ABSTRACT

The growing number of vehicles in developing countries causes a slew of issues, including the parking system. The current parking system is mostly manual, requires human intervention as a security system, and does not provide information about available parking areas. Their problems cause nonoptimal parking management. Furthermore, it can lead to income loss and criminal acts. This study addresses one of the possible solutions by using the internet of things (IoT) concept. The parking system is built by utilizing a smart card, machine-to-machine (M2M) communication, and cloud monitoring. As a result, the smart parking system prototype has been provided. The parking system business process can be done automatically, and it provides a more secure parking security system. The proposed parking system architecture also provides a practical system. The system only took around 1 second to perform the data transmission between nodes.

1. INTRODUCTION

The growing number of vehicles in developing countries has created new issues. One of the biggest problems is the parking. The majority of developing countries face parking issues such as a traditional ticketing parking system, unmanaged parking areas, ineffective use of parking lots, parking security systems, and disobedient parking users. Nonetheless, parking is one of the retributions collected by the local government in some countries, such as Indonesia. Furthermore, an unmanaged parking system can cause regional income loss and criminal acts. Nowadays, the most common parking system in developing countries relies on manual ticketing systems both in on-street parking and parking areas. This parking system uses a paper ticket as a parking receipt. In terms of security, mainly in the parking area, parking users take the parking ticket at the parking gate before entering the parking area. The ticket will be checked by a security guard before the users can leave the parking area. The biggest problem with this system is that parking users should keep a piece of paper until they leave the parking area. Moreover, the ticket doesn't guarantee security. As long as someone has the ticket, they can leave the parking area, even drive other cars [1], [2]. From the parking user's point of view, sometimes it is hard to know the available parking slots. The current system doesn't tell parking users about the exact available slots. Therefore, parking users take a long time to find an available slot. The available parking slot information is ideally shown to parking users before entering the parking area. It can be done by monitoring all parking slots in the parking area. This is the main challenge of an advanced parking system. The internet of things (IoT) may be the best approach to solving this issue.

Scanned with OKEN Scanner

IoT is a paradigm to make a machine able to communicate with another machine. It is known as machine to machine (M2M) communication [3]. It is widely adopted in many fields, such as home automation, industrial automation, health care, even the parking system [4]-[8]. IoT is also able to allow a common electronic device to be remotely controlled. By using sensors and gateways, it guarantees that users are able to manage the devices by sharing information between the devices. This paradigm is also referred to as smart devices. Combining smart devices into a smart environment not only advances device utilization, but also improves the user experience.

A smart parking system is a system that utilizes the IoT concept to manage the parking area. It can automate the parking gate, record the transaction, validate or authorize the parking user, or even send information about available parking slots [9]-[11]. In this study, we describe the design of a smart parking system using the IoT concept to enhance the parking system by automating parking records and improving the security system by utilizing a smart card. The smart parking system will provide fully automatic with secure control and the ability to monitor the system using an online monitoring system. It is also designed to minimalize the use of the internet on the part of users. The smart parking system was also brought in to manage the multiple numbers of a parking lot, allowing it to maximize the parking lot's utility. This study aims to understand the architecture and communication between various hardware, software, and communication protocols to develop an ideal smart parking system.

2. RESEARCH METHOD

In this study, we focused on applied research. We focus on the application of smart cards and the internet of things (IoT) to the smart parking system. This study was done by implementing, testing, and evaluating the ability of smart card and IoT concepts to solve a practical problem in the parking management system.

2.1. Smart card design

In this study, we designed a smart card parking system that is only accessible by authorized users. Therefore, we need a device to prove user identity. There are numerous alternatives to providing proof of identity, such as barcodes, quick response (QR) codes, or smart cards and QR codes are the simplest solution. However, local storage on another device or a cloud service is required to store user identity. It leads to communication problems due to the internet connection as a parking area is placed on the ground. Moreover, a smart card device is able to store a small amount of information. It is often embedded with radio frequency identification data (RFID) for data communication and data storage [12]. The advantages of RFID compared to barcodes and QR codes are able to perform many-to-many communication. Hence, it can be read by many readers [1], [13]. It also doesn't need an internet connection to do the data exchange. Another advantage of RFID based smart cards is their wireless data transmission. These abilities of RFID based smart cards are ideal as proof of identity in the smart parking system.

We utilized a mifare-based smart card. Table 1 shows the data stored on the smart card in our design. The Mifare smart card is able to store 1024 bytes of data divided into 16 sectors [14]. Each sector is divided into 4 blocks. In our smart card design, we only employ one sector. Moreover, we divided the smart card's blocks into two purposes. The first purpose is to store authentication data. We use the first block to store authentication data. The authentication data is static data that will be written one time during smart card personalization. The authentication data takes 16 bytes of data. The second purpose is to store transactional data. The transactional data reserves 3 blocks of data. The transactional data stores 7 types of information, which are the vehicle registration number, transaction date, the vehicle's status, parking gate number, the user's ID number, smart card expired date, and smart card status. Vehicle status records information about vehicles that enter and exit the parking lot. Moreover, the smart card status stores information about the active status of the smart card in our smart parking system. Table 1 displays detailed data on the amount of data stored on the smart card.

Table 1. Smart card data design

No	Data	Type	Length
1	Vehicle Registration Number	ASCII	10
2	Transaction Date	Hexadecimal	4
3	Vehicle in Status	Numberic	1
4	Parking Gate Number	Numeric	1
5	The User's ID Number	ASCII	18
6	Smart Card Expired Date	Hexadecimal	4
7	Smart Card Status	Numeric	1

Scanned with OKEN Scanner

According to the Indonesian government, the vehicle registration number in Indonesia consists of a regional code, registration number, and an expired date. However, in this study, the smart card-only stores a regional code and a registration number. The combination of regional code and registration number can be 7 characters or 10 characters in length. The format of the registration number is "L NNNN LL" or "LL NNNN LLL where the first and the second digit of "L" is a regional code. The "N" is the random registration number. The last "L" is a sub-regional code. The regional and sub-regional codes are presented in the Latin alphabet and regional numbers are presented in numerical numbers from 0 to 9. For example, N 1234 A is a vehicle that was registered in city of Malang. N is the regional code for Malang, 1234 is the random register number, and A is the sub-region of city of Malang. As explained before, it should be known that the vehicle registration number consists of the Latin alphabet and a numerical number. Therefore, we store the vehicle registration number in ASCII format. The current transaction date will be used to write the transaction date data. It occurs every time the user taps the smart card into the reader on the smart parking gate. It means that transaction date data will be overwritten every time new transactions occur. We use the DDMMYY format to store transaction date data and it takes only 4 bytes of data. Moreover, we use a hexadecimal format to store transaction dates for efficiency purposes. Vehicle status is the boolean value of a vehicle entering or leaving the parking area. Since the Mifare smart card is unable to store a boolean data type, we use a single byte of a numerical data type to store the data. The "0" value is for a vehicle's leaving status. The "1" value is for the vehicle's entering status. Parking gate number data is another type of numerical data. It only takes 1 byte of data, as in our smart parking system. The user's ID number uses an ASCII format to store the data. We designed it to take 18 bytes of ASCII format data to store the information. The ASCII format was employed to make the user's ID data more robust. The data type for smart card expiration date data is transaction date data. The last data that is stored on the smart card is the smart card's status. Since it only keeps active status data, we only need a single byte of a numerical data type. The data format is identical to the vehicle's status.

2.2. Smart parking system design overview

The smart parking system utilizes multiple sensors and gateways in order to make the system run seamlessly. It should be able to use multiple communication protocols to fulfill user requirements. In this study, the system relies on multiple groups of devices connected over a network that is controlled by an application using a specific protocol based on their roles. As shown in Figure 1, we divide the system into four major groups: sensor nodes, client nodes, gateways, and servers. The sensor node is used to collect information and display the results from the server. The sensor node can not directly connect to the server due to different types of protocols. Therefore, the communication between the sensor node and the server will be handled by the gateway. The gateway is able to communicate using multiple protocols for message exchange. The client node is a node that can directly communicate with the server. In this system, we use a client node to personalize the smart card and monitor the system. The server collects all information from the sensor node and client node. It is also able to send requested information from the sensor and client node.

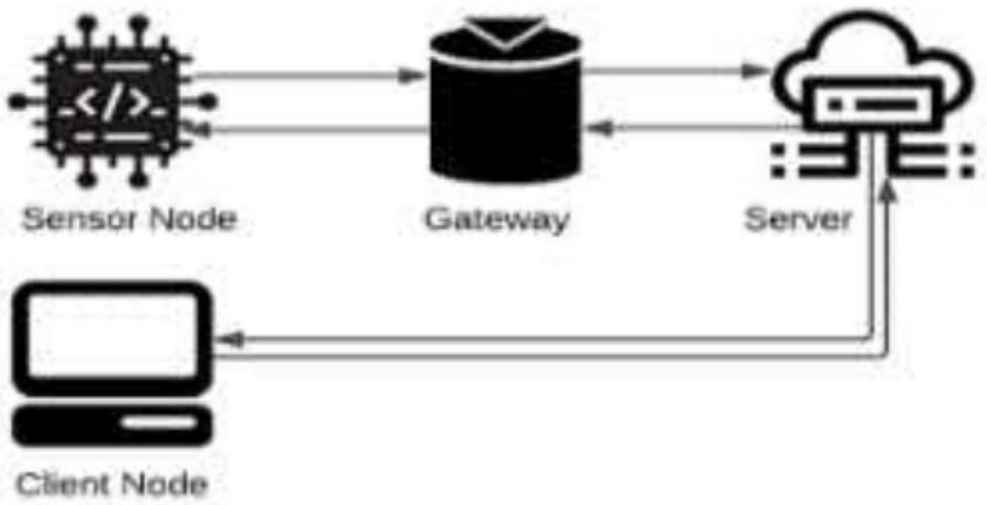


Figure 1. Smart parking design overview

2.3. Communication protocols

The internet of things (IoT) system development is based on real-time, yet reliable communication protocols. There are many protocols that can be used in the IoT system. However, the selection of protocols suitable for development purposes is challenging and it is a difficult task for the developer to choose an effective one. The reason for these difficulties is the nature of the IoT system, and it depends on IoT messaging requirements. Unlike the Web system, which utilizes a single simple standard protocol, the hypertext transfer protocol (HTTP), the IoT system can not rely on a single protocol for all purposes [15]. This is because the sensor node in the IoT system can be used for many types of network communication. Therefore, there is one standard that must be provided by IoT system protocols. They should be fast and

Scanned with OKEN Scanner

reliable for business transactions [16]. Protocols that provide fast and reliable transactions include the advanced message queuing protocol (AMQP) and the java message service. Moreover, numerous protocols have been designed for data collection purposes, especially from the sensor node. Protocols focused on data collection include message queueing telemetry transport (MQTT) and constrained application protocol (CAP). Some protocols are designed for instant messaging and online presence, such as the extensible messaging and presence protocol (XMPP) and the session initiation protocol (SIP) [16]. However, in some cases, the IoT system requires a user interface (e.g. the web) to administer the services. Therefore, these systems require web applications through an internet service, representational state transfer (REST), which runs over HTTP [17], [18], is the most popular architecture for data exchange [17], [18]. REST implements 4 essential components to perform data exchange, which are uniform resource locator (URL), HTTP verb, HTTP response, and various data formats (e.g. extensible markup language (XML) or javascript object notation (JSON)) [19], [20]. Because of its numerous requirements, the IoT system cannot rely on a single protocol.

2.4. Smart parking architechture

The complete architecture of the smart parking system is shown in Figure 2. The architecture is divided into four groups, as mentioned in section 2.2. The first group is the sensor nodes. It consists of 2 roles. The first role is the smart parking gate system and the second role is information display. The second group is the gateway node. The gateway node acts as a message broker for sensor nodes and a cloud service. The third group is the server node, which collects and processes the data from the sensor node. The last group is the client, which is an application that can be used by the smart parking system administrator.

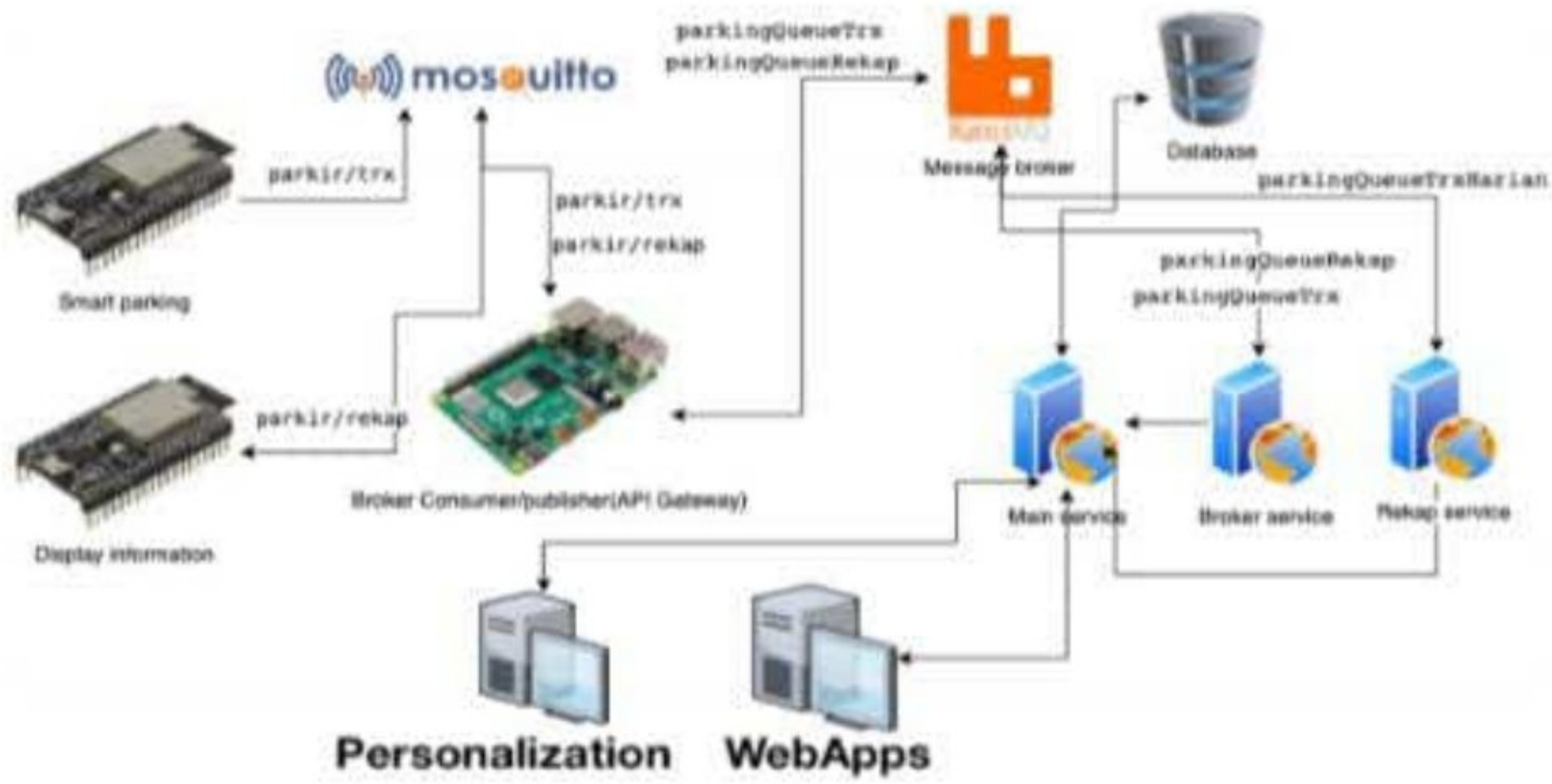


Figure 2. Smart parking architechture

2.4.1. Sensor node

We employ NodeMCU as a sensor node controller. The smart card reader is placed on the smart parking gate system. It can be found at both the entry and exit gates. It is used to read the user's smart card data before the user enters the parking area. We utilize the PN532 module as a smart card reader. PN532 is a module for contactless communication at 13.56 Mhz which includes microcontroller functionality based on 80C51 with 40Kb of ROM and 1Kb of RAM [21]. The information display node reads the data from the server and displays it to the user. The information that is outgoing or incoming to the sensor node will be processed by the broker through the MQTT protocols. MQTT works as a publish/subscribe mechanism for machine-to-machine (M2M) communication. MQTT clients publish a message to the MQTT broker, which forwards the message to the other clients. The message is published using a certain address, called a topic. MQTT clients can subscribe to multiple topics to receive every message published by the broker. The MQTT protocol is suitable for communication between the sensor node and the broker due to the small amount of header for data transfer. In this architecture, we employ Mosquitto as a message broker system. Mosquitto was chosen because it was an open-source project that anyone could use and modify. It is also suitable for low power single board to full server specifications. It also uses TCP as a transport protocol and employs TLS/SSL to secure its communication.

Scanned with OKEN Scanner

2.4.2. Gateway node

The Raspberry Pi was employed as a gateway node. In our architecture, the gateway node acts as an MQTT broker and an AMQP client. The gateway, as an MQTT broker, serves as both a publisher and a subscriber, depending on the sensor node's perspective. The MQTT broker acts as a subscriber while receiving a message from a smart parking gate system. However, it also acts as a publisher while sending a message to a smart parking information display node from a cloud service. The gateway node is also in charge of sending and receiving data from the cloud service via the AMQP protocol. When compared to the nodeMCU, a microcontroller, the Raspberry Pi has several advantages as a gateway node. The Raspberry Pi is a single-board computer that has enough memory to perform several tasks. We employ a 4GB memory version of the Raspberry Pi to handle communication tasks on both MQTT and AMQP. The AMQP protocol was used due to its ability to support a request/response architecture and also a publish/subscribe architecture [22]. Compared to other protocols such as MQTT, HTTP, or CoAP, AMQP offers a more secure communication protocol [23]. AMQP employs various approaches to TLS negotiation as well as a single port TLS model, Pure TLS and WebSocket tunnel TLS [23]. Because of the wide range and variety of messaging services, AMQP has been chosen as a business standard [24]. It is suitable for communication between a lightweight IoT node and a web client/server through a request/response architecture. In this research, we used RabbitMQ as an AMQP protocol broker. The RabbitMQ was installed on the cloud server.

2.4.3. Server node

The server node employs Docker as a container system. We virtualize all the servers for the smart parking system using a Docker container system. We use a container system to increase the effectiveness of a single server instance [25]. It is also easier to maintain the process and communication between the virtualized servers. In our smart parking system, we employ 4 servers, which are the database server, the main server, the AMQP broker server, and the recapitulation server. The main server is responsible for performing data transactions, including smart parking system personalization and monitoring. The AMQP broker server is responsible for managing communication between the main server and the gateway node. It acts both as a publisher and a subscriber. It acts as a publisher when sending a message to the gateway. On the other hand, it acts as a subscriber when it receives a message from the gateway and recapitulation server. Moreover, the AMQP broker server also communicates with the main server to send data from the gateway. In this case, communication occurs using REST. The recapitulation server manages all of the transactions occurring in the smart parking system.

2.4.4. Client node

As shown in Figure 2, the smart parking system has two client nodes, which are personalization and web monitoring nodes. The personalization client was employed in the registration system. It performs smart parking registration and writes data to the smart card for the users. We employ a python-based desktop application as a personalization client. The personalization client is attached to the smart card writer. We also use the PN532 module on this client. The details about the data on the smart card were already discussed in the previous chapter. The data collected from personalization was sent to the main server using an application programming interface (API) based on the REST architecture. For security and processing reasons, we use a REST-based API rather than directly accessing the database server.

As a monitoring node, a monitoring application was created. We utilize a web application to create a monitoring note. It contains a dashboard for the smart parking system administrator to monitor the system and transactional data. We employ the Java programming language and the REST API to communicate from the monitoring client to the main server.

2.5. Data communication

The smart parking system communicates with each order using MQTT, AMQP, and REST, as explained in the previous section. Communication between nodes occurs as a result of a specific topic, most notably on a sensor node, gateway node, and server node. The communication between nodes can be divided into 3 main pairs of nodes. The first pair of nodes are the sensor node and the gateway node. As mentioned before, the communication in this pair uses the MQTT protocol. Therefore, it uses a certain topic to send and receive the data, which is parking/trx and parking/rekap. The smart parking gate system publishes data to the gateway node using the parking/trx topic. It contains smart card data combined with a smart card serial number and a CRC32 value as a data corruption detection algorithm. The data was sent using this following format:

smart_card_serial_number#vehicle_registration_number#trascation_date#vehicle_in_status#gate_number#id #exp_date#card_status#crc32_value.

Scanned with OKEN Scanner

Moreover, gateway node also able to publish a data. It is used to send information from the server node to the sensor node. It happens when the parking/recap topic is used. The information was sent using the javascript object notation (JSON) format and it was displayed on the information display. It contains the transaction ID, the transaction date, the gate ID, the gate name, the number of parking slots used, the number of available parking slots, and the current time.

The second pair are the gateway node and the server node. These pairs employ AMQP protocols to communicate with each other. The gateway node sends the data to the server node using the parkingQueueTrx topic. It contains the card serial number, gate number, transaction date, status, and CRC32 data. The communication between the gateway node and the server node is managed by the broker server. The broker server subscribes to the parkingQueueTrx topic to receive the data. Moreover, the data is forwarded to the main server through the POST method of REST. The data was collected and processed on the main server before being returned to the gateway node. The previously processed data is returned to the broker's server. It was published using the parkingQueueRekap topic. It contains information about the transaction date, gate number, gate code name, and parking slot availability number. Furthermore, REST communication was used to communicate between a server and a server.

The third pair is the server node and the client node. In this pair, we employ the HTTP protocol through REST for information exchange. Both personalization and monitoring applications are able to request information from the server node served by the main server. They use HTTP methods such as GET and POST to receive information and send data to the main server.

3. RESULTS AND ANALYSIS

The smart parking prototype employs desktop and web applications at the client node. The desktop application was used as a smart card personalization system. Figure 3 shows the interface of the personalization client. The "Perso Kartu" interface is the interface to write data to a smart card, as shown in Figure 3(a). It contains a UUID, Name, Student ID, Birthdate, Licence Number (vehicle license plate number), and Sex. Figure 3(b) shows the interface to read smart card data. The web application was used to manage a smart parking system in real-time. The Web application shows information about available parking slots for each parking area managed by a smart parking system. Moreover, it also provided a smart parking administrator with information about the transactional data in the smart parking system. Transactional data was obtained from the gateway node. Figure 4(a) and Figure 4(b) show the web monitoring application dashboard and the transactional interface sequentially.

Figure 3. Smart parking system personalization interface: (a) smard card write data interface and (b) smart card read data interface

Scanned with OKEN Scanner

(a)

(b)

Figure 4. Web monitoring application: (a) dashboard interface and (b) transcation data interface

Figure 5 depicts the entire prototype of a smart card parking system capable of simulating real-world conditions. The initial condition of the smart parking gate is closed, as shown in Figure 5(a). To enter the parking area, users must have a smart card that is already registered in the smart parking system. The smart card acts as a user and vehicle identifier to open the parking gate. Therefore, only known users can enter or leave the parking area. The authorized user is able to open the gate as shown in Figure 5(b). Furthermore, the data on the smart card was read by the smart parking gate and the data was sent to the gateway. As described in section 2.5, we employ the MQTT protocol to send the data from the sensor node to the gateway using the parking/trx topic. Table 2 shows the message published to the gateway node by using the parking/trx topic. The data is then forwarded to the server node using the AMQP protocol using the parkingQueueTrx topic. Table 3 shows the message example sent from the gateway node to the server node through the parkingQueueTrx topic. Finally, the data will be collected by the server node and it can be requested by a web monitoring application.

Scanned with OKEN Scanner

Furthermore, information about slot availability in a specific parking area is displayed in the information display. The information was sent from the server node to the AMQP broker using the parkingQueueRekap topic. Table 3 shows the information details sent using parkingQueueRekap. The gateway node which subscribes to parkingQueueRakap forwarded the information to information display using parking/rekap topic through MQTT protocol. The details of the information are shown in Table 2. Smart parking data transactions also occur during parking checkouts. In this scenario, the system changed the vehicle's status and updated the transaction date record on the smart card and the server node.

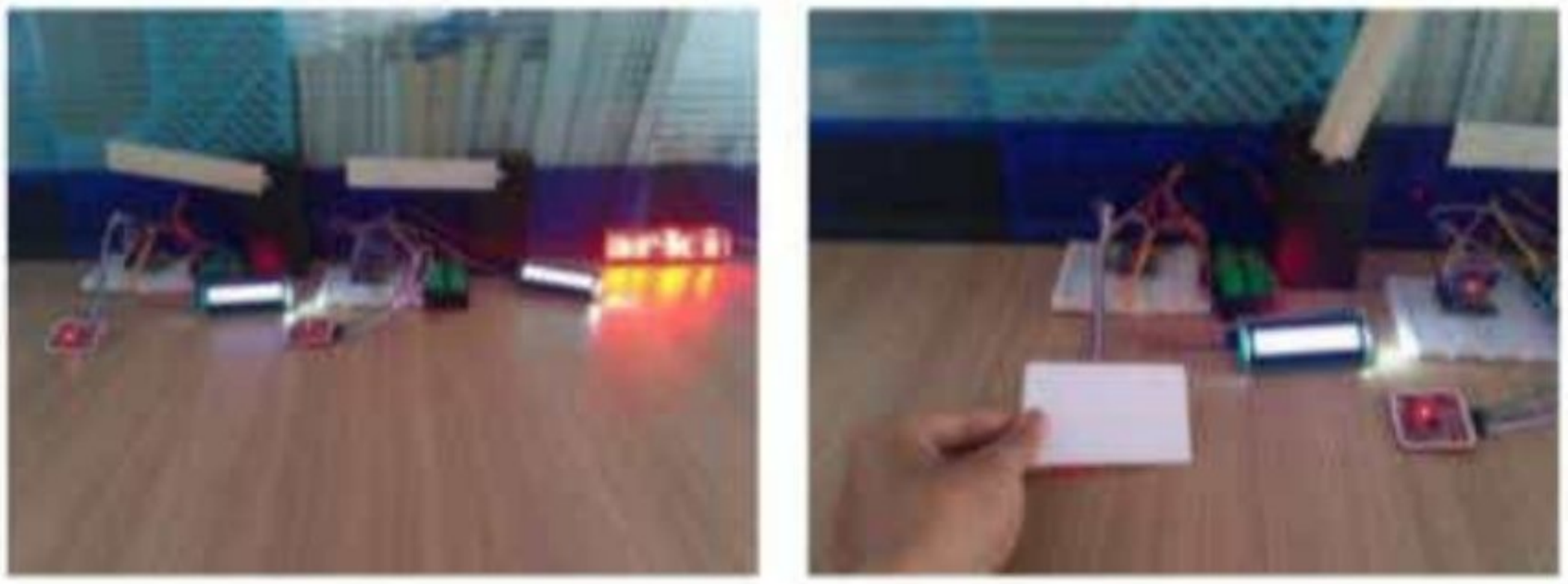
In ideal conditions, data transmission should be as quick as possible in real-world conditions. In order to measure data transmission performance in our system, we perform data logging for each data transmission lifecycle. A total of 54 pieces of information have been gathered. It found that data transmission between the sensor node and the gateway node (vice versa) took around 1 second. Moreover, it took 7.53 ms to transmit data from the gateway node and the server node and 7.51 ms to transmit data from the server node to the gateway node. The outcome demonstrates that a smart parking architecture is viable enough to be put into practice.

Table 2. MQTT topics

Topic	Message Example
parking/trx	0b7e7c22#AB1234YQ#1601992316#1#K001#1341180009#1745544405#1#2128116156
parking/rekap	[{"id":1,"tanggal":"2020-10-06","kode":"K001","nama":"kantongku","terisi":3,"sisa":7,"catat":"15:32:28"}]

Table 3. AMQP topics

Topics	Message Example
parkingQueueTrx	{"serial": "0b7e7c22", "kode_gate": "K001", "tanggal_transaksi": "2020-10-06 14:09:28", "status": true, "crc32": "253664524"}
parkingQueueRekap	[{"id":1,"tanggal":"2020-10-06","kode":"K001","nama":"kantongku","terisi":3,"sisa":7,"catat":"15:32:28"}]



(a) (b)

Figure 5. The prototype of smart parking system: (a) smart parking closed gate scenario, (b) smart parking opened gate scenario

4. CONCLUSION

In this study, a smart parking system was designed using the IoT concept. It employs several nodes, including a sensor node and a gateway node, to record and display information about the parking lot's availability. This can be useful for a user who wants to park their vehicle in a certain parking area. The transaction data can also be used to predict peak session times, allowing parking lot managers to plan ahead of time. This study also presented a simple security system using a smart card. It demonstrates that the use of smart cards can prevent unauthorized users from checking out or inspecting the parking area. Furthermore, the study will be focused on additional features of the smart parking system. It can focus on additional features related to system security, such as automatic detection of vehicle license plate numbers. It can increase smart parking security when combined with a smart card.

Scanned with OKEN Scanner