Electronic Road Pricing System Prototype

Lundy Orlando Tjandra, Saptadi Nugroho, Darmawan Utomo
Department of Electronics and Computer Engineering
Satya Wacana Christian University
Salatiga, Indonesia
Email: lundy.orlando@gmail.com, saptadi_nugroho@yahoo.com, darmawan@staff.uksw.edu

***Abstract*—Traffic congestions are common in every large cities. This paper presents Electronic Road Pricing system design prototype based on embedded system Raspberry Pi with RFID and image processing capability based on OpenALPR. The system works in a way that vehicle passing under the gantry will be recorded by photographing the image of license plate and reading the RFID tag. The image and tag will be serialized to XML and transmitted to server through a socket. The output is transaction record and license plate reading from OpenALPR. Early test showed that RFID is able to work optimal at range between 0-360 cm and is able to work at vehicles moving at 10, 20, and 40 Km/h except OpenALPR which failed two out of three tests because of minimum data training. After trained with 1000 Indonesian license plates, the OpenALPR is able to recognize nine out of ten that given randomly.**

Keywords—RFID; ERP; Embedded; OpenALPR; Image Processing

I. INTRODUCTION

According to Statistics Indonesia *(Badan Pusat Statistik)* there are 67 millions of vehicles operating in 2009 and the number in 2013 grows to 104 million in Indonesia [1]. This means 55% percent growth rate in five years. With only 2.5% road growth rate [2], bottleneck in Indonesian main roads are bound to happen at some point and will became worse over time. Traffic congestion happens on roads connecting point of interest areas such as business district. In order to relieve these roads from congestion, a system that discourage road user from using it must exist and one of them is Electronic Road Pricing (ERP).

ERP has been used as a means of easing traffic in congested areas in Singapore, London, and Stockholm. This system has been proven to reduce the amount of vehicles entering restricted areas/zones (RZ) effectively. It also increases the amount of vehicles utilizing the highway around the RZ thus increasing road usability in the RZ. A study by Singaporean Land Transport Authority proves the effectivity of ERP system [3]. In 1998 after switching from Singapore Area Licensing Scheme (ALS) to Electronic Road Pricing (ERP), there is a significant decrease in the road usage heading to RZ during peak hour and all day long. Road usage around RZ increased as high as 12.4% from East/West direction and 9.2% from West/East direction during morning peak hour. After the implementation of ERP system there is a behavioral change in the number of vehicles entering the Ring Roads and avoiding the RZ.

In this paper we propose an idea of the ERP system prototype using camera, Raspberry Pi, RFID system, and a personal computer as server. Unlike the system used by Singaporean Land Transport Authority, this system works without using any bulky onboard device such as in vehicle unit (IVU) that Land Transport Authority Singapore has except passive RFID tag alone [4]. The system is modelled in mono lane operation where one RFID reader and camera covers only a lane with an improvement in license plate recognition system and the exclusion of bulky IVU and stereoscopic cameras for vehicle classification [5, p. 8/6]. The system classifies vehicle on the database side using license plate number and associated RFID tag requiring all vehicle to be registered first before operating on the road. The OpenALPR as a tool to recognize license plate is trained with varied Indonesian license plate images and fonts. The system is able to recognize the characters from a license plate image. The license plate number and the RFID tag will be compared to the data stored in the database system for further recording and identification.

II. SYSTEM CONCEPT AND DESIGN

The process to recognize the license plate number and reading the RFID tag are described here along with the ERP system prototype design.

A. Image Processing (OpenALPR)

OpenALPR is an open source framework specifically designed to recognize license plates from all over the world by combining few libraries such as OpenCV as a means of image processing and Tesseract OCR as a means of reading the characters out of license plates. It is designed to work in a pipeline and there are 11 steps such as detection, binarization, char analysis, plate edges, de-skew, character segmentation, OCR and post processing [6].

1) Detection

Detection phase is used to get the license plate region in the picture. Local Binary Pattern (LBP) is used as algorithm and the regions detected in this phase will be sent into Binarization phase.

2) Binarization

Binarization process happen once for each license plate region detected by detection phase. Wolf-Julien and Sauvola method are used as a method in binarizing the license plate regions.

3) Char Analysis

Character analysis will try to detect character in license plate region. The process is done multiple times.

4) Plate Edges

The plate edges tries to find the precise rectangle of the license plate.

5) De-skew

The de-skew stage will normalize region orientation.

6) Character Segmentation

The character segmentation phase tries to choose the region character in a license plate.

7) OCR

The OCR phase will output streams of possible characters and their confidences.

8) Post Processing

This step determines the best possible plate letter combinations.

B. Radio Frequency Identification (RFID)

Radio Frequency Identification is a wireless communication and identification method using electromagnetic fields as a media for sending data. Generally RFID system consists of a reader and a tag. There are two kinds of RFID system, the active and the passive one. In an active RFID system, the RFID tag uses a battery to power the transmission signals actively to the reader hence reducing reaction time. On the other hand in passive RFID system the tag uses the electromagnetic wave transmitted by the reader as power source hence it is relatively slower.

One of the RFID system is manufactured by COLI Building Material (BM-0702) using South Korean PHYCHIPS SoC PR9200. There are two interfaces available from BM-0702 the RS232 and RS485, and in this project, the RFID runs on RS232 mode at 112500 bps with transmit power at 25dBm. The RFID tag used in this system uses ISO 18000-6C (EPC Gen2) standard and works in the 860-960 MHz band. The tag has 512 bit of memory split into four banks. The four banks are Reserved Memory, Electronic Product Code (EPC) Memory, Tag ID (TID) Memory and User memory.

In this project, only one of the four memory above is used because the slowness of the tag in responding read command at certain memory banks sent by the reader. Therefore the only tag values used in this project is TID Memory; the default data sent by the tag when interrogated by reader [7, p. 25].

C. System Design

The ERP system consists of multiple Raspberries placed around an Access Point that acts as a router. As seen of Fig. 1 Raspberry Pi is connected to a camera, RFID reader, and communicating wirelessly to server and database. The Raspberries are dependent to server in order to process the image as Raspberry Pi doesn't have enough computing power required to process image within an acceptable time.

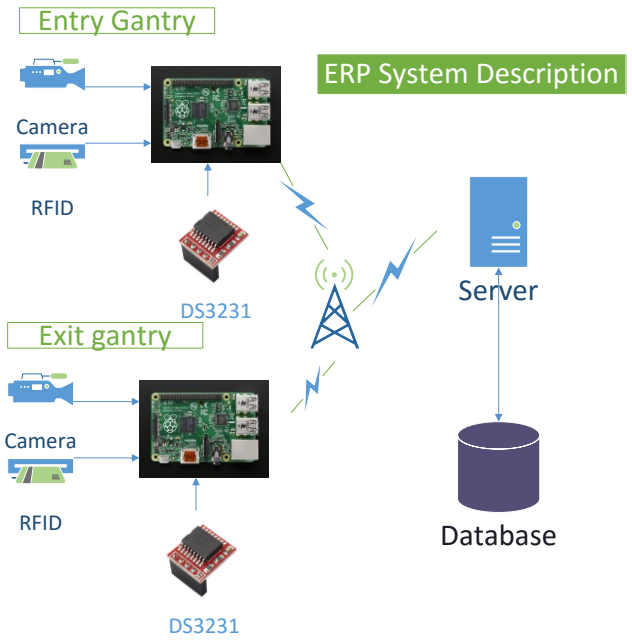

FIG 1. SYSTEM DESIGN

As seen on Fig. 1 Raspberry Pi is connected to camera, RFID reader, and communicating wirelessly to server. USB to serial with PL2303HXD chipset is used as an interface between RFID and Raspberry Pi clocking at 115200 bps. Camera communicates via built in CSI port. Real Time Clock (RTC) DS3231 communicates via I^2C bus in GPIO pin 3 and 5 (GPIO2SDA/GPIO3SCL) with 0x68 as default address and accessed using DS1307 kernel module . The RTC will be refreshed automatically when the Raspberries are connected into internet via Network Time Protocol (NTP).

The Raspberries are communicating with server and database through wireless LAN by utilizing socket programming. The server listens at port 10000 asynchronously and the clients (Raspberries) will attempt to connect to server when a vehicle is detected. The server is designed to be able to process 10 queues of incoming connection.

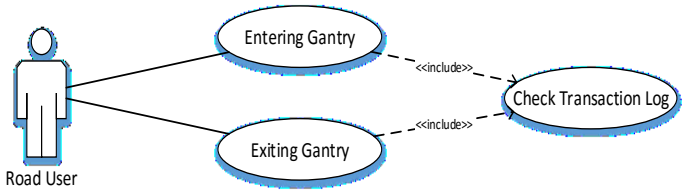

FIG 2. SYSTEM USE CASE DIAGRAM

The system works independently so simultaneous event such as exiting and entering gantry at the same time isn't going to block each other. The system works when the RFID

tag in vehicle is detected by the reader. Once detected, the Raspberry Pi is going to take picture and the tag. After the picture, tag and dates are taken, they will be encapsulated into an XML file before sent to server through socket. An encapsulation system is required to reduce overhead in the process, simplifying the data communication and easing the parsing process from socket stream. The XML file received by the server then processed further by decoding the content and uploading it into database. Database will record the transaction accordingly into transaction table, triggering the transaction bill column to update itself after looking the proper transaction price in road table. For each transaction (ins and outs) the vehicle amount, transaction price and total vehicle column in road table are updated by triggers too. This is done in order to be able to charge user based on road occupancy level. The user then will be able to see the transaction via a web server. The user then will pay the bill when it is time to pay the annual tax or by charging the RFID card directly (this method isn't applied in the trial because of card slowness). RFID and image recognition are applied in order to prevent card swapping (by matching card tag with license plate number). The only actor is a user as seen on Fig 2.

D. Software Design

The ERP system consists of several software split into three main parts: the client, server and database.

1) Client

The client software is written in Python 3. Python is chosen because the large libraries included in Python Standard Library. A few libraries used are `picamera`, `serial`, `IO`, `socket` and `time`.

2) Server

The server software is written in C# language on top of .NET framework. For user part, they can check their transaction record through a web page made from ASP.NET.

3) Database

The database management system (DBMS) used in this project is SQL Server 2014. There are four large tables in the database and they are Vehicle Owner Table, Road Table, Vehicle Table and Transaction Table. Each of these tables are tied with triggers. This is how the database is designed.

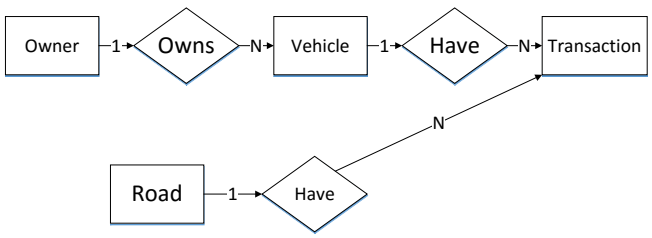

FIG 3. ENTITY RELATIONSHIP DIAGRAM

As seen on Fig 3. There are four main entity and they are Owner, Vehicle, Transaction and Road. Owner can own many Vehicle, a Vehicle have a lot of transaction and last Road have a lot of Transaction. The Transaction which represent Transaction Table in database contains [Transaction ID] as primary key, [Tag RFID] and [Road ID] as foreign key. For each transaction, system will record it into transaction table and then becomes a dependency from other table. At user side (web side) transaction table will be joined and displayed with other table such as vehicle table as info to user.

III. SYSTEM TEST

In this section, we explain the system test done in order to measure the capabilities of the ERP prototype system. This section contains three major parts; client test, server test, and system test.

A. Client Test

1) RFID and Raspberry Pi Test

The tag and the RFID reader are put 60 cm from each other until 540 cm. The RFID reader and RFID tag are able to work optimally between 0-360 cm. If the distance is farther than that, the RFID will fail to respond in timely manner even the RFID is working at maximum transmit power (25dBm) [7, p. 46].

2) Client Program Execution Time

Time is taken from Raspberry Pi 2 using `time.time()` method in the python client program. It runs in on-demand mode. The processing time to record a transaction needs 1.52 seconds in average [7, p. 47].

B. Server Test

1) Database Test

Database test is required to know the average time needed to record a transaction. Database tests are run on Intel® Core™ i3 3110M 2.4 GHz laptop with 8GB RAM and 5400 RPM HDD. After 30 tests run, the processing time to record a transaction is about 158 ms [7, p. 50]. The recorded data consist of the RFID tag, the entering time, the exiting time, the original picture, the OpenALPR read result, raw XML file and bills.

2) Socket Receive Time

The socket receive time is required to know the time needed from server program to receive data from socket. On average it took 313ms per transaction [7, p. 51]. A large time required per transaction happened because data are sent wirelessly and in non-optimal condition e.g. far and no line of sight.

C. System Test

In this test, the system tries to record vehicles passing the gantry. The RFID tag is able to be read correctly and in a timely manner by the RFID reader. The problem rises at the first system test that didn't include sufficient training data; both cascade training and OCR training are skipped to minimum level of training so the result is not satisfactory. OpenALPR failed 2 out of 3 from first trial [7, p. 53]. From the tests done, we can see on average it took 1,52 sec from client program execution time, 313ms from socket receive time (including recording into database) and 261ms from OpenALPR. In total we got 2097ms or 2 seconds per transaction.

1) OpenALPR Test

At the second test, ten random Indonesian license plates are passed to OpenALPR. Test are done as seen on Fig 4 and 5. Before the test, OpenALPR are given custom trained data for Indonesian license plate; both cascade for plate recognition and OCR for character recognition. Reading success rate after training is 9 out of 10 or 90% irrespective of wrong OCR confidence level [7, p. 55].

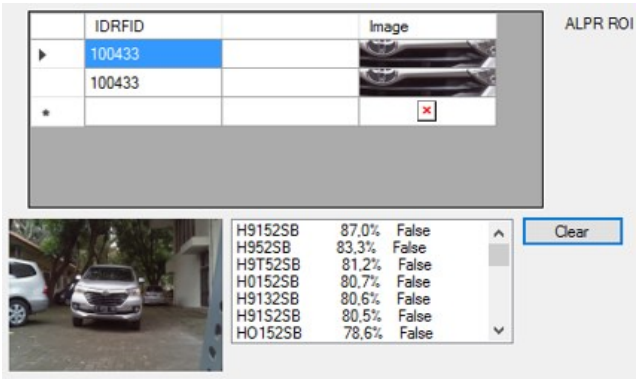

FIG 4. SERVER PROGRAM PROCESSING IMAGE AND RECORDING TRANSACTION

FIG 5. VIEW FROM THE RASPBERRY PI

TABLE III-1. OPENALPR READ RESULTS

Test	License Plate	OpenALPR	Time[b]
1	H8785ZC	H8785ZC confidence 87,85[a] H885ZC confidence 85,31	254
2	H8591PB	H8591PBJ confidence 89,17 H8591PB confidence 89,10[a]	323
3	H8524QB	H8520B confidence 89,57 H852QB confidence 87,98 H852OB confidence 85,36 H85240B confidence 84,76 H8524QB confidence 83,17 [a]	235
4	H7330FC	H330FC confidence 93,48 H33OFC confidence 84,95 H33QFC confidence 84,00 Failed	244
5	H8857DE	H8857DE confidence 94,13 [a] H88570E confidence 85,98 H8857OE confidence 85,61	256
6	H9341LB	H931LB confidence 90,45 H9341LB confidence 87,52 [a]	254
7	H9097NB	H9097NB confidence 85,2 [a] H9U97NB confidence 83,86	235
8	H9298HB	H9298HB confidence 91,87 [a] H9298H8 confidence 84,64	250
9	H30B	H30B confidence 91,60 [a] H30BO confidence 86,67	323
10	H38B	H38B confidence 91,32 [a] HS8B confidence 77,70	238

a. Correct Reading

b. average time 261ms

As seen on Table III-1, there are few wrong confidence levels. The wrong confidence levels are caused by the OCR and Post Process part in OpenALPR. A reliable system should have a correct confidence level so an unattended system can be made. Ideally the confidence level should be high and the license plate is read correctly.

IV. CONCLUSION

ERP prototype system that are able to record vehicles passing the gantry by taking photograph of the license plate and reading the RFID tag has been presented. The system prototype works successfully and it took approximately two seconds per transaction in average. This system shows that it is possible to build an ERP system based on Raspberry Pi with RFID and image recognition capability. In controlled testing the OpenALPR is able to detect and read 90% of license plates from random images containing a car. A few improvements have to be made including better image recognition, better reaction time and reading range by RFID reader, and better gantry design.

ACKNOWLEDGMENT

The authors would like to thank M. Hills of OpenALPR Technology, Inc and L. Fu from COLI Building Material.

REFERENCES

[1] Badan Pusat Statistik Republik Indonesia, "Statistics Indonesia - Kendaraan Bermotor," 13 March 2014. [Online]. Available: http://www.bps.go.id/tab_sub/view.php?tabel=1&id_subyek=17&notab=12. [Accessed 15 February 2015].

[2] Badan Pusat Statistik Republik Indonesia, "Statistics Indonesia - Panjang Jalan," 13 March 2014. [Online]. Available: http://www.bps.go.id/tab_sub/view.php?tabel=1&daftar=1&id_subyek=17&notab=11. [Accessed 15 February 2015].

[3] Land Transport Authority, "Research & Publications | Land Transport Authority Singapore," 2014. [Online]. Available: http://www.lta.gov.sg/content/dam/ltaweb/corp/PublicationsResearch/files/FactsandFigures/Traffic%20Flow.pdf. [Accessed 15 February 2015].

[4] Singapore Land Transport Authority, "In Vehicle Unit | Land Transport Authority," [Online]. Available: https://www.lta.gov.sg/content/ltaweb/en/roads-and-motoring/managing-traffic-and-congestion/in-vehicle-unit-iu.html. [Accessed 9 July 2016].

[5] P. Blythe, "RFID for Road Tolling, Road Use Pricing, and Vehicle Access Control," *RFID Technology (Ref. No. 1999/123), IEE Colloquium on,* pp. 8/1 - 8/16, 1999.

[6] M. Hill, "OpenALPR-Design," New Designs Unlimited, LLC, 9 August 2014. [Online]. Available: https://github.com/openalpr/openalpr/wiki/OpenALPR-Design. [Accessed 15 February 2015].

[7] L. O. Tjandra, Prototype Sistem Electronic Road Pricing (ERP), Salatiga: Satya Wacana Christian university, 2016.