware requirements of the water quality testing device. Other than that, research on the communication protocol and image processing techniques are needed to be done.

1.4.2 Project deliverable

A research paper

A water quality testing device prototype

2 Literature Review

2.1 Existing portable water quality testing device

2.1.1 RANA Water Quality Kit

RANA water quality kit is developed with the purpose of measure the water quality during the first 3 days after a natural disaster happened (Barham et al. 2011). It is developed by the DoD in the United States in 2011.

The kit is able to measure the temperature, turbidity, pH, dissolved oxygen and conductivity of water by deploy the kit on the water. Those water quality parameters are measured by using corresponding probes, and the probes might need calibration. The kit will record the water quality data to the built-in data logger for every 2 hours. The kit then uses a handheld device, such as smartphone, to send the data to a database by using PEAK telecommunication network.

RANA kit has a similar design to a water quality kit design by Cheny and others in 2005 (Cheny et al. 2005). The 2005 water quality kit designed by Cheny and others also uses probes and a handheld device, which is pocket PC, to send the data. Even though both kit uses wireless technology to transmit data, the RANA kit is much better than the 2005 water quality testing kit because it uses telecommunication network, which can transmit data as long as the kit is within the network coverage instead of Bluetooth, which has short transmission distance.

The strength of RANA kit is that it requires minimal human effort. After the kit is deploy at water by a person, the person can leave the place. Other than that, it has a data logger to back up the data, in case the data is not able to transmit to the database through the smartphone. The weakness of the kit is it has the risk of flip over if the water is torrential. Besides, the kit might float away from the initial deployment place.

The kit can be improved by adding a gyroscope sensor or an accelerometer to detect if the kit is upright to the water to prevent flipping over. Other than that, it can be improved by adding several motors to control the position of the kit on the water.

2.1.2 Low Cost Water Quality Testing Device

This low cost water quality testing device is developed by Indu and Choondal in 2016 (Indu & Choondal 2016). The purpose of this device is to raise the awareness and interest about water quality in general society, as the device is low cost.

The device can detect the pH, conductivity and temperature of water through corresponding probes. The analog values then go through an ADC, and then to the microcontroller. The microcontroller is connected to a screen, so the data will show on the screen.

Unlike the RANA kit, this lost cost device has no data logger, so the data shown on the screen is not saved anywhere. Other than that, the low cost device has no communication system, which means it is a stand-alone device and the water quality data cannot be export

As its name, this strength of this device is it is low cost. Other than that, unlike RANA kit, which requires set up and calibration, this low cost device is easy-to-use, as long as the probes are in water, the data is shown on the screen.

The weakness of this device is it only detects basic parameters of water quality, unlike the RANA kit which detects specific parameters like dissolved oxygen level in water. Other than that, the data obtained by the device only shows on screen, and there is no way to export the data out of the device.

The device can be improved by adding external ways to export the data out of the system. The simplest way is send it through the serial connector of the microcontroller.

2.1.3 Remote Sensing Water Quality Kit

Remote sensing water quality kit is developed by Siam and others in 2019 (Siam et al. 2019). This remote sensing kit is designed to be place inside the water reservoir of household. It is anticipated as an important part of futuristic smart city concept, as this device if an IoT device.

The remote sensing kit has a similar design with the previous low cost water quality device, as it uses several water quality sensors, such as pH, turbidity, conductivity and temperature. The sensors are connected to a multiplexer, and then connect to a microcontroller. The microcontroller then connect with a WiFi module to send the water quality data to the cloud. User are able to view the data through website.

The reason that this remote sensing kit is anticipated as an important part in the smart city concept is it has the potential to grow into a reliable and efficient network, like the wireless network system proposed in 2017 by Nguyen and Phung (Nguyen & Phung 2017). In fact, a similar version of the system, which is an IoT based water quality monitoring system, is proposed by Jerom and Manimegalai in 2020 (Jerom & Manimegalai 2020)

The strength of this kit is the water quality data are available online. Other than that, it only requires 1 time instalment, which is put the kit inside the reservoir. The weakness of this kit is unlike RANA kit, it does not has a data logger, which means that if the data is not successfully upload to the cloud, the data is gone. Other than that, similar to the low cost device, it only detects basic parameters of water quality.

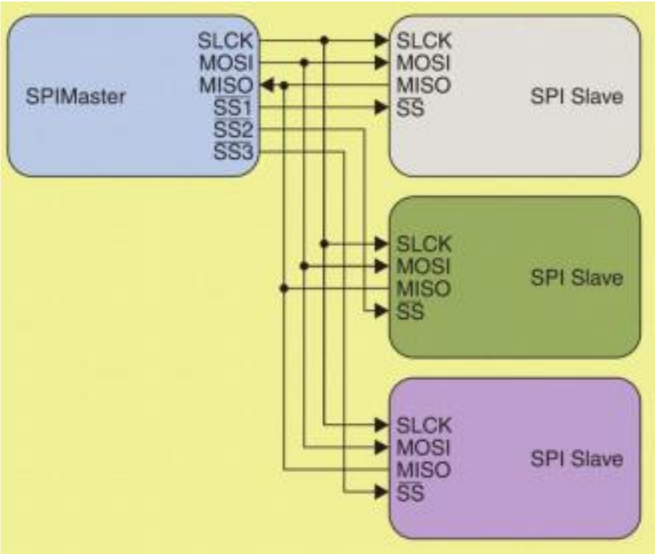
This kit can be improved by adding a backup storage to it to overcome the risk of data loss. Other than that, adding another back up way of communication system might be useful too, as the reservoir is a closed area, which might cause the kit loss its WiFi connection.

2.2 Communication protocols

2.2.1 SPI

For SPI, there are 4 important communication pins, which are the SCLK, MOSI, MISO and SS. The biggest advantage of SPI is a master device can communicate with several slave devices (Leens 2009).

Figure 1 Single master with several slaves



The SCLK and the MOSI signal is coming out from the master device to the slave devices. Other than that, the SS signal is also coming out from the master device to the slave devices, and it is used to choose the slave which the master wishes to communicate with. Meanwhile, the MISO signal is coming out from the slave devices to the master device. All the pins in SPI is unidirectional.

2.2.2 UART

For UART, there are 2 communication pins, which are the TX and RX (Harutyunyan 2020). The TX and RX are asynchronous. The TX and RX work at the same baud rate, where baud rate is the data transfer speed (Gupta 2020). The baud rate of UART can be set, and the unit is bits per second.

There are 3 types of mode for UART, which are simplex, half-duplex and full-duplex. In simplex mode, only one pin is working. For half-duplex, either the TX pin or RX pin is working at a time. For full-duplex, both the TX and RX pin are working at the same time.

Figure 2 UART data packet

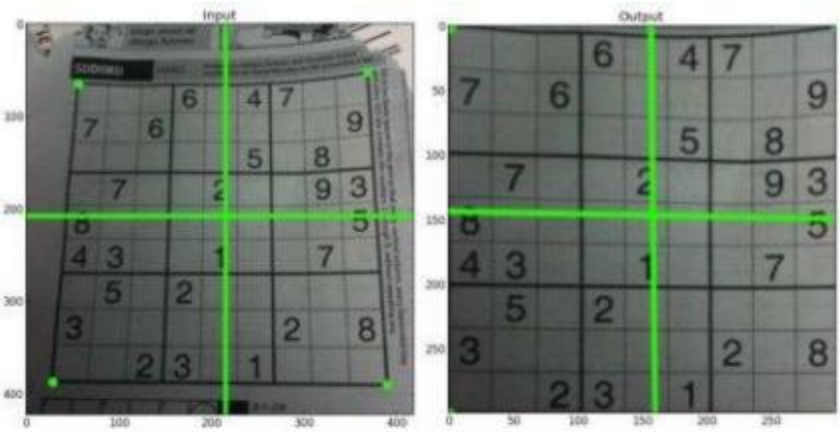
UART are able to transmit and receive 8-bit of data, but the data is transmit and receive bit by bit serially (Sharma 2015).

2.3 Image processing techniques

2.3.1 Perspective transform

Perspective transform is an image processing technique that can transform an image from a 3D perspective into a good 2D perspective image. In other words, it can change the viewpoint of an image (Mody et al. 2017). This technique is used in different fields, such as estimation of number of people in crowded scenes (Lin 2001) and road traffic sign detection (Wu 2016).

Figure 3 Example of perspective transform



In order to apply perspective transform, 4 points need to be identified. These 4 points indicates the corner of an image. Other than that, the width and height of result image need to be identified too. Besides, a 3×3 transformation matrix is needed to convert the original 3D image into 2D image.

Figure 4 Equation for perspective transform

$$\begin{bmatrix} t_i x'_i \\ t_i y'_i \\ t_i \end{bmatrix} = \texttt{map\_matrix} \cdot \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix}$$

$$dst(i) = (x'_i, y'_i), src(i) = (x_i, y_i), i = 0, 1, 2, 3$$

The transformation matrix can be calculated by using the original points and the destination points. This technique can straighten a slanted and distorted image.

2.3.2 Colour space conversion

Colour space conversion is extremely useful when dealing with image in computer. This is because by converting the colour space from one to another, the raw meaningless data can be transform into useful data. Other than that, by changing the colour space, the image can be process easier (Tseng & Lee 2018).

RGB to CIELAB

The CIELAB has 3 channels, which is the L* representing perceptual lightness, and a*b* representing the human perception of colours. According to Tseng and Lee (2018), by converting RGB colour space into CIELAB colour space, it is easier for them to perform low light image enhancement.

Figure 5 Results from the low light image enhancement

Besides that, it is found out that the CIELAB colour space is good for colour analysis, such as colour acceptability decision making (Connolly & Fleiss 1997). It is normally used in colour inspection of flat objects, such as cards and papers.

In order to perform change the image from RGB to CIELAB, the RGB is first convert to the CIEXYZ, then from the CIEXYZ to CIELAB.

Figure 6 Equation for RGB to CIELAB conversion

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \leftarrow \begin{bmatrix} 0.412453 & 0.357580 & 0.180423 \\ 0.212671 & 0.715160 & 0.072169 \\ 0.019334 & 0.119193 & 0.950227 \end{bmatrix} \cdot \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

$$X \leftarrow X/X_n, \text{where} X_n = 0.950456$$

$$Z \leftarrow Z/Z_n, \text{where} Z_n = 1.088754$$

$$L \leftarrow \begin{cases} 116 * Y^{1/3} - 16 & \text{for } Y > 0.008856 \\ 903.3 * Y & \text{for } Y \leq 0.008856 \end{cases}$$

$$a \leftarrow 500(f(X) - f(Y)) + delta$$

$$b \leftarrow 200(f(Y) - f(Z)) + delta$$

$$f(t) = \begin{cases} t^{1/3} & \text{for } t > 0.008856 \\ 7.787t + 16/116 & \text{for } t \leq 0.008856 \end{cases}$$

$$delta = \begin{cases} 128 & \text{for 8-bit images} \\ 0 & \text{for floating-point images} \end{cases}$$

RGB to Grayscale

RGB to Grayscale is another type of colour space conversion, where the 3 channel RGB colour are transform into 1 grayscale channel. By doing this conversion, the colour data of the image is abandoned, and in exchange, the size of the image has become smaller and the image process speed on the grayscale image is faster (Raveendran et.al 2018).

There are several ways to convert RGB to Grayscale, as each algorithms has its own advantage and disadvantage (Ahmad et.al 2018).

Table 1 RGB to Grayscale equations

Name	Equation

Intensity	$\mathcal{G}_{Intensity} \leftarrow \frac{1}{3}(R+G+B).$
Luminance	$\mathcal{G}_{Luminance} \leftarrow 0.3R+0.59G+0.11B.$
Lightness	$\mathcal{G}_{Lightness} \leftarrow \frac{1}{100}(116f(Y)-16),$ $Y=0.2126R+0.7152G+0.0722B.$ $f(t)=\begin{cases} t^{1/3} & \text{if } t>(6/29)^3 \\ \frac{1}{3}\left(\frac{29}{6}\right)^2 t+\frac{4}{29} & \text{otherwise.} \end{cases}$
Value	$G_{Value}=\max(R,G,B).$
Luster	$\mathcal{G}_{Luster} \leftarrow \frac{1}{2}(\max(R,G,B)+\min(R,G,B)).$

Table 2 Advantages and Disadvantages of RGB to Grayscale equations

Name	**Advantage**	**Disadvantage**
Intensity	Easy to calculate, require minimum computational power, and fast in operation.	It assume all the RGB channels have the same weight, which is not human friendly, because human perceive colours differently.
Luminance	Balanced weight among RGB, closer to human perception on brightness.	-
Lightness	Closer to human perception of colour.	-
Value	Provide absolute brightness information which can keep some	Slow in speed, as it needs to compare all the values within the channel, and

	feature of the original image.	also might affected by the brightness easily.
Luster	Less sensitive to brightness change.	Slow in speed, as it needs to compare all the values within the channel

2.3.3 Colour difference

Colour difference is a metric to identify the difference between 2 colours. CIE has created several standardized methods to identify the colour difference, and it is called delta-E (Sayed et.al 2017).

Table 3 Colour difference equations developed by CIE

Name	**Equation**
CIE76	$\Delta E^*_{ab} = \sqrt{\Delta L^{*2} + \Delta a^{*2} + \Delta b^{*2}}$
CIE94	$\Delta E^*_{94} = \sqrt{(\frac{\Delta L^*}{K_L S_L})^2 + (\frac{\Delta C^*}{K_C S_C})^2 + (\frac{\Delta H^*}{K_H S_H})^2}$
CIEDE2000	$\Delta E^*_{00} = \sqrt{(\frac{\Delta L'}{K_L S_L})^2 + (\frac{\Delta C'}{K_C S_C})^2 + (\frac{\Delta H'}{K_H S_H})^2 + R_T \left(\frac{\Delta C'}{K_C S_C}\right)\left(\frac{\Delta H^*}{K_H S_H}\right)}$

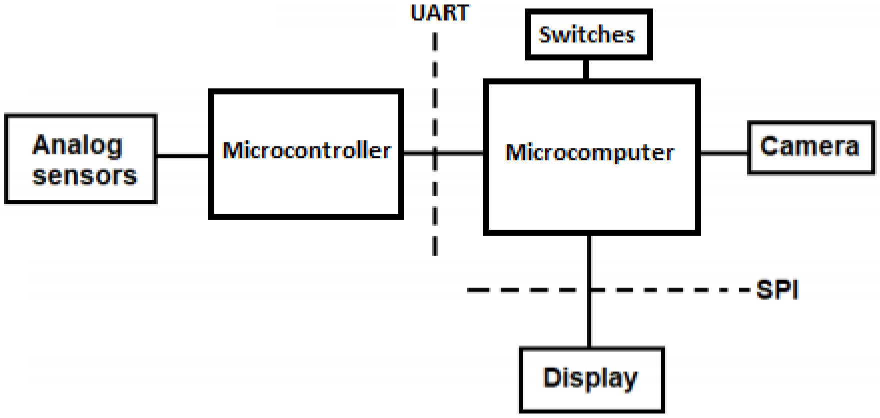
By looking at the equations, it is found that as the time passed, the equations are getting more complex, which means that it requires more computational power and the calculation process speed will be slower down. In exchange, a more precise and accurate colour difference can be known.

Do note that from 1994 onwards, the colour space that used to calculate the colour difference changed from L*a*b* to L*C*H*, which L*C*H* colour space represent lightness, chroma and hue.

3 Methodology

3.1 Overview

Figure 7 Block diagram of the water quality testing device



The water quality testing device has a display which will show a main menu, and there will be 3 buttons, which is up, down and enter to control the main menu. By using this main menu, user can easily navigate to the functions of the water quality testing device

There will be 2 modes for this water quality testing device, which is instant data acquisition and continuous data acquisition. The instant data acquisition is using image processing technique to get the data, while the continuous data acquisition is using sensors to get the data.

The instant data acquisition process is the camera will capture the image of the test strip and send it to the microcomputer. The image will then process by the microcomputer by using image processing techniques, and the useful information will be obtained. Then, the information will be saved in the microcomputer and a cloud database, as well as shown on the display.

The continuous data acquisition is the analog sensors will get the data of the water and send it to the microcontroller. The microcontroller with then process the analog data and transform it into useful information. The information will then be shown on the display.

The microcontroller used in this project is Arduino Uno. The analog sensors are pH sensor and TDS sensor. The microcomputer and camera used is Raspberry Pi Zero with camera module. The display is 1.3” IPS LCD Module. The test strip used is VANSFUL 14 in 1 test strip.

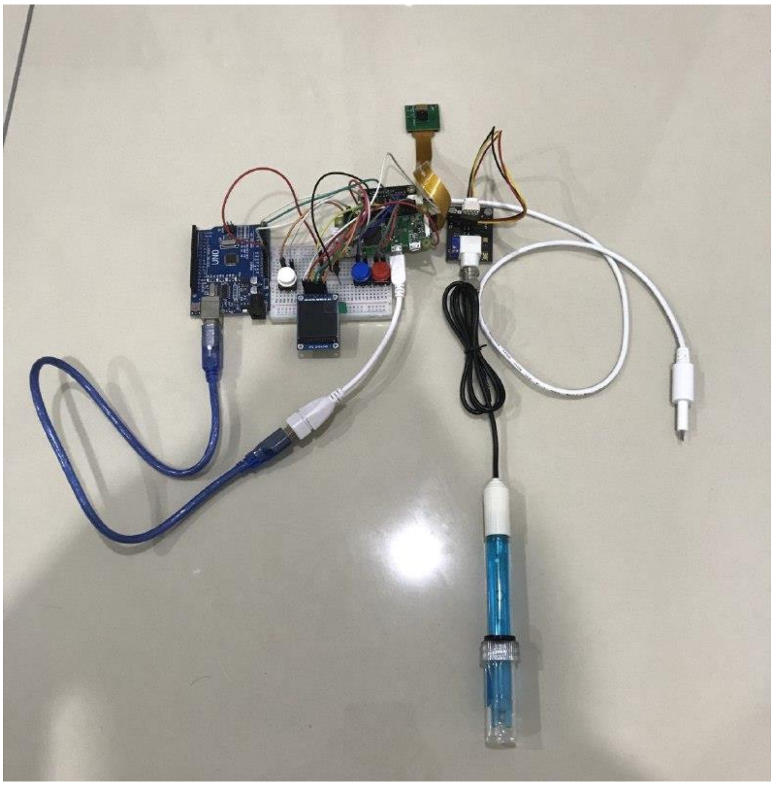
The connections of the water quality testing device is shown as below:

Table 4 Pin connections of water quality testing device

Pin		**Pin**
Arduino Uno 5V		pH sensor Vcc
Arduino Uno 5V		TDS sensor Vcc
Arduino Uno GND		pH sensor GND
Arduino Uno GND		TDS sensor GND
Arduino Uno A0		pH sensor analog
Arduino Uno A1		TDS sensor analog
Arduino Uno USB		Raspberry Pi Zero USB
Raspberry Pi Zero 3V3		LCD Module Vcc
Raspberry pi Zero GND	**connected to**	LCD Module GND
Raspberry pi Zero SCLK		LCD Module SCL
Raspberry pi Zero MOSI		LCD Module SDA (MOSI)
Raspberry pi Zero GPIO24		LCD Module RES
Raspberry pi Zero GPIO22		LCD Module DC
Raspberry pi Zero GND		Up button terminal B
Raspberry pi Zero GND		Down button terminal B
Raspberry pi Zero GND		Enter button terminal B
Raspberry pi Zero GPIO16		Up button terminal A

Raspberry pi Zero GPIO26		Down button terminal A
Raspberry pi Zero GPIO2		Enter button terminal A

Figure 8 Prototype of the water quality testing device



3.2 Instant Data Acquisition Process

The data of the VANSFUL 14 in 1 colour chart are extract manually and saved into a json file. The json file contains the first 9 set of data of the colour chart, as only the first 9 set of data is relevant. The first 9 set of data are free chlorine, pH, total alkalinity, hardness, iron, copper, lead, nitrate and nitrite. The json file has the exact numerical value and the RGB value of each set.

First, the test strip is place in front of a very dark background, and the camera will capture the image of the test strip. The image capture will be send to the Raspberry Pi Zero, and the image processing techniques are coded in Python. After that, the image will go through RGB to grayscale colour conversion and perspective transform. This is to get the exact test strip image, and ignore the background.

The RGB to grayscale algorithm used by the Python program is Luminance algorithm. This is because the Luminance algorithm has balanced weight on RGB colour, and it is close to how human perceive brightness. Other than that, it can operates fast and require less computational power.

The colours on the test strip are separated, and go through RGB to CIELAB colour conversion. After that, it will be compare to the corresponding colour chart data, which the colour chart data is converted from RGB to CIELAB too. By finding the CIE74 distance between the test strip colour data with its corresponding colour data in the colour chart, the delta-E is obtained. Based on this delta-E, the colour chart data with smallest delta-E will be saved, and then if the delta-E is more than 10, the value will be NIL, as the difference is big.

Based on the value obtained, a simple harmfulness algorithm is used to determine if the water quality is safe. If all the value are within the safe range, the water quality will be identify as ‘safe’. If some of the values are within the harm range, and it is less than 30% of all the values, the water quality will be identify as ‘careful’, as it requires user to pay more attention to it. If more than 30% of the values are within the harmful range, the water will be identify as ‘harmful’.

After that, the results will be save into a json file. The json file contains the date, time, geo-location, all the delta-E, the RGB values, and the exact values. Besides, the results can be seen on the LCD display. On the other hand, the date, time, geo-location, all the RGB values and the exact values will be saved into a cloud database, which is Google Sheet.

3.3 Continuous Data Acquisition Process

For continuous data acquisition, the pH sensor and the TDS sensor are being placed into water. The sensors will send the analog data to the Arduino Uno, and the program in the Arduino Uno will process those analog data into valuable information. Those information will be constantly updated as the sensors keep updating the analog data.

The Arduino Uno will first sample some analog values, and then get the averaged value. The reason of it is to prevent the sensors to be too sensitive to the analog data. For pH, the Arduino Uno will change convert the analog value to millivolts, then convert the millivolts into pH value. For TDS, the Arduino Uno will use a predefined library to convert the analog value to ppm. The pH value and ppm are valuable information.

The Arduino Uno will then package the information into 1 byte, and send it to the Raspberry Pi once every second. Based on the value obtained, a simple harmfulness algorithm is used to determine if the water quality is safe. If the pH value and the TDS value is within the safe range, the water will be identify as ‘safe’. If one of it is not within the safe range, the water will be identify as ‘careful’. If both the pH value and TDS value is not within the safe range, the water will be identify as ‘harmful’.

According to SDWF (n.d.), the safe drinking water pH value range within 6.5 and 8.5. The safe drinking water TDS value is lesser than 500 ppm.

At last, the Raspberry Pi will then show those information on the LCD display.

3.4 Experiment

3.4.1 Instant data acquisition

The water quality testing device is tested with 4 types of water. The types are filtered water, lab solution, carbonated water, and tap water.

Figure 9 Filtered water test strip

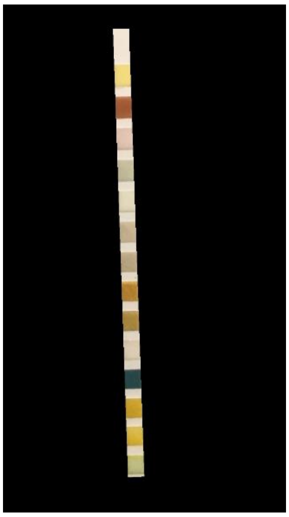


Figure 10 Lab solution test strip

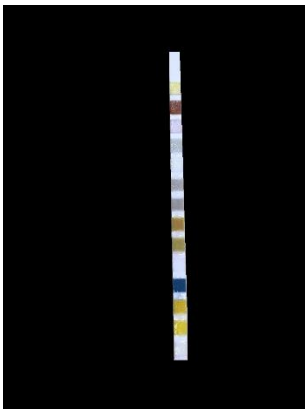


Figure 11 Carbonated water test strip

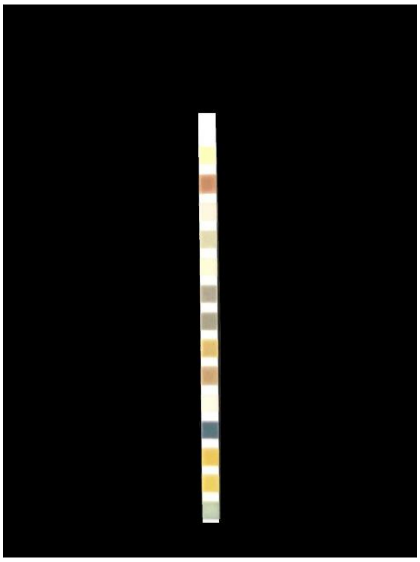
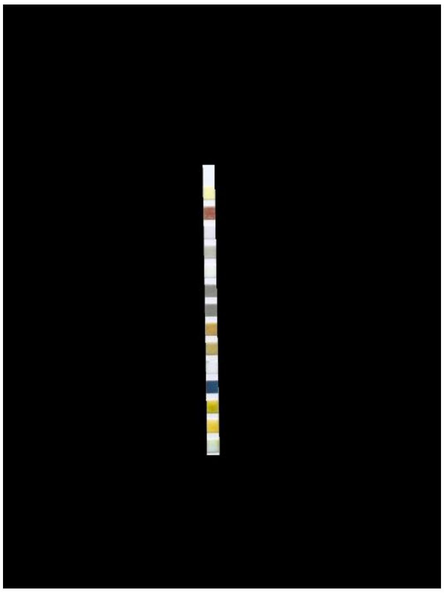


Figure 12 Tap water test strip



3.4.2 Continuous data acquisition

The water quality testing device is tested with 2 types of water, which are carbonated drink and filtered water.

Figure 13 Carbonated drink and filtered water

4 Results

4.1 Instant data acquisition

Figure 14 Filtered water output

Figure 15 Lab solution

Figure 16 Carbonated water output

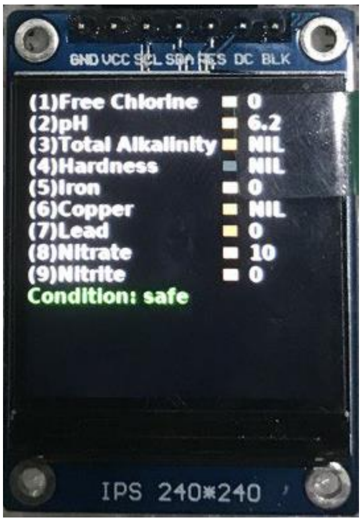


Figure 17 Tap water output

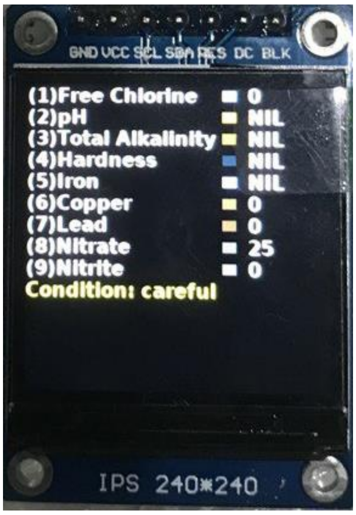


Figure 18 Water quality information in the device

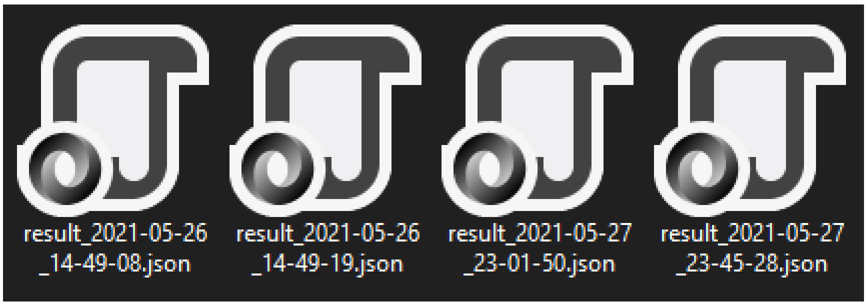
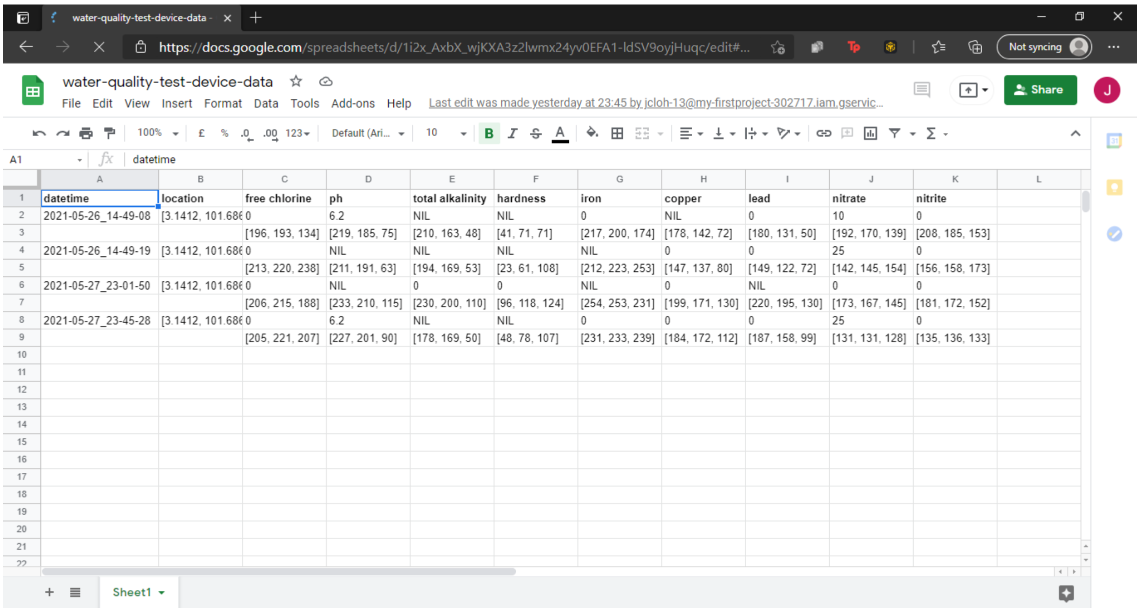


Figure 19 Water quality testing device on the cloud



datetime	location	free chlorine	ph	total alkalinity	hardness	iron	copper	lead	nitrate	nitrite
2021-05-26_14-49-08	[3.1412, 101.686	0	6.2	NIL	NIL	0	NIL	0	10	0
		[196, 193, 134]	[219, 185, 75]	[210, 163, 48]	[41, 71, 71]	[217, 200, 174]	[178, 142, 72]	[180, 131, 50]	[192, 170, 139]	[208, 185, 153]
2021-05-26_14-49-19	[3.1412, 101.686	0	NIL	NIL	NIL	NIL	0	0	25	0
		[213, 220, 238]	[211, 191, 63]	[194, 169, 53]	[23, 61, 108]	[212, 223, 253]	[147, 137, 80]	[149, 122, 72]	[142, 145, 154]	[156, 158, 173]
2021-05-27_23-01-50	[3.1412, 101.686	0	NIL	0	0	NIL	0	NIL	0	0
		[206, 215, 188]	[233, 210, 115]	[230, 200, 110]	[96, 118, 124]	[254, 253, 231]	[199, 171, 130]	[220, 195, 130]	[173, 167, 145]	[181, 172, 152]
2021-05-27_23-45-28	[3.1412, 101.686	0	6.2	NIL	NIL	0	0	0	25	0
		[205, 221, 207]	[227, 201, 90]	[178, 169, 50]	[48, 78, 107]	[231, 233, 239]	[184, 172, 112]	[187, 158, 99]	[131, 131, 128]	[135, 136, 133]

4.2 Continuous data acquisition

Figure 20 Carbonated drink output

Figure 21 Filtered water output

5 Discussion

5.1 Instant data acquisition

The results of 4 types of water are compared.

Table 5 Comparison of instant data acquisition results

Types of water	Filtered Water	Lab solution	Carbonated Water	Tap Water
Water Quality	Safe	Careful	Safe	Careful

By looking at the comparison, it is found out that the output is quite satisfactory, as the lab solution and tap water are marked ‘careful’, which means the user might need to pay more attention to the water. Other than that, the filtered water and carbonated water are marked as ‘safe’ by the water quality testing device, which means the user can consume the water and use the water safely.

Even though the results are quite satisfactory, but it can be improved. For example, the carbonated water has a pH level of 6.2, meanwhile normal carbonated water has pH level of 4.5. The performance can be improved by using more complex image processing methods, such as instead of using Luminance algorithm to convert image to grayscale, use Lightness algorithm, as Lightness algorithm have better imitation of how human perceive colours. However, with better and more complex algorithm, it will consume more computational power and might be more expensive in terms of time and money.

Other than that, it is shown that the water quality information is successfully saved into json file, and the information is uploaded to Google Sheet successfully.

5.2 Continuous data acquisition

The results of 2 types of water are compared.

Table 6 Comparison of continuous data acquisition results

Types of water	Carbonated drink	Filtered Water
Water Quality	Harmful	Careful

By looking at the comparison, it is found out that the water quality of continuous data acquisition method are tend to be more cautious compared to instant data acquisition. The quality of carbonated drink is identify as ‘harmful’, and the quality of filtered water is identify as ‘careful’. This might due to the fact that there are only 2 sensors to determine the overall harmfulness of the water, and it might also due to the strict safe water range.

In order to improve the performance, more sensors can be added to the Arduino Uno. Other than that, better heuristic function can determined to better define the water quality information.

6 Conclusion

In conclusion, the overall design of the water quality testing device is sufficient for home usage, especially in rural area where the water quality is often bad. The speed of the water quality testing device is fast as for instant mode, the water quality information can be obtained within a short time, instead of the traditional method that requires days to weeks. For continuous mode, the water quality information is updated per every second. The results of the water quality testing device is consider good, as it did not have any bad classification, such as when the water quality is bad and the device identify it as safe. However, the water quality testing device can be improved. The improvement can focus on the accuracy and the precision of the device, by implementing a better image processing methods for instant mode, and sample more data as well as increase the speed of data collecting for continuous mode.

7 References

Ahmad, I Moon, I & Shin, SJ 2018, ‘Color-to-grayscale algorithms effect on edge detection — A comparative study’, *2018 International Conference on Electronics, Information, and Communication (ICEIC)*, pp. 1-4.

Barham, S et al 2011, ‘Rapid Adaptive Needs Assessment (RANA) water quality kit’, *2011 IEEE Systems and Information Engineering Design Symposium*, pp.46- 46.

Cheny, J Freel, C Hoyer, B Jones, A Sehgal, A & Louis ,GE 2005, ‘Development of innovative techniques for testing and mapping private well water quality’, *2005 IEEE Design Symposium, Systems and Information Engineering*, pp.192-197.

Connolly, C & Fleiss, T 1997, ‘A study of efficiency and accuracy in the transformation from RGB to CIELAB color space’, *IEEE Transactions on Image Processing*, vol. 6, no. 7, pp. 1046-1048.

Harutyunyan, S Kaplanyan, T Kirakosyan, A & Momjyan, A 2020, ‘Design And Verification Of Autoconfigurable UART Controller’, *2020 IEEE 40th International Conference on Electronics and Nanotechnology (ELNANO)*, pp. 347-350.

Indu, K & Choondal, JJ 2016, "Modeling, development & analysis of low cost device for water quality testing," *2016 IEEE Annual India Conference (INDICON)*, pp. 1-6.

Jerom, BA & Manimegalai, R 2020, ‘An IoT Based Smart Water Quality Monitoring System using Cloud’, *2020 International Conference on Emerging Trends in Information Technology and Engineering (ic-ETITE*), pp. 1-7.

Leens, F 2009, ‘An introduction to I2C and SPI protocols’, *IEEE Instrumentation & Measurement Magazine*, vol. 12, no. 1, pp. 8-13.

Lin, SF Chen, JY and Chao, HX 2001, ‘Estimation of number of people in crowded scenes using perspective transformation’, *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, vol. 31, no. 6, pp. 645-654.

Mody, M et al. 2017, ‘Flexible and efficient perspective transform engine’, *2017 IEEE International Conference on Consumer Electronics-Asia (ICCE-Asia)*, pp. 111- 114.

Nguyen, D & Phung, PH 2017, ‘A Reliable and Efficient Wireless Sensor Network System for Water Quality Monitoring’, *2017 International Conference on Intelligent Environments (IE)*, pp. 84-91.

Raveendran, S Edavoor, PJ Kumar, YBN & Vasantha, MH 2018, ‘Design and Implementation of Reversible Logic based RGB to Gray scale Color Space Converter’, *TENCON 2018 - 2018 IEEE Region 10 Conference*, pp. 1813-1817.

Roland, J 2019, *Hard Water vs. Soft Water: Which One Is Healthier*, healthline, viewed 10 September 2020, < https://www.healthline.com/health/hard-water-and-soft-water>.

Sayed, ME Sammani, F & Albashier, MAM 2017, ‘An accurate method to calculate the color difference in a single image’, *2017 International Conference on Robotics, Automation and Sciences (ICORAS)*, pp. 1-3.

Sharma, M Agarwal, N and Reddy, SRN 2015, ‘Design and development of daughter board for USB-UART communication between Raspberry Pi and PC’, *International Conference on Computing, Communication & Automation*, pp. 944-948.

SDWF, *TDS AND PH*, Safe Drinking Water Foundation, viewed 10 September 2020, < https://www.safewater.org/fact-sheets-1/2017/1/23/tds-and-ph>.

Siam, M Munna, JI Hasan, M & Rahman, T 2019, ‘Remote Sensing Kit for Contamination Event Detection in Water’, *2019 IEEE R10 Humanitarian Technology Conference (R10- HTC)(47129)*, pp. 175-179.

Tseng, C & Lee, S 2018, ‘A Low-Light Color Image Enhancement Method on CIELAB Space’, *2018 IEEE 7th Global Conference on Consumer Electronics (GCCE)*, pp. 141-142.

World Health Organization 2019, *Drinking-Water*, World Health Organization, viewed 10 September 2020, < https://www.who.int/news-room/fact-sheets/detail/drinking-water#:~:text=Globally%2C%20at%20least%202%20billion,000%20diarrhoeal%20deaths%>.

Wu, Y Chen, Z 2016, ‘A detection method of road traffic sign based on inverse perspective transform’, *2016 IEEE International Conference of Online Analysis and Computing Science (ICOACS)*, pp. 293-296.

8 Appendices

8.1 Python code for instant mode

```
import cv2 as cv
import numpy as np
import matplotlib.pyplot as plt
import json
from datetime import datetime
import geocoder
import gspread

#https://docs.opencv.org/2.4/modules/imgproc/doc/miscellaneous_transformations.html
?highlight=cvtcolor#cvtcolor
def rgb_to_lab(color_array):
  r = color_array[0]/255
  g = color_array[1]/255
  b = color_array[2]/255

  x = (0.412453*r) + (0.357580*g) + (0.180423*b)
  y = (0.212671*r) + (0.715160*g) + (0.072169*b)
  z = (0.019334*r) + (0.119193*g) + (0.950227*b)

  x = x/0.950456
  z = z/1.088754
  if (y > 0.008856):
    L = (116*(y**(1/3))) - 16
  else:
    L = 903.3*y

  if (x > 0.008856):
    fx = x**(1/3)
  else:
    fx = 7.787*x + (16/116)

  if (y > 0.008856):
    fy = y**(1/3)
  else:
    fy = 7.787*y + (16/116)

  if (z > 0.008856):
    fz = z**(1/3)
  else:
    fz = 7.787*z + (16/116)

  delta = 128
  a = 500*(fx-fy) + delta
  b = 200*(fx-fz) + delta

  return [L,a,b]
```

```
def CIE76(lab1, lab2):
  L1 = lab1[0]
  a1 = lab1[1]
  b1 = lab1[2]

  L2 = lab2[0]
  a2 = lab2[1]
  b2 = lab2[2]

  delta_e = (((L2-L1)**2) + ((a2-a1)**2) + ((b2-b1)**2))**(1/2)
  return round(delta_e)

#test_strip_bgr = cv.imread("ori_iPhone_data.jpg")
#test_strip_bgr = cv.imread("dr_then_data.jpg")
#test_strip_bgr = cv.imread("cola_data2.jpg")
test_strip_bgr = cv.imread("sepang_water_data2.jpg")
test_strip_rgb = cv.cvtColor(test_strip_bgr, cv.COLOR_BGR2RGB) #use for
plt.imshow
#cv.imshow("test_strip_bgr",test_strip_bgr)

test_strip_gray = cv.cvtColor(test_strip_bgr, cv.COLOR_BGR2GRAY)
#cv.imshow("test_strip_gray",test_strip_gray)
plt.figure()
plt.subplot(1,2,1)
plt.imshow(255-test_strip_gray,cmap='Greys') #255-pic because plt shows value
different with opencv

#threshold
#black is 255, white is 0 (opencv)
_,test_strip_binary = cv.threshold(test_strip_gray,50,255,cv.THRESH_BINARY)
#cv.imshow("test_strip_binary",test_strip_binary)
plt.subplot(1,2,2)
plt.imshow(255-test_strip_binary,cmap='Greys')
#plt.show()

plt.figure()
test_strip_rgb_copy = test_strip_rgb.copy()
contours, _ = cv.findContours(test_strip_binary, cv.RETR_EXTERNAL,
cv.CHAIN_APPROX_NONE)
#print(np.shape(contours))
#cv.drawContours(test_strip_rgb_copy, contours, -1, (0,255,0), 3)
#'''
for idx, c in enumerate(contours):
  (x,y,w,h) = cv.boundingRect(c)
  cv.rectangle(test_strip_rgb_copy,(x,y),(x+w,y+h),(0,255,0),3)

  #https://docs.opencv.org/3.4/dd/d49/tutorial_py_contour_features.html
  rect = cv.minAreaRect(c)
  box = cv.boxPoints(rect)
```

```
  ori_box = box
  #print(ori_box)
  box = np.int0(box)
  #print(box)
  cv.drawContours(test_strip_rgb_copy,[box],0,(0,0,255),3)
#'''
plt.subplot(1,2,1)
plt.imshow(test_strip_rgb_copy)

#do perspective transform on the test strip
#by printing the box [bot_left, top_left, top_right, bot_right]
ori_box = np.array(ori_box).astype(np.float32)
dest_box = []
dest_box.append([0,0+h])#bot_left
dest_box.append([0,0])#top_left
dest_box.append([0+w,0])#top_right
dest_box.append([0+w,0+h])#bot_right
dest_box = np.array(dest_box).astype(np.float32)

T = cv.getPerspectiveTransform(ori_box, dest_box)
only_test_strip = cv.warpPerspective(test_strip_rgb, T, (w,h))

#rotate 180 degree
M = cv.getRotationMatrix2D((w/2,h/2), 180, 1)
only_test_strip_rotated = cv.warpAffine(only_test_strip, M, (w,h))
only_test_strip_rotated = cv.GaussianBlur(only_test_strip_rotated,(7,7),0)

plt.subplot(1,2,2)
plt.imshow(only_test_strip_rotated)

#plt.show()

#divide the teststrip into 15 parts, where 15th is useless
#https://stackoverflow.com/questions/15589517/how-to-crop-an-image-in-opencv-
using-python
n = 15
test_strip_size = only_test_strip_rotated.shape
#print(test_strip_size)
h, w, _ = test_strip_size
new_h = int(h/n)
new_w = w
img_list = []
for i in range(n):
  img_list.append(only_test_strip_rotated[i*new_h:(i*new_h)+new_h, 0:new_w])
#Colorpsace = RGB

#for i in range(n):
#  cv.imwrite("part_{}.png".format(i),cv.cvtColor(img_list[i],
cv.COLOR_RGB2BGR))
```

```
#test on 0 to 8 only
with open("./ori_chart.json") as file:
  chart_data = json.load(file)

result = {}
result["datetime"] = "{}".format(datetime.now().strftime("%Y-%m-%d_%H-%M-
%S"))
result["location"] = geocoder.ip("me").latlng
for i in range(9):
  result[str(i)] = {"delta_e":-1,\
                    "value":-1}
for i in range(9):
  data = chart_data[str(i)]
  img = img_list[i]
  h, w, _ = img.shape
  center_y = int(h/2)
  center_x = int(w/2)
  img_center_rgb = img[center_y,center_x].tolist()
  result[str(i)]["rgb_value"] = img_center_rgb
  img_center_lab = rgb_to_lab(img_center_rgb)
  current_min_delta_e = 442 #sqrt(3*(255^2))

  for key in data.keys():
     current_data_lab = rgb_to_lab(data[key])
     delta_e = CIE76(img_center_lab,current_data_lab)
     #print(delta_e)
     if (delta_e < current_min_delta_e):
       current_min_delta_e = delta_e
       result[str(i)]["delta_e"] = delta_e
       result[str(i)]["value"] = key

  for key in data.keys():
    if (result[str(i)]["delta_e"] > 10):
       result[str(i)]["value"] = "NIL"
  #break

print(result)
#https://www.geeksforgeeks.org/how-to-convert-python-dictionary-to-json/
output_filename = "result_{}.json".format(result["datetime"])
with open(output_filename, "w") as file:
  json.dump(result, file)

#check water harmness
with open(output_filename) as file:
  all_data = json.load(file)

  harmness = []
  for i in range(9): #only 9 data was collected
    if all_data[str(i)]["value"] != "NIL":
      data_value = int(float(all_data[str(i)]["value"]))
```

```
    if (i == 0):
      if (data_value >= 3 and data_value <=20):
        harmness.append(1)
      else:
        harmness.append(0)

    if (i == 2):
      if (data_value >= 180 and data_value <= 240):
        harmness.append(1)
      else:
        harmness.append(0)

    if (i == 4):
      if (data_value >= 5 and data_value <= 500):
        harmness.append(1)
      else:
        harmness.append(0)

    if (i == 5):
      if (data_value >= 1 and data_value <= 300):
          harmness.append(1)
      else:
        harmness.append(0)

    if (i == 6):
      if (data_value >= 20 and data_value <= 500):
          harmness.append(1)
      else:
        harmness.append(0)

    if (i == 7):
      if (data_value >= 25 and data_value <= 50):
          harmness.append(1)
      else:
        harmness.append(0)

    if (i == 8):
      if (data_value >= 5 and data_value <= 80):
          harmness.append(1)
      else:
        harmness.append(0)


print(harmness)
sum_of_harmness = 0
for i in harmness:
    sum_of_harmness += i

if sum_of_harmness == 0:
    print("Safe")
```

```
  elif (sum_of_harmness/len(harmness) <= 0.3):
      print("Careful")

  else:
      print("Harmful")

#upload to google sheet
gc = gspread.service_account(filename="./google_sheet_credentials.json")
sh = gc.open("water-quality-test-device-data")
worksheet = sh.sheet1

res = worksheet.get_all_values()
with open(output_filename) as file:
  all_data = json.load(file)

  sheet_data = [all_data["datetime"], str(all_data["location"])]
  for i in range(9):
      sheet_data.append(all_data[str(i)]["value"])

  print(sheet_data)
  worksheet.append_row(sheet_data)
  rgb_data = ["",""]
  for i in range(9):
      rgb_data.append(str(all_data[str(i)]["rgb_value"]))
  print(rgb_data)
  worksheet.append_row(rgb_data)

plt.show()
```

8.2 Arduino code for continuous mode

```
#include <EEPROM.h>
#include <String.h>
#include "GravityTDS.h"

#define TdsSensorPin A1
GravityTDS gravityTds;

#define SensorPin A0          // the pH meter Analog output is connected with the
Arduino's Analog
unsigned long int avgValue;  //Store the average value of the sensor feedback
float b;
int buf[10],temp;

float temperature = 25,tdsValue = 0;

void setup()
{
```

```
  Serial.begin(9600);
  gravityTds.setPin(TdsSensorPin);
  gravityTds.setAref(5.0);  //reference voltage on ADC, default 5.0V on Arduino UNO
  gravityTds.setAdcRange(1024);  //1024 for 10bit ADC;4096 for 12bit ADC
  gravityTds.begin();  //initialization
  //Serial.println("Ready");    //Test the serial monitor
}
void loop()
{
  for(int i=0;i<10;i++)       //Get 10 sample value from the sensor for smooth the value
  {
    buf[i]=analogRead(SensorPin);
    delay(10);
  }
  for(int i=0;i<9;i++)        //sort the analog from small to large
  {
    for(int j=i+1;j<10;j++)
    {
      if(buf[i]>buf[j])
      {
        temp=buf[i];
        buf[i]=buf[j];
        buf[j]=temp;
      }
    }
  }
  avgValue=0;
  for(int i=2;i<8;i++)                      //take the average value of 6 center sample
    avgValue+=buf[i];
  float phValue=(float)avgValue*5.0/1024/6; //convert the analog into millivolt
  phValue=3.5*phValue;                      //convert the millivolt into pH value
  //Serial.print("pH:");
  //Serial.print(phValue,2);
  //Serial.println();

  gravityTds.setTemperature(temperature);  // set the temperature and execute
temperature compensation
  gravityTds.update();  //sample and calculate
  tdsValue = gravityTds.getTdsValue();  // then get the value
  //Serial.print(tdsValue,0);
  //Serial.println("ppm");
  //Serial.println("\n");0
  String data = String(phValue,2) + "," + String(tdsValue,0) + "\n";
  Serial.write(data.c_str());
  delay(1000);
  //delay(800);

}
```

8.3 Python code of overall combination

```
#main program
import ST7789

import RPi.GPIO as GPIO
from PIL import Image
from PIL import ImageDraw
from PIL import ImageFont

import time
import random

import os
import json
import serial

BAUD = 9600
TIMEOUT = 3#s timeout

serial_port = "/dev/ttyUSB0"

disp = ST7789.ST7789(
    port=0,
    cs=ST7789.BG_SPI_CS_FRONT,
    dc=25, #data command pin (GPIO pin num)
    rst=24, #reset pin (GPIO pin num)
    #backlight=13, #this is not needed, as not connected
    mode=3,
    spi_speed_hz=80 * 1000 * 1000
)

WIDTH = disp.width
HEIGHT = disp.height

image_data_names = ["Free Chlorine",\
                    "pH",\
                    "Total Alkalinity",\
                    "Hardness",\
                    "Iron",\
                    "Copper",\
                    "Lead",\
                    "Nitrate",\
                    "Nitrite"]
up_switch = 16#GPIO16
down_switch = 26#GPIO26
enter_switch = 2#GPIO2

GPIO.setmode(GPIO.BCM)
GPIO.setup(up_switch, GPIO.IN, pull_up_down=GPIO.PUD_UP)
```

```
GPIO.setup(down_switch, GPIO.IN, pull_up_down=GPIO.PUD_UP)
GPIO.setup(enter_switch, GPIO.IN, pull_up_down=GPIO.PUD_UP)

img = Image.new("RGB", (WIDTH, HEIGHT), color=(0,0,0)) #background
draw = ImageDraw.Draw(img) #enable drawing on the background
font = ImageFont.truetype("/usr/share/fonts/truetype/dejavu/DejaVuSans-Bold.ttf", 15)

GPIO.add_event_detect(up_switch, GPIO.FALLING)
GPIO.add_event_detect(down_switch, GPIO.FALLING)
GPIO.add_event_detect(enter_switch, GPIO.FALLING)

WHITE = (255,255,255)
GREY = (180,180,180)
BLACK = (0,0,0)
GREEN = (0,255,0)
YELLOW = (255,255,0)
RED = (255,0,0)

files = os.listdir()
result_jsons = []
for file in files:
    if (file[:6] == "result" and file[-5:] == ".json"):
        result_jsons.append(file)

disp.begin()
def start_screen():
    #clear screen
    draw.rectangle((0, 0, WIDTH, HEIGHT), BLACK)

    #initial screen
    prev_text = "Previous data"
    new_text = "New Test"

    selection = 0

    font_x,font_y = draw.textsize(prev_text, font)
    draw.rectangle((0, 0, WIDTH, font_y), GREY)
    draw.text((0,0),prev_text,font=font, fill=WHITE)
    draw.text((0,font_y),new_text,font=font, fill=WHITE)

    disp.display(img)
    while(1):
        disp.display(img)
        if(GPIO.event_detected(up_switch)):
            selection = 0
            draw.rectangle((0, 0, WIDTH, HEIGHT), BLACK)
            draw.rectangle((0, 0, WIDTH, font_y), GREY)
            draw.text((0,0),prev_text,font=font, fill=WHITE)
            draw.text((0,font_y),new_text,font=font, fill=WHITE)
```

```
        if(GPIO.event_detected(down_switch)):
            selection = 1
            draw.rectangle((0, 0, WIDTH, HEIGHT), BLACK)
            draw.rectangle((0, font_y, WIDTH, font_y*2), GREY)
            draw.text((0,0),prev_text,font=font, fill=WHITE)
            draw.text((0,font_y),new_text,font=font, fill=WHITE)

        if (GPIO.event_detected(enter_switch)):
            return selection
            break

        disp.display(img)
    pass

def show_prev_data():
    #clear screen
    draw.rectangle((0, 0, WIDTH, HEIGHT), BLACK)

    #initial screen
    selection = 0

    for i,filename in enumerate(result_jsons):
        font_x,font_y = draw.textsize(filename, font)
        if (i == selection):
            draw.rectangle((0, selection*font_y, WIDTH, (selection*font_y)+font_y),
GREY)
        draw.text((0,i*font_y),filename.strip(".json"),font=font, fill=WHITE)

    disp.display(img)
    while(1):
        if(GPIO.event_detected(up_switch) and selection !=0):
            selection -= 1
            draw.rectangle((0, 0, WIDTH, HEIGHT), BLACK)
            draw.rectangle((0, selection*font_y, WIDTH, (selection*font_y)+font_y),
GREY)
            for i,filename in enumerate(result_jsons):
                font_x,font_y = draw.textsize(filename, font)
                draw.text((0,i*font_y),filename.strip(".json"),font=font, fill=WHITE)

        if(GPIO.event_detected(down_switch) and selection !=len(result_jsons)-1):
            selection += 1
            draw.rectangle((0, 0, WIDTH, HEIGHT), BLACK)
            draw.rectangle((0, selection*font_y, WIDTH, (selection*font_y)+font_y),
GREY)
            for i,filename in enumerate(result_jsons):
                font_x,font_y = draw.textsize(filename, font)
                draw.text((0,i*font_y),filename.strip(".json"),font=font, fill=WHITE)

        if (GPIO.event_detected(enter_switch)):
            return selection
```

```
        disp.display(img)
    pass

def show_data_selected(selection):
    #clear screen
    draw.rectangle((0, 0, WIDTH, HEIGHT), BLACK)

    filename = result_jsons[selection]

    font_sizes = []
    max_font_x = 0
    max_font_y = 0
    for i,name in enumerate(image_data_names):
        font_x,font_y = draw.textsize("({}){}".format(i,name), font)
        font_sizes.append((font_x,font_y))
        if (font_x > max_font_x):
            max_font_x = font_x
        if (font_y > max_font_y):
            max_font_y = font_y

    rect_x_offset = 5
    rect_y_offset = 5

    with open(filename) as file:
        all_data = json.load(file)

        for i in range(len(image_data_names)):
            draw.text((0,int(i*max_font_y)),"({}){}".format(i+1,image_data_names[i]), 
font=font, fill=(255,255,255))
            draw.rectangle([(int(max_font_x+rect_x_offset),\
                             int((i*max_font_y)+rect_y_offset)),\
                            ((int(max_font_x+rect_x_offset+10),\
                              int((i*max_font_y)+max_font_y-
rect_y_offset)))],tuple(all_data[str(i)]["rgb_value"]))
            draw.text((int(max_font_x+rect_x_offset*5),int((i*max_font_y))), 
all_data[str(i)]["value"], font=font, fill=WHITE)

        harmness = [0]*9
        harmness = []
        for i in range(9): #only 9 data was collected
            if all_data[str(i)]["value"] != "NIL":
                data_value = int(float(all_data[str(i)]["value"]))
                if (i == 0):
                    if (data_value >= 3 and data_value <=20):
                        harmness.append(1)
                    else:
                        harmness.append(0)

                if (i == 2):
```

```
            if (data_value >= 180 and data_value <= 240):
                harmness.append(1)
            else:
                harmness.append(0)
        if (i == 4):
            if (data_value >= 5 and data_value <= 500):
                harmness.append(1)
            else:
                harmness.append(0)

        if (i == 5):
            if (data_value >= 1 and data_value <= 300):
                harmness.append(1)
            else:
                harmness.append(0)

        if (i == 6):
            if (data_value >= 20 and data_value <= 500):
                harmness.append(1)
            else:
                harmness.append(0)

        if (i == 7):
            if (data_value >= 25 and data_value <= 50):
                harmness.append(1)
            else:
                harmness.append(0)
        if (i == 8):
            if (data_value >= 5 and data_value <= 80):
                harmness.append(1)
            else:
                harmness.append(0)
    #print(harmness)
    sum_of_harmness = 0
    for i in harmness:
        sum_of_harmness += i

    if sum_of_harmness == 0:
        #print("Safe")
        water_condition = "safe"
        draw.text((0,len(image_data_names)*max_font_y),"Condition:
{}".format(water_condition),font=font,fill=GREEN)

    elif (sum_of_harmness/len(harmness) <= 0.3):
        #print("Careful")
        water_condition = "careful"
        draw.text((0,len(image_data_names)*max_font_y),"Condition:
{}".format(water_condition),font=font,fill=YELLOW)

    else:
```

```
            #print("Harmful")
            water_condition = "harmful"
            draw.text((0,len(image_data_names)*max_font_y),"Condition:
{}".format(water_condition),font=font,fill=RED)
        '''
        for i in range(9): #only 9 data was collected
            data_value = int(float(all_data[str(i)]["value"]))
            if (i == 0):
                if (data_value >= 3 and data_value <=20):
                    harmness[i] = 1

            if (i == 2):
                if (data_value >= 180 and data_value <= 240):
                    harmness[i] = 1

            if (i == 4):
                if (data_value >= 5 and data_value <= 500):
                    harmness[i] = 1

            if (i == 5):
                if (data_value >= 1 and data_value <= 300):
                    harmness[i] = 1

            if (i == 6):
                if (data_value >= 20 and data_value <= 500):
                    harmness[i] = 1

            if (i == 7):
                if (data_value >= 25 and data_value <= 50):
                    harmness[i] = 1

            if (i == 8):
                if (data_value >= 5 and data_value <= 80):
                    harmness[i] = 1

        sum_of_harmness = 0
        water_condition = ""
        for i in harmness:
            sum_of_harmness += i

        if sum_of_harmness == 0:
            #print("Safe")
            water_condition = "safe"
            draw.text((0,len(image_data_names)*max_font_y),"Condition:
{}".format(water_condition),font=font,fill=GREEN)

        elif (sum_of_harmness <=3):
            #print("Careful")
            water_condition = "careful"
```

```
            draw.text((0,len(image_data_names)*max_font_y),"Condition:
{}".format(water_condition),font=font,fill=YELLOW)

        else:
            #print("Harmful")
            water_condition = "harmful"
            draw.text((0,len(image_data_names)*max_font_y),"Condition:
{}".format(water_condition),font=font,fill=RED)
        '''
        while(1):
            disp.display(img)
            if (GPIO.event_detected(enter_switch)):
                break
    pass

def new_testing():
    #clear screen
    draw.rectangle((0, 0, WIDTH, HEIGHT), BLACK)

    #initial screen
    instant_text = "Instant (Image)"
    continuous_text  = "Continuous (Sensor)"

    selection = 0

    font_x,font_y = draw.textsize(instant_text, font)
    draw.rectangle((0, 0, WIDTH, font_y), GREY)
    draw.text((0,0),instant_text,font=font, fill=WHITE)
    draw.text((0,font_y),continuous_text,font=font, fill=WHITE)
    disp.display(img)
    while(1):
        if(GPIO.event_detected(up_switch)):
            selection = 0
            draw.rectangle((0, 0, WIDTH, HEIGHT), BLACK)
            draw.rectangle((0, 0, WIDTH, font_y), GREY)
            draw.text((0,0),instant_text,font=font, fill=WHITE)
            draw.text((0,font_y),continuous_text,font=font, fill=WHITE)

        if(GPIO.event_detected(down_switch)):
            selection = 1
            draw.rectangle((0, 0, WIDTH, HEIGHT), BLACK)
            draw.rectangle((0, font_y, WIDTH, font_y*2), GREY)
            draw.text((0,0),instant_text,font=font, fill=WHITE)
            draw.text((0,font_y),continuous_text,font=font, fill=WHITE)

        if (GPIO.event_detected(enter_switch)):
            return selection
            break

        disp.display(img)
```

```
    pass

def instant_testing():
    pass

def sensor_testing():

    #clear screen
    draw.rectangle((0, 0, WIDTH, HEIGHT), BLACK)
    ser = serial.Serial(serial_port, BAUD, timeout = TIMEOUT)
    while(1):
        draw.rectangle((0, 0, WIDTH, HEIGHT), BLACK)
        value = ser.readline()
        value_list = value.decode('utf-8').strip().split(",")

        pH = value_list[0].strip()
        TDS = value_list[1].strip()

        pH_text = "pH: {}".format(pH)
        TDS_text = "TDS: {}ppm".format(TDS)

        font_x,font_y = draw.textsize(pH_text, font)
        draw.text((0,0),pH_text,font=font, fill=WHITE)
        draw.text((0,font_y),TDS_text,font=font, fill=WHITE)

        sum_of_harmness = 0
        if not(float(pH.strip()) >= 6.5 and float(pH.strip()) <= 8.5):
            sum_of_harmness += 1
        if (int(TDS.strip()) > 500):
             sum_of_harmness += 1

        #print(sum_of_harmness)
        if sum_of_harmness == 0:
            #print("Safe")
            water_condition = "safe"
            draw.text((0,2*font_y),"Condition: 
{}".format(water_condition),font=font,fill=GREEN)

        elif (sum_of_harmness == 1):
            #print("Careful")
            water_condition = "careful"
            draw.text((0,2*font_y),"Condition: 
{}".format(water_condition),font=font,fill=YELLOW)

        else:
            #print("Harmful")
            water_condition = "harmful"
            draw.text((0,2*font_y),"Condition: 
{}".format(water_condition),font=font,fill=RED)
```

```
        if (GPIO.event_detected(enter_switch)):
            break
        disp.display(img)

    pass

def main():
    selection = start_screen()
    if (selection == 0):
        data_selection = show_prev_data()
        show_data_selected(data_selection)
    if (selection == 1):
        test_selection = new_testing()
        if (test_selection == 0):
            instant_testing()
        if (test_selection == 1):
            sensor_testing()
    pass

if __name__ == "__main__":
    while(1):
        main()
    GPIO.cleanup()
```