SCHOOL OF ELCTRICAL & ELECTRONIC ENGINEERING

UNIVERSITI SAINS MALAYSIA (ENGINEERING CAMPUS)

ACADEMIC 2020 / 2021

YEAR 4 SEMESTER I

EEM422 MACHINE VISION

MINI PROJECT:

AUTOMATED METER READING

TEAM NAME: AUTOSEREPET

MEMBERS:

1. **PHIROM SASOMSAP A/L EHDI (139491)**
2. **MUHAMMAD AQIL UZAIR BIN ISMAIL (139483)**

LECTURER:

1. **ASSOC PROF DR KHOO BEE EE**
2. **DR NOR RIZUAN BIN MAT NOOR**

SUBMISSION DATE: 25 JAN 2021

1. WORK DISTRIBUTION

Phirom Sasomsap A/L Ehdi (139491)

- Capture the image from the from USM TNB Meter for implementation into the programing.
- Design the coding for K-Nearest Neighbour (KNN) Character Recognition.
- Design the coding for Trackbar Brightness & Contrast.
- Design Graphical User Interface (GUI) of the system.
- Video Demonstration
- Report: Methodology, Results & Discussion, References.

Muhammad Aqil Uzair Bin Ismail (139483)

- Report: Abstract, Introduction, Future Suggestion, Conclusion.

2. ABSTRACT

An automated meter reading system was proposed because during Covid-19 pandemic meter readers prevented from working during movement control order (MCO). The concept is to capture meter image remotely and then extract data from it. Normally, meter readers will come to customers residential area and went from one house to another to read the electric meter and key in the data into the system to generate paper bill. C++ functions and OpenCV algorithms are implemented to develop the system because of its practicality. The benefits of this system are it can increase sustainability, it can increase efficiency and it can increase productivity.

3. INTRODUCTION

During COVID19, TNB meter and water meter reading are not able to be performed. The objective of this project is to demonstrate machine vision can be utilized to enable meter reading remotely. The scope of this project is to capture the image and interpret the reading. It does not cover the transmission of the information.

In Malaysia, every month citizen need to pay for electricity bill and water bill. At every house, there are water meter and TNB meter that are used to record the usage of electricity and water. The TNB meter and water meter are located outside the house so that it will be easy for staff of TNB and water management company to check the reading. There are two types of these meters that are analog meter and digital meter. At house that are built within these past five years usually these house use digital meter while the old house still use the analog meter.

Before the current pandemic, TNB staff and the staff of water will come to every house every month and read the reading at the meter. The staff bring an electronic device that will calculate the usage and print the bill and then the staff will put the bill into each house respective mailbox. Before the bill is calculated the staff need to read the meter and key in manually into the electronic device. Therefore, during the pandemic that practice of going from home to home need to be stopped for a while because of the movement control order enforced by the government to control the pandemic. So, for a while the meter reading could not be performed because during the movement control order only essential services are allowed to operate. Although, electric and water are part of essential services, the number of workers allowed to work need to be reduced. So, only workers that are important to maintain the operation of essential service continue to work.

This may be not a problem to the consumers because they can continue to enjoy these two services as usual but for the service providers they may experience some difficulty to sustain the operation. Since they could not obtain the consumer usage directly, they could not charge the consumers with electricity bill every month that lead to the problem mentioned earlier. Using the application of machine vision, this problem can be solved.

Machine vision is technology used to provide imaging-based automatic inspection and analysis for specific applications. By using machine vision, data analysis can be performed effectively and accurately. According to newspaper report by Berita Harian on June 2020, there are 6.3 million registered TNB consumers. Data analysis from all this consumers must be done effectively and accurately. So, proposed solution in this paper is using machine vision application to capture image remotely and to interpret data from it. To capture image remotely, the consumers will use their own smartphone to capture the meter image and send it to the TNB and water company server and then the computer will perform the data reading from the image. In this paper the transmission of image from the consumer smartphone to these two service provider will not be discussed. To interpret data from the image, OpenCV is used to create a system to analyse the image. OpenCV is an open source computer vision and machine learning software library.

4. METHODOLOGY

Trackbar Brightness & Contrast

The trackbar or the slider adjustment are added to the image to control the contrast and the brightness of an image.

```
int iSliderValue1 = 50;
createTrackbar("Brightness", "TNB Meter", &iSliderValue1, 200);
```

The coding above is for brightness slider adjustment. We have set the brightness value as 200 which is able to detect the image of TNB Meter even at night.

```
int iSliderValue2 = 50;
createTrackbar("Contrast", "My Window", &iSliderValue2, 200);
```

Similar with the contrast, we have set the value to 200 which is to reduce the intensity of an image.

KNN Character Recognition

This Project uses the K-Nearest Neighbor approach to train the system. The program is trained to classify different kinds of numbers. First we need to create the image training number for the purpose to detect the number from TNB Meter. The image below shown the image training number that we have created.

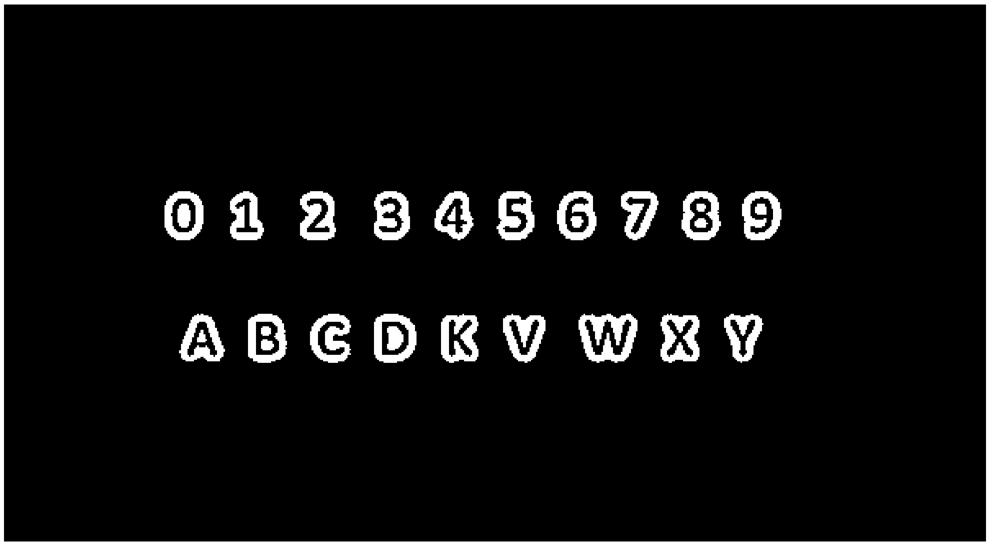


Figure 1: Image of training number

We need to train all the number into one file. The to generate the data is attached in zip.file. The figure below shown how to determine the correct data.

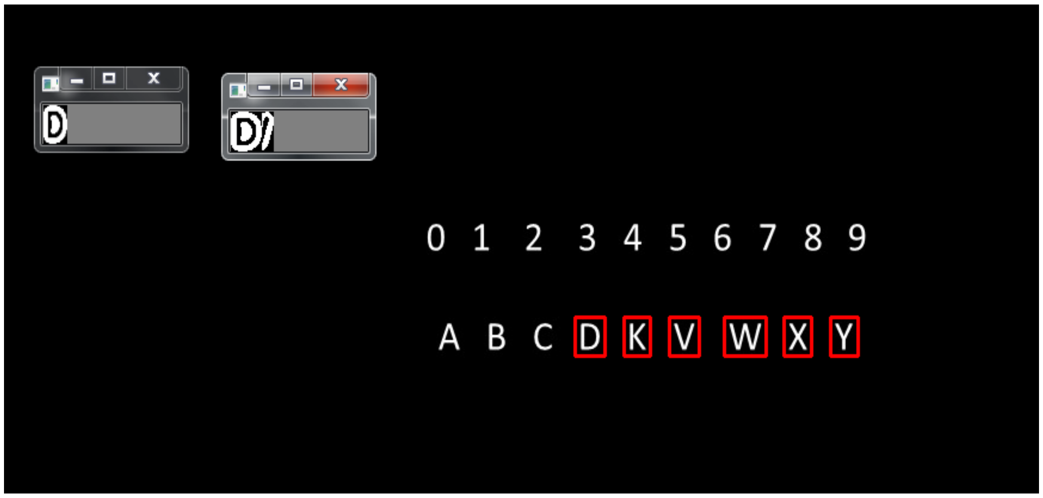
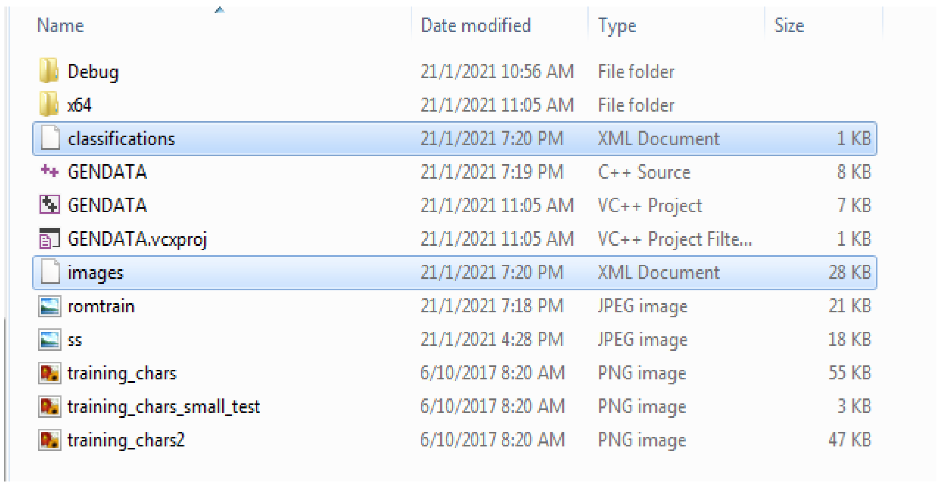


Figure2: Set the correct data into one file

So now we have data file (clasifications and image). From here, we need move the both file to the our project(Miniproject file) for number or alphabets detection of an image.



Name	Date modified	Type	Size
Debug	21/1/2021 10:56 AM	File folder	
x64	21/1/2021 11:05 AM	File folder	
classifications	21/1/2021 7:20 PM	XML Document	1 KB
GENDATA	21/1/2021 7:19 PM	C++ Source	8 KB
GENDATA	21/1/2021 11:05 AM	VC++ Project	7 KB
GENDATA.vcxproj	21/1/2021 11:05 AM	VC++ Project Filte...	1 KB
images	21/1/2021 7:20 PM	XML Document	28 KB
romtrain	21/1/2021 7:18 PM	JPEG image	21 KB
ss	21/1/2021 4:28 PM	JPEG image	18 KB
training_chars	6/10/2017 8:20 AM	PNG image	55 KB
training_chars_small_test	6/10/2017 8:20 AM	PNG image	3 KB
training_chars2	6/10/2017 8:20 AM	PNG image	47 KB

Figure3: The KNN data file.

Graphical User Interface (GUI)

Graphical user interface (GUI) is designed in this project to display the output which allow the user to press the button. To implement the GUI system, we need to create the cvui.h file by creating the new file and add all library into cvui.h file (the link for cvui.h library is provided at the references). After that we need to declare the coding as shown below and the we can continue with our GUI's coding.

```
#define CVUI_IMPLEMENTATION
#include "cvui.h"
```

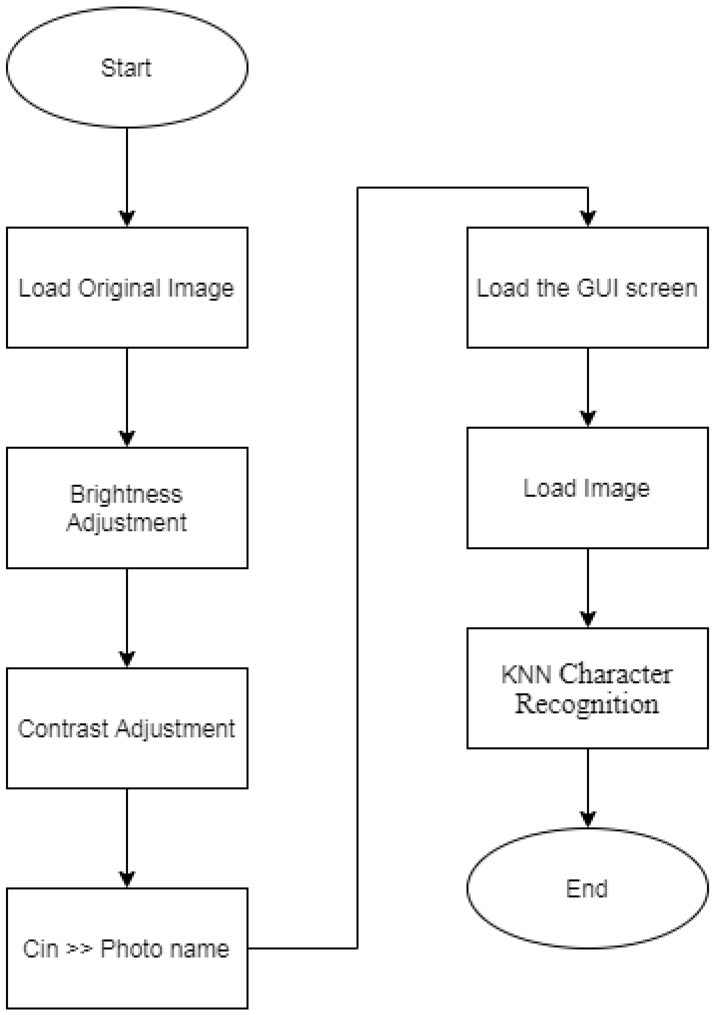


Figure 4: Flowchart of the overall image processing algorithm

5. RESULTS&DISCUSSION

When the program was started, the image of TNB Meter with brightness and contrast adjustment will display on the screen. The image that have captured is not clearly, since the meter reading also can't be read exactly. The figure below shown the original image that have captured and attached with the brightness and contrast trackbar.

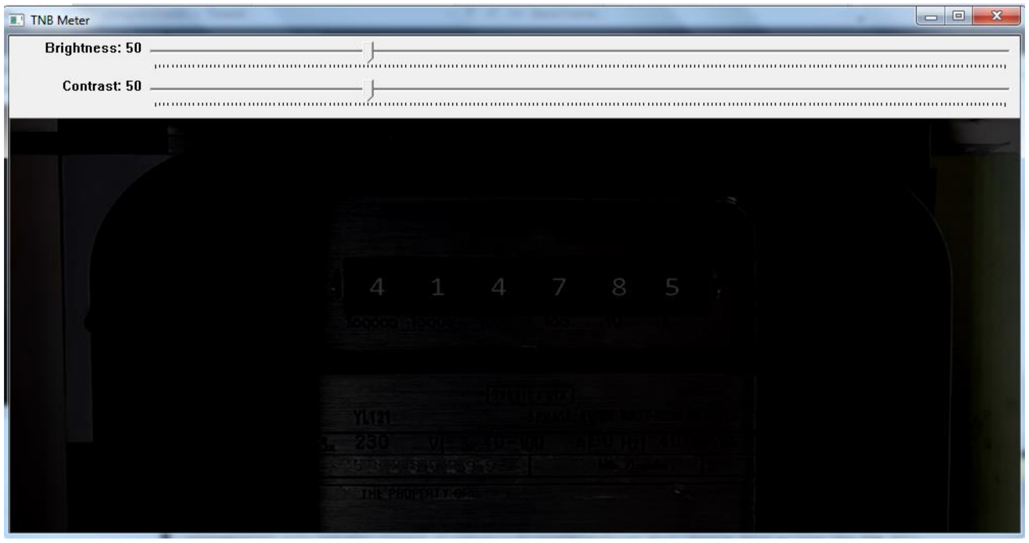


Figure 5: The image of TNB Meter

Whenever the user change the position of a trackbar, the value of an integer variable is also changed. Using that value, we can change a property of an image. From here, we have adjusted the brightness to 75 and the contrast is adjusted to 193. The observation is recorded as below which is the image is become clearly to be seen since the meter reading also look better than original of an image.

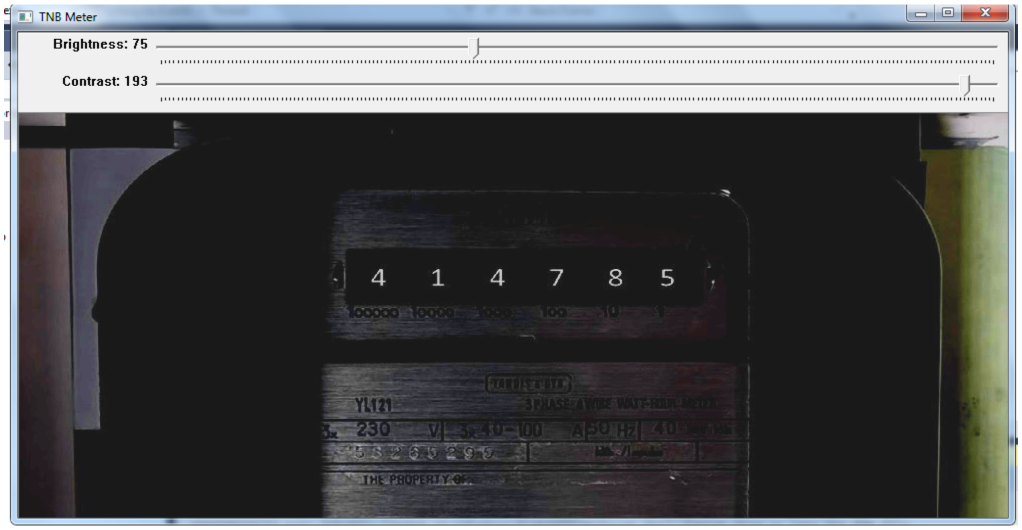
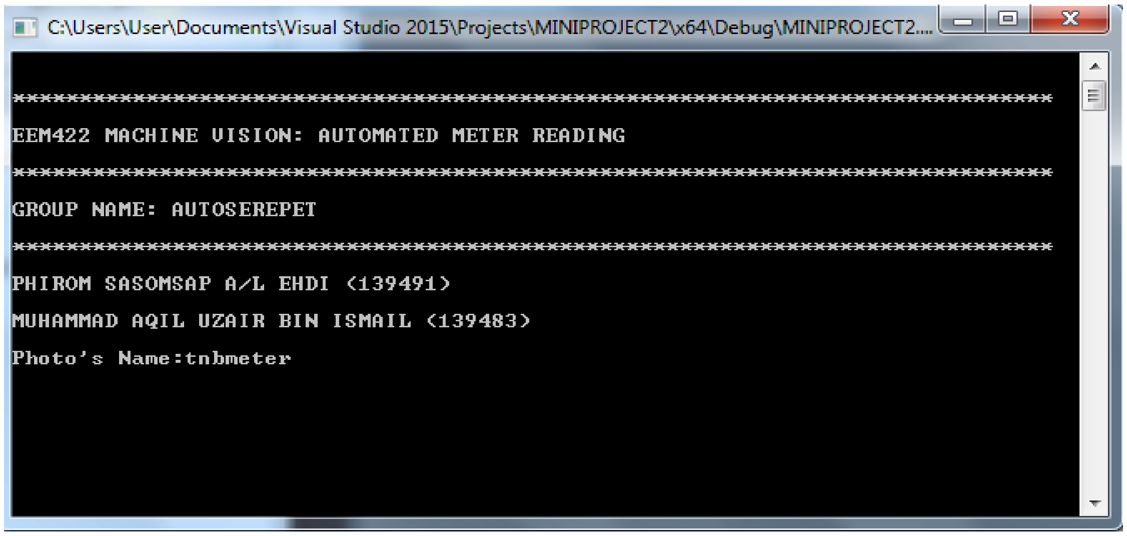


Figure 6. The image when the brigtness and contrast has adjusted.

After the ESC key is pressed, the command prompt is displayed as below. From here, the user need to write the name of the photo to detect the number inside the TNB Meter. The name of photo that we have implemented is "tnbmeter". Then press Enter.

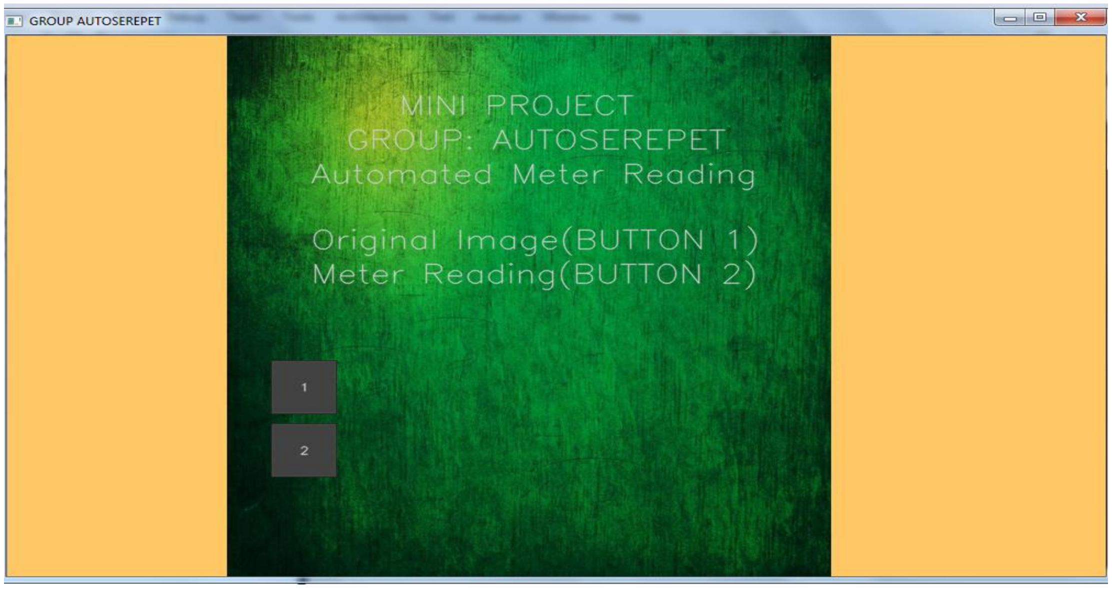


```
C:\Users\User\Documents\Visual Studio 2015\Projects\MINIPROJECT2\x64\Debug\MINIPROJECT2....

*****************************************************************************
EEM422 MACHINE VISION: AUTOMATED METER READING
*****************************************************************************
GROUP NAME: AUTOSEREPET
*****************************************************************************
PHIROM SASOMSAP A/L EHDI (139491)
MUHAMMAD AQIL UZAIR BIN ISMAIL (139483)
Photo's Name:tnbmeter
```

Figure 7. Command Prompt for key in the photo name.

Now, the template with two button is displayed on the screen. The button is for display the original of an image and button 2 for crop and detect the number of an image .



GROUP AUTOSEREPET

MINI PROJECT
GROUP: AUTOSEREPET
Automated Meter Reading

Original Image(BUTTON 1)
Meter Reading(BUTTON 2)

Figure 8: Template with two button (GUI system)

After the button 1 and 2 in pressed, the original image is displayed and from the original image, the system will crop and detect the number of TNB Meter.

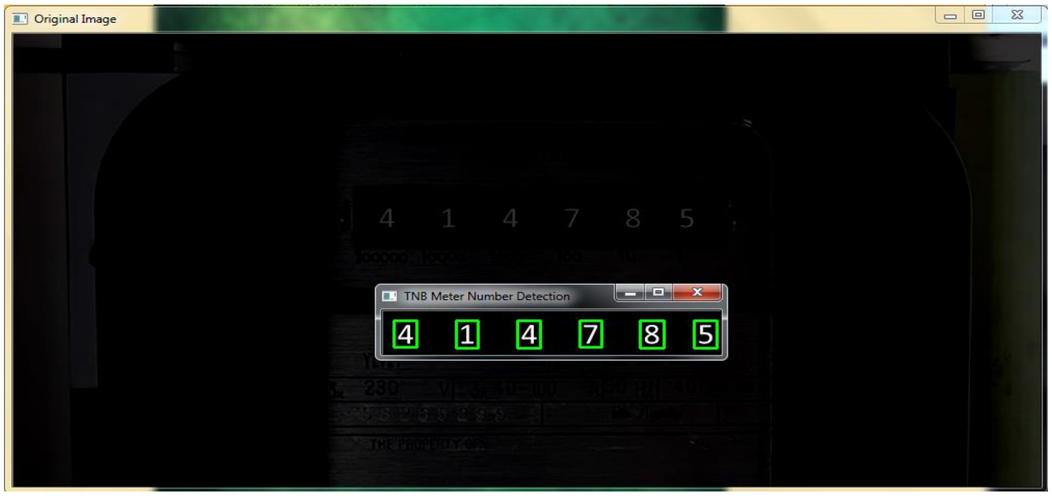
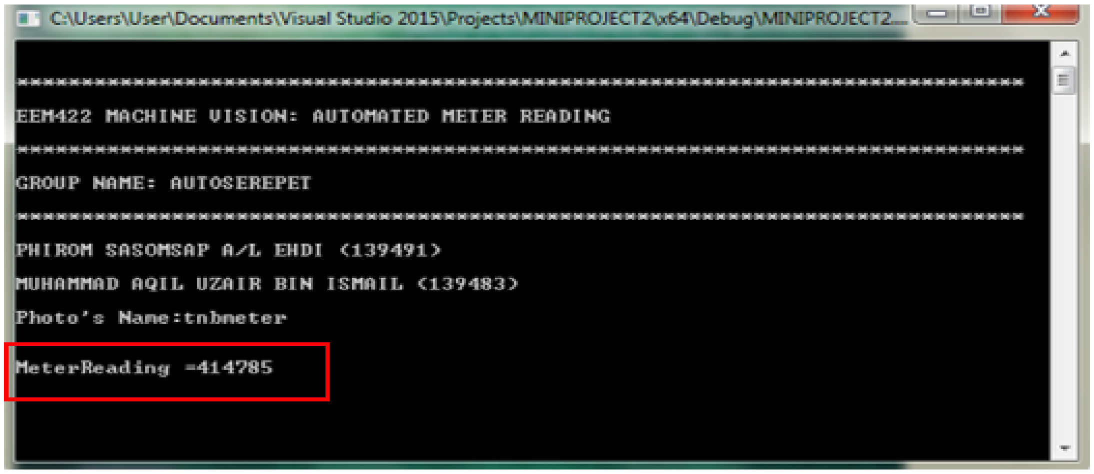


Figure 9: The system crop & detect the number from original image.

When the system crop and detect the number from TNB Meter, the results will display in command prompt as shown as below



```
C:\Users\User\Documents\Visual Studio 2015\Projects\MINIPROJECT2\x64\Debug\MINIPROJECT2...
*****************************************************************************
EEM422 MACHINE VISION: AUTOMATED METER READING
*****************************************************************************
GROUP NAME: AUTOSEREPET
*****************************************************************************
PHIROM SASOMSAP A/L EHDI (139491)
MUHAMMAD AQIL UZAIR BIN ISMAIL (139483)
Photo's Name:tnbmeter

MeterReading =414785
```

Figure 7: The exact value of meter reading in command prompt.

6. FUTURE SUGGESTION

The system still has many room for improvement. Firstly, the system can be linked to the meter to get real time reading. Currently, the system received image captured by customers that have been uploaded to the server. By link the system to the meter, customers don't need to capture meter image anymore instead the system can be scheduled so when the time come the system will automatically calculate the bill according to the real time reading. The current automated meter reading only can read the meter reading and display it to the user and then the user need to calculate the usage manually. Also the system can be improved to read multiple image at one time as currently in can only read one image at a time. In addition, the system can be improved by link it to application on smartphone so customers will get notification about their usage on smartphone. The problems with paper bill is that some customers forgot where they placed their bill.

7. CONCLUSION

In conclusion, an automated meter reading system has been developed to overcome the problem of reading TNB meter and water meter directly due to the pandemic. The system consist of two main part that are capturing image and interpreting data from the image. Image capturing are done by using any camera that can show the full meter reading and at acceptable brightness level. Then computer will interpret data from the image using the system developed. The system was developed using OpenCV software and C++ programming. To develop the system, template matching, edge detection, greyscale conversion have been applied. To make the system more user friendly, simple GUI has been designed which contains additional functions. Finally, the automated meter reading system has been tested using different image condition it able to show the correct number on the image.

8. REFERENCES

1. Define CVUI_IMPLEMENTATION before the inclusion of cvui.h:

Sources: *https://github.com/Dovyski/cvui/blob/master/cvui.h*

2. OpenCV KNN Recognition Character:

Source:https://www.youtube.com/watch?v=CK0OCeCN9zg&t=22s&ab_channel=ChrisDahms

3. Brightness and Contrast Trackbar:

Source: *https://www.opencv-srf.com/2011/11/track-bars.html*

4.OpenCV C++ API:

Source: *https://www.opencv-srf.com/2017/11/opencv-cpp-api.html*

9. APPENDIX

```
#include<opencv2/core/core.hpp>
#include<opencv2/highgui/highgui.hpp>
#include<opencv2/imgproc/imgproc.hpp>
#include<opencv2/ml/ml.hpp>
#include "opencv2/imgcodecs.hpp"
#include "opencv2/photo.hpp"
#include<iostream>
#include<sstream>
#define CVUI_IMPLEMENTATION
#include "cvui.h"
#define WINDOW_NAME "GROUP AUTOSEREPET"
#define WINDOW1_NAME "original image"

using namespace std;
using namespace cv;

// global variables //
const int MIN_CONTOUR_AREA = 100;
const int RESIZED_IMAGE_WIDTH = 20;
const int RESIZED_IMAGE_HEIGHT = 30;

//////////////////////////
class ContourWithData {
public:
	// member variables //
	std::vector<cv::Point> ptContour;           // contour
	cv::Rect boundingRect;                      // bounding rect for contour
	float fltArea;                              // area of contour

/////////////////////////
	bool checkIfContourIsValid() {                              // obviously in a
production grade program
		if (fltArea < MIN_CONTOUR_AREA) return false;           // we would have a
much more robust function for
		return true;                                            // identifying if a
contour is valid !!
	}

////////////////////////
	static bool sortByBoundingRectXPosition(const ContourWithData& cwdLeft, const
ContourWithData& cwdRight) {      // this function allows us to sort
		return(cwdLeft.boundingRect.x < cwdRight.boundingRect.x);
// the contours from left to right
	}

};

//////////////////////////////////////////
int main(int argc, char** argv) {
	Mat src = imread("tnbmeter.jpg", 1);
	//if fail to read the image
	if (!src.data)
	{
		cout << "Error loading the image" << endl;
		return -1;
	}
	// Create a window
	namedWindow("TNB Meter", 2);

	int iSliderValue1 = 50;
	createTrackbar("Brightness", "TNB Meter", &iSliderValue1, 200);
	int iSliderValue2 = 50;
	createTrackbar("Contrast", "TNB Meter", &iSliderValue2, 200);
	while (true)
	{
		//Change the brightness and contrast of the image
		Mat dst;
		int iBrightness = iSliderValue1 - 50;
		double dContrast = iSliderValue2 / 50.0;
		src.convertTo(dst, -1, dContrast, iBrightness);

		//show the brightness and contrast adjusted image
		imshow("TNB Meter", dst);

		// Wait until user press some key for 50ms
		int iKey = waitKey(50);

		//if user press 'ESC' key
		if (iKey == 27)
		{
			break;
		}
	}

	std::vector<ContourWithData> allContoursWithData;           // declare empty
vectors,
	std::vector<ContourWithData> validContoursWithData;         // we will fill these
shortly


			// read in training classifications ///////////////

	cv::Mat matClassificationInts;      // we will read the classification numbers
into this variable as though it is a vector

	cv::FileStorage fsClassifications("classifications.xml", cv::FileStorage::READ);
// open the classifications file

	if (fsClassifications.isOpened() == false) {
// if the file was not opened successfully
		std::cout << "error, unable to open training classifications file, exiting
program\n\n";    // show error message
		return(0);
// and exit program
	}

	fsClassifications["classifications"] >> matClassificationInts;      // read
classifications section into Mat classifications variable
	fsClassifications.release();                                        // close the
classifications file


					// read in training images
///////////////////////////////////////////
```

```
	cv::Mat matTrainingImagesAsFlattenedFloats;		// we will read multiple 
images into this single image variable as though it is a vector

	cv::FileStorage fsTrainingImages("images.xml", cv::FileStorage::READ);		// 
open the training images file

	if (fsTrainingImages.isOpened() == false) {						
// if the file was not opened successfully
		std::cout << "error, unable to open training images file, exiting 
program\n\n";		// show error message
		return(0);									
// and exit program
	}

	fsTrainingImages["images"] >> matTrainingImagesAsFlattenedFloats;		// 
read images section into Mat training images variable
	fsTrainingImages.release();							// 
close the traning images file


						// train 
//////////////////////////////////////////////////////

	cv::Ptr<cv::ml::KNearest>  kNearest(cv::ml::KNearest::create());		// 
instantiate the KNN object


	kNearest->train(matTrainingImagesAsFlattenedFloats, cv::ml::ROW_SAMPLE, 
matClassificationInts);

	// test 
///////////////////////////////////////////////////////////////////////////////

	cv::Mat matTestingNumbers = cv::imread("romread4.jpg");		// read in the 
test numbers image

	if (matTestingNumbers.empty()) {					// if unable to 
open image
		std::cout << "Cannot Detect Meter Reading";		// show error message 
on command line
		return(0);							// and exit 
program
	}

	cv::Mat matGrayscale;		//
	cv::Mat matBlurred;		// declare more image variables
	cv::Mat matThresh;		//
	cv::Mat matThreshCopy;		//

	cv::cvtColor(matTestingNumbers, matGrayscale, CV_BGR2GRAY);		// convert to 
grayscale


				// blur
	cv::GaussianBlur(matGrayscale,		// input image
		matBlurred,			// output image
		cv::Size(5, 5),			// smoothing window width and height in pixels
		0);				// sigma value, determines how much the image 
will be blurred, zero makes function choose the sigma value
```

```
                                                          // filter image from grayscale
to black and white
       cv::adaptiveThreshold(matBlurred,                               // input image
              matThresh,                            // output image
              255,                                  // make pixels that pass the
threshold full white
              cv::ADAPTIVE_THRESH_GAUSSIAN_C,       // use gaussian rather than mean,
seems to give better results
              cv::THRESH_BINARY_INV,                // invert so foreground will be
white, background will be black
              11,                                   // size of a pixel neighborhood used
to calculate threshold value
              2);                                   // constant subtracted from the mean
or weighted mean

       matThreshCopy = matThresh.clone();              // make a copy of the thresh
image, this in necessary b/c findContours modifies the image

       std::vector<std::vector<cv::Point> > ptContours;        // declare a vector for
the contours
       std::vector<cv::Vec4i> v4iHierarchy;                    // declare a vector for
the hierarchy (we won't use this in this program but this may be helpful for reference)

       cv::findContours(matThreshCopy,             // input image, make sure to use a
copy since the function will modify this image in the course of finding contours
              ptContours,                             // output contours
              v4iHierarchy,                           // output hierarchy
              cv::RETR_EXTERNAL,                      // retrieve the outermost contours
only
              cv::CHAIN_APPROX_SIMPLE);               // compress horizontal, vertical,
and diagonal segments and leave only their end points

       for (int i = 0; i < ptContours.size(); i++) {               // for each contour
              ContourWithData contourWithData;
// instantiate a contour with data object
              contourWithData.ptContour = ptContours[i];
// assign contour to contour with data
              contourWithData.boundingRect = cv::boundingRect(contourWithData.ptContour);
// get the bounding rect
              contourWithData.fltArea = cv::contourArea(contourWithData.ptContour);
// calculate the contour area
              allContoursWithData.push_back(contourWithData);
// add contour with data object to list of all contours with data
       }

       for (int i = 0; i < allContoursWithData.size(); i++) {                     // for
all contours
              if (allContoursWithData[i].checkIfContourIsValid()) {                  //
check if valid
                     validContoursWithData.push_back(allContoursWithData[i]);
// if so, append to valid contour list
              }
       }
       // sort contours from left to right
       std::sort(validContoursWithData.begin(), validContoursWithData.end(),
ContourWithData::sortByBoundingRectXPosition);
```

```
	std::string strFinalString;         // declare final string, this will have the
final number sequence by the end of the program

	for (int i = 0; i < validContoursWithData.size(); i++) {            // for each
contour



		                   // draw a green rect around the current char
		cv::rectangle(matTestingNumbers,                             // draw
rectangle on original image
			validContoursWithData[i].boundingRect,        // rect to draw
			cv::Scalar(0, 255, 0),                        // green
			2);                                           // thickness

		cv::Mat matROI = matThresh(validContoursWithData[i].boundingRect);
// get ROI image of bounding rect

		cv::Mat matROIResized;
		cv::resize(matROI, matROIResized, cv::Size(RESIZED_IMAGE_WIDTH,
RESIZED_IMAGE_HEIGHT));     // resize image, this will be more consistent for recognition
and storage

		cv::Mat matROIFloat;
		matROIResized.convertTo(matROIFloat, CV_32FC1);             // convert Mat
to float, necessary for call to find_nearest

		cv::Mat matROIFlattenedFloat = matROIFloat.reshape(1, 1);

		cv::Mat matCurrentChar(0, 0, CV_32F);

		kNearest->findNearest(matROIFlattenedFloat, 1, matCurrentChar);     //
finally we can call find_nearest !!!

		float fltCurrentChar = (float)matCurrentChar.at<float>(0, 0);

		strFinalString = strFinalString + char(int(fltCurrentChar));        //
append current char to full string
	}

	cout << "\n\n" <<
"*******************************************************************************";
	cout << "\n\n" << "EEM422 MACHINE VISION: AUTOMATED METER READING ";
	cout << "\n\n" <<
"*******************************************************************************";
	cout << "\n\n" << "GROUP NAME: AUTOSEREPET ";
	cout << "\n\n" <<
"*******************************************************************************";
	cout << "\n\n" << "PHIROM SASOMSAP A/L EHDI (139491)";
	cout << "\n\n" << "MUHAMMAD AQIL UZAIR BIN ISMAIL (139483)";
	cout << "\n\n" << "Photo's Name:";



char tnbmeter;
	cin >> tnbmeter;
	cout << "\n\n" << "MeterReading =" << strFinalString << "\n\n";       // show the
full string
```

```
	//Push button
	Mat mainImage = imread("green.jpg", 1);
	resize(mainImage, mainImage, Size(550, 600), 0, 0, INTER_AREA);`
	Mat frame = Mat(Size(1000, 600), CV_8UC3);
	cvui::init(WINDOW_NAME);

	while (true)
	{
		frame = Scalar(100, 200, 300);
		cvui::image(frame, 200, 0, mainImage);
		cvui::beginColumn(frame, 20, 15, 20, 30);
		cvui::space(50);
		cvui::text("                     MINI PROJECT",1.0);
		cvui::space(15);
		cvui::text("                  GROUP: AUTOSEREPET ", 1.0);
		cvui::space(15);
		cvui::text("                Automated Meter Reading", 1.0);
		cvui::space(50);
		cvui::text("                Original Image(BUTTON 1)", 1.0);
		cvui::space(15);
		cvui::text("                Meter Reading(BUTTON 2)", 1.0);
		cvui::endColumn();

		cvui::beginColumn(frame, 240, 360, 100, 200);
		if (cvui::button(60, 60, "1"))
		{
			imshow("Original Image", src);
		}
		cvui::space(10);
		if (cvui::button(60, 60, "2"))
		{
			imshow("Meter Reading", matTestingNumbers);   // this line show you
the line detected image as output
		}

		cvui::endColumn();

		cvui::imshow(WINDOW_NAME, frame);
		if (cv::waitKey(20) == 27)
		{
			break;
		}

	}

	waitKey(0);
	return(0);


}
```