International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 09 Issue: 11 | Nov 2022 www.irjet.net p-ISSN: 2395-0072

Real Time & Automated Meter Reading using Image Processing Reducing Human Error

Rahul Vemuri[1], Viraj Sapte[2], Jim Cheriyan[3], Rishabh Maniyar[4], Dr. Abhay Kshirsagar[5]

[1,2,3,4]Student, Dept. of Electronics Engineering, Vivekanand Education Society's Institute of Technology, Mumbai, Maharashtra, India
[5]Associate Professor, Dept. of Electronics Engineering, Vivekanand Education Society's Institute of Technology, Mumbai, Maharashtra, India

---

Abstract - *Utilities (Gas, and Electricity) plays a pivotal role in our everyday life. To keep track of energy consumption meters are used. Earlier Analog meters were used which posed some problems like the parallax error, the needles getting stuck, drifting out of calibration, susceptible to shock and vibrations. Eventually, with the replacement of analog meters with digital meters, these problems were solved. However, a common disadvantage of analogue meters and digital meters was the need for workers to be physically present to record readings from the meter and subsequently relay those values to the utility provider firm. This conventional method is susceptible to errors and the users have to pay more than what they actually had to pay.*

The focus is to solve this problem with a system that automatically captures the image of the meter and sends it to the server wherein image processing is done to obtain the meter readings. Another advantage of the system is the reduction of human resources and expenses for recording meter readings physically.

The setup will include a webcam that will capture images at predetermined intervals. Raspberry Pi Zero W is used to send these clicked images to the Firebase Realtime Database. This image is then retrieved to extract the data regarding the utility consumption by using a Python library called EasyOCR. To increase the efficiency of extracting the correct data from the snapshot of an energy meter, software tools such as Google Vision API, Amazon Rekognition, and Node-RED are used.

Key Words: **Analog Meters, Raspberry Pi Zero W, EasyOCR, Node-RED, Amazon Rekognition**

1. INTRODUCTION

Utilities (Gas, and Electricity) are essential services that play a major role in our lives. With the rise in productivity and modernization, the usage of these utilities is increasing every day. In the case of electricity billings, consumers often encounter issues with defective bills, which then become their sole problem, requiring them to lookout for a resolution from the MSEB. Human intervention in collecting the readings of the meter, is the only explanation for that issue since, the utility provider company, through their workforce, checks the numbers shown in the electric meter panel periodically, usually every month. This procedure that involves noting down the readings of the meter, is done manually. The operator captures an image of the energy meter and the readings for the same are recorded. The operator reports the findings back to the utility provider and the provider will calculate how much each customer will be charged. The entire procedure takes a long time and is vulnerable to human error. Moreover, it is not feasible to perform this manual process for the utility meters set up in rural areas Due to the increased population and industrialization in countries such as India, a better method for real time and expedient meter reading is proposed in this paper.

1.1 LITERATURE REVIEW

In previous works, many models were designed using different image processing techniques and algorithms. In [1] the author proposes a system that takes automatic meter readings with the aid of a sensor. A camera is mounted in front of the house's energy meter to capture photographs. Contour algorithm is used with a processor to separate digits and measure the bill for the month. The Raspberry Pi is used in this paper because it is a kind of minicomputer. It is not possible to run Microsoft Windows on it because it has a different processor, hence it is recommended to use Linux operating systems that are very similar to Windows. Raspberry Pi is used to surf the internet and send an email about the electricity consumption of the current month. The bill is then wirelessly transmitted to the server via GSM and shown on an LCD for the user's reference. According to the concept of AMR, it allows for easy savings by meter reading, increased data accuracy, faster billing, and better customer service. The camera is placed in front of the E-meter. The camera takes a picture when the order is sent. The bill is calculated and sent using GSM using this image, which is processed by the Raspberry Pi using the contour algorithm.

Another attempt [2] is made in this work, which offers a system that employs mobile phones to take a picture of the power meter. Image processing functions are then implemented to automatically identify the meter reading's digits. Preprocessing, Digit Segmentation, and Digit

Recognition are the three key processes that this image goes through. In the Preprocessing Stage, the RGB image is converted to a grayscale image, which is followed by Image Binarization. A noise reduction algorithm is used to negate the noise produced in the image. Later, the meter reading's area in the binarized image is cropped. Next comes the second stage of Data Segmentation that aims to create six segmented digits by scanning the clipped numeric region left to right, vertically and horizontally. In order to make the recognition stage easier, all segmented digits are finally resized to make them all roughly the same size. Next is the stage of Digit Recognition in which one segmented sample of each digit is used to extract the features of that digit and is stored as a template. Hence every digit will have its unique template. These templates are then used to identify the digits from the segmented images by matching the features. This system recognizes the digits of the electric meter reading in three major stages of image processing with a sufficient level of accuracy of 96.49% for each digit and an overall reading accuracy of 85.71%.

The authors in [3] propose a system wherein a timer is associated with a webcam. The timer instructs the camera to click the image of the meter at regular intervals of time. The captured image is then converted to a binary image, followed by adjustments by altering the contrast and brightness. The meter reading area is cropped by adjusting the trim height and width to obtain the numeric area. The Support Vector Machine learning algorithm is applied to the pre-processed image to detect and segment the digits of the meter reading. The Support Vector Machine is then reapplied to each of the segmented images in order to distinguish digits from 0 to 9. Finally, the output and other information, including the name and number of the customer, the date and time, are also sent to the server. When a certain amount of time has passed without the server receiving the meter reading, the server thinks that the camera is broken and dispatches service personnel to replace the damaged camera.

An Automatic Meter Reading (AMR) system prototype is described in the research work in [4]. Three components make up the prototype design: an interface circuit, a traditional or electromechanical meter, and ZigBee modules. The design of a cost-effective interface circuit and the programming of ZigBee modules are the most important tasks. The modules are designed to behave like End-Devices, Routers, and Coordinators in the ZigBee wireless network and Concentrators and Range Extenders. According to its characteristics, the meter reading system has a distributed structure. A data collector, wireless communication networks, and a server system make up the system. Data from digital meters is sent through two boards: a data collection module and a ZigBee transmission module. This is where the ZigBee true System-on-Chip CC2430 makes a big difference. It combines the excellent performance of the market-leading CC2420 RF transceiver with a 32/64/128 KB flash memory, 8 KB RAM, an industry-standard upgraded 8051 MCU, and a plethora of additional practical features. It produces very little noise and consumes very little fuel. The camera takes a picture of the meter reading and transmits it over Zig-Bee to the server PC. There, the image is identified and segmented, and the digits are read to be used in creating bills. The method used in this study processed images using the following steps: After converting the image to grayscale, the numeric area is cropped, projected, and then binarized using a threshold. Segmentation and recognition are then carried out using a digital recognition algorithm.

In [5] the exploration work is tied in with perusing the upsides of the electrical meter by utilizing a web camera that snaps a picture of the meter, perceives the digits, and afterward stores the yield in a content document. A web camera of 325x288 is connected to a laptop. In front of the meter, the camera is mounted. The camera will take an image of the electricity meter in a sequence of time intervals. Then, it will be transformed using the gray-scale conversion method into a grayscale image. The fundamental idea behind this conversion is to preserve brightness while removing the color and saturation information. The ideal way to change RGB to grayscale is shown in the following equation.

Luminance = 0.299 * Red + 0.587 * Green + 0.114 * Blue

The previous stage's gray-scale image is transformed into a binary image (Black & White). This conversion is the most crucial step in all phases of the EMRR method, and more specifically in the plate extraction process to resolve illumination issues. In the next stage, the image is trimmed in such a way that only the meter reading region will be extracted to identify the digits in the next stage. Then, use the Vertical Edge Detection Algorithm (VEDA) to segment the image and record the outcome in an array matrix after searching the image horizontally till you locate a white pixel. Finally, compare each area of the image to the templates, and then record the results as the meter reading in a text file.

In [6] a system is proposed wherein at specific intervals the image is captured by a video camera. After imaging, the software can save the image to the file system, where it can be read later. Since the camera does not have to be connected all of the time, this is very useful for production and testing. After that, the picture is pre-processed. This step should adjust and refine the image so that the individual digits of the counter can be detected and isolated in the next step. The extracted digit images are then fed into the character recognition system (OCR). It must, however, be interactively trained with a selection of all possible characters (the digits 0 through 9). This results in a file containing a collection of training data. The OCR loads this training data during normal activity and can thus identify an unknown character. This classification comes with a degree of error. It makes sense to check the readings for plausibility

before proceeding with the processing. If it passes, the observed counter value is saved in a database along with the current time. Later evaluations, such as the generation of graphics with hourly, regular, and weekly power usage, can be made using the data stored here.

1.1 PROPOSED SYSTEM

The proposed system consists of a Webcam which is placed in the sight of the utility meter, as shown in Fig-1. The Webcam is connected to Raspberry Pi Zero W which is powered by a 5 V power bank. At specific intervals, the webcam captures the image of the meter which is subsequently sent to Firebase Realtime Database with the help of Raspberry Pi Zero W. The image is then retrieved from the Database and image processing methods are applied.

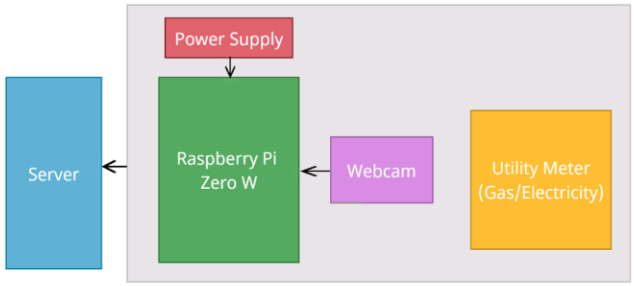

Fig -1: System Block Diagram

Fig -2-: Raspberry Pi Setup

Step 1: Setting up Raspberry Pi Zero W with integrated camera module as shown in Fig 2. To begin with, install 'Raspberry Pi Imager' software on the Laptop/PC. Take a 32/64 GB microSD card and using an adapter/card reader, connect it to a Laptop/PC. Run Raspberry Pi Imager and format the microSD card (make sure that the microSD card is selected using the 'Choose Storage' option). From the 'Choose OS' option select 'Raspberry Pi OS with desktop'. Select the 'Write' option. Insert the microSD card into Raspberry Pi Zero W. Connect Raspberry Pi Zero W to a monitor using mini-HDMI to HDMI cable. Connect the OTG cable to micro-USB port. Connect mouse, keyboard to OTG cable. Power the Raspberry Pi Zero W. Connect it to Wi-Fi. Open the 'Terminal'. Run update and upgrade commands to download the package lists from the repositories and update to get information on the newest versions of packages and their dependencies.

Step2: Taking image at regular intervals with automated python code.

Step 3: Pushing the image to server. Connect USB Webcam to Raspberry Pi Zero W and run 'fswebcam image_name.jpg' to capture the image. Then execute the python code to push the clicked image to Firebase database so that it can be retrieved for further processing

Step 4: Retrieving image from server.

The image from firebase is retrieved using python code implementing the pyrebase library. Pyrebase is an interface between Python and Firebase's REST API. It allows us to store and retrieve data, images, etc from firebase. Before retrieving the image, we have to establish a connection between our python code and Firebase which is done by including the configuration dictionary or JSON. We initialize this configuration using the function initialize_app(). Images in Firebase are stored in a specific section called Storage. So, to access it we call the storage() function followed by a dot operator variable that is initialized with the initialize_app function.

This storage function is further used to retrieve the image using the child function attached to it using a dot operator with the specific name of the image stored in Firebase as a parameter. Finally, the image is downloaded by using the download function having attributes filename initialized with the name by which it has to be stored and another attribute as path initialized with pathname.

Step 5: Applying image processing at the server end.

Using EasyOCR

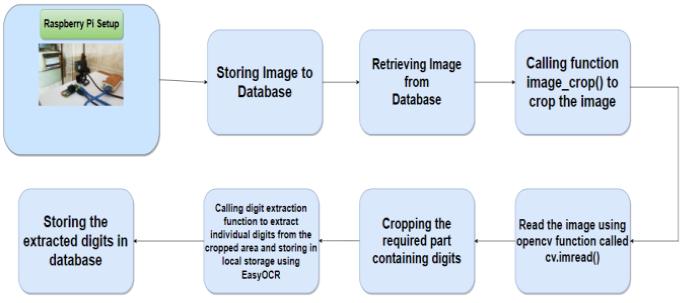

Fig -3: Work Flow of the system using EasyOCR

The image from firebase is retrieved using the pyrebase library and is stored in the local storage of our system. The necessary part of the image is the portion containing the digits. So that job is accomplished by a user-defined function called **image_crop()**. The image currently resides in the local storage of the system. To use that stored image we are using a comprehensive image processing library called **OpenCV**. Opencv has an image reading function called imread() which is used to process the image as shown in Figure 4.2.2. The arguments provided to the imread() function is the path to the image in the local system **imread()** function reads the image in the form of a matrix. The next part involves cropping the necessary portion of the image which is manually done considering the dimensions required. After the necessary portion is extracted, the further process involves segregation of each digit in the image which is done by calling another user-defined function called **digit_extraction()**. The digit extraction function extracts each digit and stores it in the local storage. Each digit is recognized by EasyOCR using easy. Reader(['en']) function where en signifies the English language. The reader function is used in collaboration with the readtext() function where image each individual image is sent as an argument with an additional argument called allowlist which restricts the recognition between 0 to 9. The recognized digits are displayed to the user.

Using Amazon Rekognition

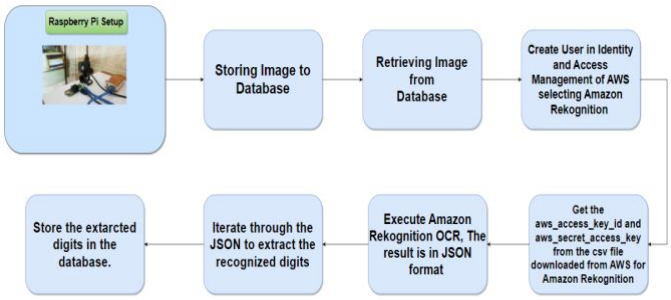

Fig -4: Work Flow of the system using Amazon Rekognition

The image which is stored in Firebase is retrieved using the pyrebase library which is a dedicated python library to store and retrieve images from Firebase. The retrieved image is stored in the local storage of the system. After storing the image, we have to get the AWS credentials to use the text detection module of Amazon Rekognition. We acquire these credentials in CSV format from the Identity and Management Access Section of Amazon Web Services where we opt for Amazon Rekognition. We read the CSV file in python to get the aws_access_key_id and aws_access_secret_key to authenticate Amazon Rekognition. Amazon Rekognition API is accessed in python using the boto3 library. Boto3 is the Python SDK for AWS. Boto3 allows us to access AWS resources from python scripts. It can create, update and delete AWS resources. We use the client function along with the boto3 library to authenticate rekognition. The image stored in the local system is read in binary format which is provided as a dictionary argument to detect text function of Amazon Rekognition. The response is received in a JSON format. The digits are retrieved iterating through the TextDetections key in the JSON. We can customize the iteration as per the JSON received and the number of digits to be recognized. Figure 4 gives us a brief idea of the process explained above.

Using Google Vision API

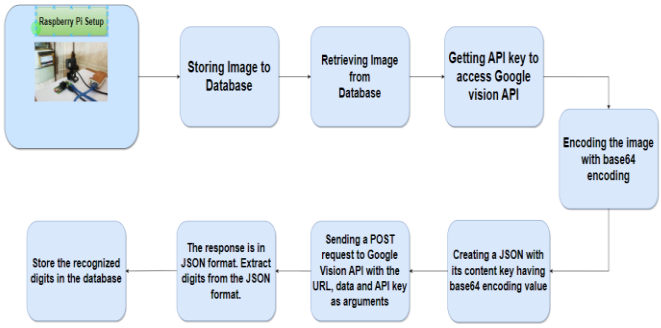

Fig -5: Work Flow of the system using Google Vision API

The image which is stored in Firebase is retrieved using pyrebase library which is a dedicated python library to store and retrieve images from Firebase. The retrieved image is stored in the local storage of the system. To authenticate the Google Vision API, we are using the API key which is downloaded from Google Developers API. The path of the stored image has to be defined before sending a request to Google Vision API. This path is provided to a user-defined function called image_request. image_request() function creates a JSON file with content key having base64 encoding value. Base64 is a group of binaries to text encoding schemes that represent binary data in ASCII string format The base64 encoding is implemented by calling a user-defined function encode_image. encode_image() function reads the image file in binary format and returns base64 encoding using the base64 library used in collaboration with the b64encode function. The JSON file, URL of Google Vision API ("https://vision.googleapis.com/v1/images:annotate"), and API key are given as arguments to the request function which is an inbuilt function of the request library in python. A POST request is rendered to the URL and the response is in JSON format. The digits recognized are then fetched by iterating through the JSON received. Figure 5 is the workflow of the above-explained events

1.2 PROPOSED SYSTEM

A programming tool called Node-RED can be used to establish novel and intriguing links between physical components, APIs, and web services. The several nodes in the palette can be easily connected using the flow editor that comes with Node-RED. The flow can then be quickly and simply delivered to the runtime. JavaScript functions can be created inside of a rich text editor. You can save and reuse helpful features, models, and flows thanks to a built-in library.

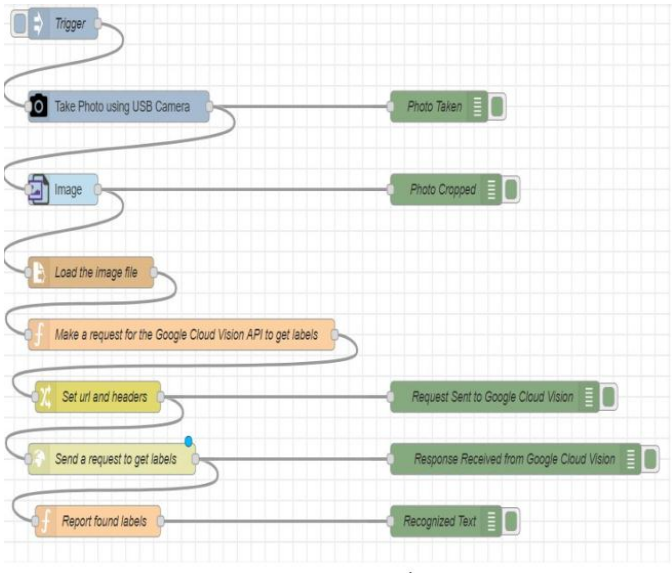

Fig -6-: Node-Red workflow

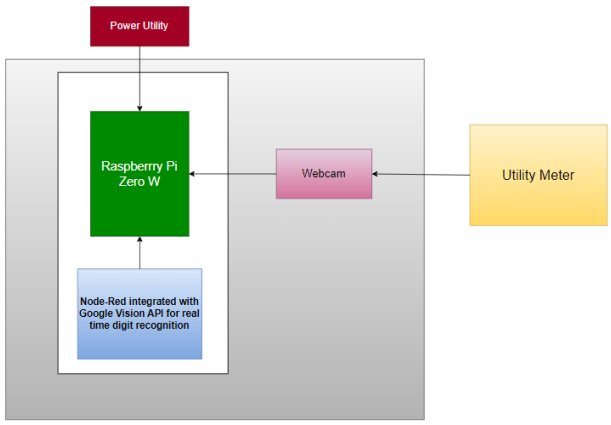

Fig -7-: System Block Diagram 2.0

The Fig 6 shows the workflow of NodeRed. NodeRed is a contingent to the proposed system as shown in Fig 1. The use of NodeRed eliminates the need of server. It does image processing at real time when the trigger is initiated as shown in Fig 7.

Open 'Terminal' and run the 'node'-red-start' command. Open any browser and enter the IP address of Raspberry Pi followed by the port address of Node-RED which is '1880', separated by a colon (:). It can be expressed as 'http://<ip-address>:1880'. Install node-red-contrib-usbcamera and node-red-contrib-image-simple-node from Manage Palette in Node-RED's web interface. Drag and drop the nodes from the node window which is on the left side of the screen and create a flow as shown in Fig 3.

Trigger (Inject Node) is used for manual/automatic triggering of the process. A timer can be used to trigger the process in certain time intervals. The camera node captures the image using a USB Webcam. It stores the image in the local storage of the Raspberry Pi where the path is to be mentioned in the config menu of the node. Image node crops the image according to the coordinates defined which results to give the reading region of the meter. This cropped image is also stored in local storage. The file node loads the cropped image and an HTTP request is sent to the Google Vision API to give the readings using the function node. Google vision API key is set in the change node. To display the meter reading, a function node is used. The final meter reading is displayed in the debug sidebar which is present on the right side of the screen. To display messages in the debug sidebar, Debug nodes (Green colored nodes) are used.

2. EXPERIMENTAL RESULTS

Fig 8 shows the cropped image of the Gas meter recognized by EasyOCR. EasyOCR successfully recognizes all the digits displayed in the adjacent side of the Fig 8. Fig 9 is the experimental result on a blurred image. EasyOCR fails to recognize all the digits successfully for blurred image which is one of the limitations of EasyOCR. The recognized digits are displayed on the right in Fig 9.

Fig -8: Image Processing using EasyOCR

Fig -9: Image Processing using EasyOCR for blurred meter image

Input meter image as shown in Fig 10 is rendered as a request to Amazon Rekognition API, The API successfully recognizes all the digits as shown in Figure 10. As far as the Gas meter image is concerned as shown on the right side of Fig 11 when rendered as a request to Amazon Rekognition

API fails to recognize all digits. It is successful in recognizing only the first six digits out of 8 digits as shown in Fig 11.

Fig -10: Image Processing using Amazon Rekognition

Fig -11: Image Processing using Amazon Rekognition

Google Vision API is successful in recognizing a wide range of utility meters outperforming other OCRs discussed above. The Gas meter image is shown in Fig 10 is given as a JSON request to the Google Vision API which sends a JSON response that gets all the digits accurately. Fig 10 shows the digits recognized by Google Vision API for the Gas meter and meter image respectively. The performance of Google Vision API for a wide range of meters was better than EasyOCR and Amazon Rekognition. It was successful in giving precise and convincing results.

Fig -12: Image Processing using Google Vision API

Node-RED with Raspberry Pi integration with Google Vision API as an OCR for digit recognition was successfully executed for real-time meter reading. Fig 13 is the image captured when we trigger the process flow of Node-Red as shown in Fig 6. Fig 13 shows the cropped portion of the reading part and the output reading that is displayed in the debug sidebar. Node-Red with Raspberry Pi using Google Vision API gives real time and expedient results with satisfactory accuracy. It also reduces the delay and saves storage space required for storing images at the server end.

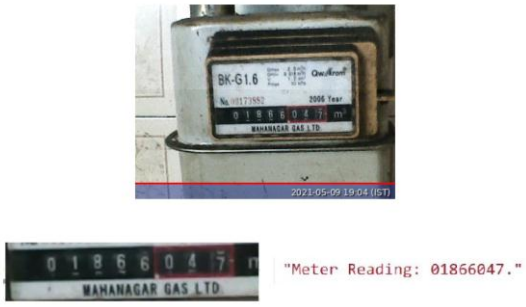

Fig -13: Image Processing using Node-Red with Raspberry Pi incorporating Google Vision API

3. CONCLUSIONS

Utility meters play a substantial role in our lives. Accurate meter readings are of paramount importance. Manual submission of these readings may cause errors. To avoid such errors our proposed system is implemented. Our system prevents any human intervention. We were successful in achieving the desired goal using Raspberry Pi Zero W integrated with Webcam in association with Node-Red incorporating Google Vision API. Google Vision API stands out as the most efficient OCR recognition with impeccable accuracy for varied utility meters.

4. FUTURE SCOPE

The future scope of the paper includes enhancing security compliances. Creating an application which integrates the hardware implementation and software implementation displaying the data at real time is the primary futuristic goal of this project. Enhancing Data Visualization models for better contemplation of data. Advanced Deep Learning models to predict futuristic data to prevent malpractices.

ACKNOWLEDGEMENT

We would like to thank our project guide **Prof. Dr. Abhay Kshirsagar**. This project work would not have been possible without the encouragement and the able guidance of our guide. His enthusiasm and optimism made this experience both rewarding and enjoyable. We would also like to thank

our Head of the Department of Electronics Department, Prof. (Mrs.) Kavita Tewari. I would also like to acknowledge the guidance and support of our Project Coordinator **Mr. Anurag Chugh**. His feedback and editorial comments were also invaluable. We would like to extend sincere thanks to Principal Dr. (Mrs.) J.M. Nair, V.E.S.I.T. for her invaluable support. We would like to express our sincere thanks towards the entire staff of the Electronics Department for their cooperation in the completion of our project.

REFERENCES

[1] Sneha Salunkhe, Dr.(Mrs.) S. S. Lokhande, " Raspberry Pi based automatic meter reading ", International Journal of Application or Innovation in Engineering & Management (IJAIEM), Volume 5, Issue 9, September 2016, pp. 069-073, ISSN 2319 - 4847.

[2] A. Elrefaei, A. Bajaber, S. Natheir, N. AbuSanab, and M. Bazi, "Automatic electricity meter reading based on image processing," *2015 IEEE Jordan Conference on Applied Electrical Engineering and Computing Technologies (AEECT)*, Amman, Jordan, 2015, pp. 1-5, DOI: 10.1109/AEECT.2015.7360571.

[3] C. Edward, Support Vector Machine Based Automatic Electric Meter Reading System, IEEE International Conference on Computational Intelligence and Computing Research, 2013, pp. 1-5.

[4] M. Shinde and P. Kulkarni, Reading of Energy Meter based on Image Processing Technology,

International Journal of Electronics Communication and Computer Engineering, Volume 5, Issue (4) July, Technovision-2014, pp. 1-5.

[5] K. Parthiban and A. M. Palanisamy, Reading Values in Electrical Meter Using Image Processing Techniques, IEEE International Conference on Intelligent Interactive Systems and Assistive Technologies, Coimbatore, INDIA, 2013, pp. 1-6.

[6] Elias Farah, Isam Shahrour, "Smart Water Leakage Detection: Feedback About the use of Automated Meter Reading Technology", Sensors Networks Smart and Emerging Technologies 2017.

[7] Champ Prapasawad, Kittitachpornprasitpol ,Wanchalermpora, "Development of an Automatic meter reading system based on ZigBee pro smart energy profile IEEE 802.15.4 standard", International Conference on Electronic Devices and Solid-State Circuit (EDSSC), pp.1-3, Dec2012.

[8] NajmusSaqibmalik, Friedrich kupzog, Michael Sonntag, "An approach to secure mobile agents in automatic meter reading", IEEE, International Conference on Cyberworlds, computer society, pp. 187-193, 2010.

[9] Abhinandan Jain, Dilip Kumar, JyotiKedia, "Smart and intelligent GSM based automatic meter reading system", International Journal of Engineering Research & Technology (IJERT), ISSN: 2278-0181, Vol2, Issue3, pp.1-6, May2012.

[10] H.G. Rodney Tan, C.H. Lee and V.H. mork," Automatic power meter reading systems using GSM network". IEEE, 8th International Power Engineering Conference, pp.465-469, 2007.