Sistem Pemantauan Temperatur, Level pH, dan Tekanan Gas untuk Pengoptimalan Produksi Biogas

Matthew Ryo Kianijaya*, Ratu Raihan Amany†, Ervandame Tarigan‡,
Eniman Yunus Syamsuddin§, Muhammad Ogin Hasanuddin¶
School of Electrical Engineering and Informatics
Institut Teknologi Bandung
Bandung, Indonesia
Email: {*13218035, †13218012, ‡13216049}@std.stei.itb.ac.id, §eniman@stei.itb.ac.id, ¶moginh@itb.ac.id

***Abstract*—Biogas merupakan salah satu jenis energi terbarukan yang memiliki potensi produksi sangat besar pada negara agraris seperti Indonesia. Produksi biogas memiliki beberapa masalah yang signifikan, yakni produksinya kurang efisien, tidak dapat diproduksi dalam jumlah besar, memakan waktu yang lama, dan memiliki kemungkinan untuk gagal jika tidak dipantau secara berkala. Maka dari itu, penulis menawarkan solusi untuk masalah yang ada. Solusi berupa produk yang merupakan sistem pemantauan proses biogas yang memantau beberapa parameter penting seperti suhu, level pH, dan tekanan gas pada biodigester. Spesifikasi produk dan hasil percobaan menggunakan produk pun akan dipaparkan dengan rinci.**

***Keywords*—Internet of Things, Biogas, Monitoring ambient parameter**

I. Introduction

Cadangan sumber energi fosil di Indonesia diprediksikan hanya dapat bertahan sampai beberapa tahun ke depan. Dengan demikian, diperlukan adanya terobosan baru dari sumber energi terbarukan yang bersifat keberlanjutan. Salah satu sumber energi terbarukan yang dapat dijadikan alternatif adalah biogas. Biogas adalah sebuah gas yang dihasilkan dari aktivitas anaerobik atau fermentasi dari bahan-bahan organik. Biogas menghasilkan energi bersih (clean energy), yang dapat dimanfaatkan untuk keperluan sehari-hari seperti memasak atau sebagai penerangan (listrik). Bahan-bahan organik utama yang menjadi bahan baku dari proses pengolahan biogas adalah limbah peternakan atau perkebunan. Dari bahan baku proses pengolahan biogas ini, dapat disimpulkan bahwa potensi produksi biogas di Indonesia sangat besar.

Pada kenyataanya, biogas masih dikategorikan sebagai sumber energi yang kurang efisien dan tidak bisa diproduksi dalam jumlah besar[ref1]. Hal ini dapat disebabkan karena tidak semua produki biogas menghasilkan hasil yang optimal dan pengolahan cenderung memakan waktu yang cukup banyak yaitu 20 sampai 40 hari[ref2]. Produksi biogas sangat dipengaruhi oleh beberapa faktor seperti suhu dan nilai pH. Untuk menghasilkan produksi yang optimal, maka kedua parameter tersebut harus dijaga nilainya pada rentang tertentu di suatu waktu proses fermentasi. Namun, parameter pada proses pengolahan biogas di Gabungan Kelompok Usahatani (Gapoktan) Wargi Panggupay yang berlokasi di Lembang, Bandung Barat, belum dapat dipantau secara kuatitatif.

Oleh karena itu, produk yang akan menjadi solusi adalah sebuah alat pemantau parameter suhu dan pH saat proses fermentasi berlangsung. Kemudian, diperlukan juga pemantauan parameter tekanan yang akan menjadi indikator bahwa proses pengolahan biogas berjalan dengan baik. Fitur dasar dari produk adalah dapat mengambil data parameter suhu, pH, dan tekanan serta produk memiliki antarmuka atau interface untuk menampilkan data parameter yang telah diambil. Fitur tambahan dari produk adalah produk mampu mengirimkan informasi ke suatu aplikasi pada smartphone pengguna dan menyimpan informasi tersebut ke dalam suatu log data yang dapat diakses. Selain itu, produk juga dapat memberikan peringatan kepada pengguna jika parameter yang dipantau berada di luar set point. Adanya fitur peringatan ini, berfungsi agar petani atau operator dapat memberikan input substrat yang sesuai agar parameter yang dipantau Kembali berada di rentang optimalnya. Produk diharapkan mampu mengambil data dengan akurat, hemat daya, mempunyai harga yang terjangkau, tidak membutuhkan perawatan yang terlalu intensif, dan mudah digunakan oleh orang awam khususnya bagi kalangan petani.

II. Related Works

Due to high demand of such systems, there has been several previous attempts to design a vehicle Tracking system, all has involved the use of GPS, with GPS receivers being the component to use and integrate in the device. One previous attempt involved the system sending location data in the form of Short Message Service (SMS) to alert the user regarding real time location of the vehicle, as well as detection of theft. Detection of theft is done through an RFID sensor to detect an event of engine turning on. Data is then presented via Google Maps link that will redirect the user to an external Google Maps application to view the location. Control such as commands to the device are done manually through the SMS via specific text-based commands [1]. Another attempt in designing a vehicle tracking system involves the design of Java-based system to allow users to interact and retrieve positional data from the device, while still using SMS-based messages to control the configuration of the device in the event of a position information request. Location data is sent via TCP socket through GPRS network to be stored in a

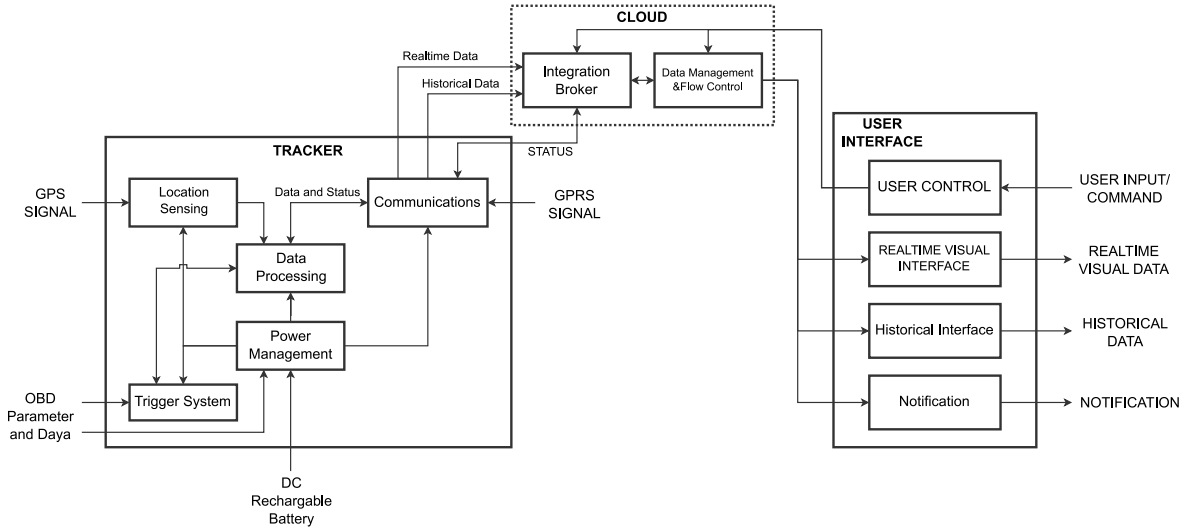

Fig. 1. Architecture of the Proposed System

database [3]. Another system was designed to be accessed with a mobile-based application running specifically on iOS hardware, where the location data is displayed via Google Maps, but without any capabilities to detect any attempt of theft, and is used solely for vehicle location tracking. The referenced system uses GPRS as a method of communication while GPS is used for location detection [4]. Lastly, a system was designed using the basic architecture of a GPS receiver while integrating the use of a cut-off switch to be able to turn off the vehicle's power once it has detected an act of theft. The detection of the act of theft still needs confirmation from the user on whether it is a true-positive / false-positive detection, every time the vehicle engine turns on. Once the user has confirmed this act of theft, the cut-off switch will then turn the car off and the system will be able to deliver the location of said vehicle [5].

III. Proposed System

The paper presents a system which is designed to locate the vehicle location using GPS and GPRS technology via hardware planted on the user's vehicle. The system will be based on Espressif ESP32 Controller to handle the operations of the devices, with integrated SIMCOM SIM800L GSM/GPRS Module to be able to communicate with the network to transmit information to the user. GPS capabilities will be handled by a separate GPS receiver to acquire latitude and longitude values. In order for the system to be able to power itself, while also detecting an act of car theft, the device will interface with the On-board diagnostics (OBD) port of the vehicle, where it will receive constant voltage to power the device, while also receiving data in the form of impulses which will be detected by the device to determine an engine start event. Besides that, the device will be able to detect if it is unplugged from the OBD Port, where an emergency power source in the form of rechargeable Lithium-Ion Batteries will be used to power the device for a certain period of time.

All the information gathered from this device will then be transmitted via Internet through GPRS network provided by support network providers. For data to be received to the end user, it will have to travel to different stages. The first part of the transmission involves the data being sent to Google Pub/Sub Client, which is a part of Google IoT Core services. The device will communicate with Pub/Sub Client via MQTT, a lightweight communications protocol to enable low power data communications to minimize power consumption, especially in the situation where emergency power is used.

On the other side, the user will interact with a web application design as a Progressive Web App (PWA), able to be accessed in different devices with access to a web browser. The decision to create a user interface based on PWA is that it enables implementation of features such as push notifications and native user experience for iOS and Android systems running on smartphones, while being more lightweight in terms of file size and processing power usage compared to traditional apps [6].

To integrate the communication of both the device and the user, Google Cloud Functions will be used to create functions that will run whenever a certain event has occurred. These functions will save incoming data from device to Firebase Firestore, pass through commands to enable and disable both Realtime Tracking Mode and Theft Monitoring Mode to the device and pairing said device to a specific user account. Thus, it can be inferred that Cloud Functions will be the backbone of the system in order for both the device and user to communicate and exchange data with each other. The proposed system can be seen in Figure 1.

IV. Tracker Subsystem

Tracker Subsystem is the subsystem that will do data acquisition on location, theft detection and other data such as network status and GPS signal availability. The process of

development for this subsystem is done using Mongoose OS platform. Mongoose OS is development framework focused on IoT aspect and allows for seamless and easier integration to cloud infrastructure such as Google Cloud, Amazon AWS and Azure IoT. It supports multiple devices such as ESP32 and ESP8266, where ESP32 will be used in this system.

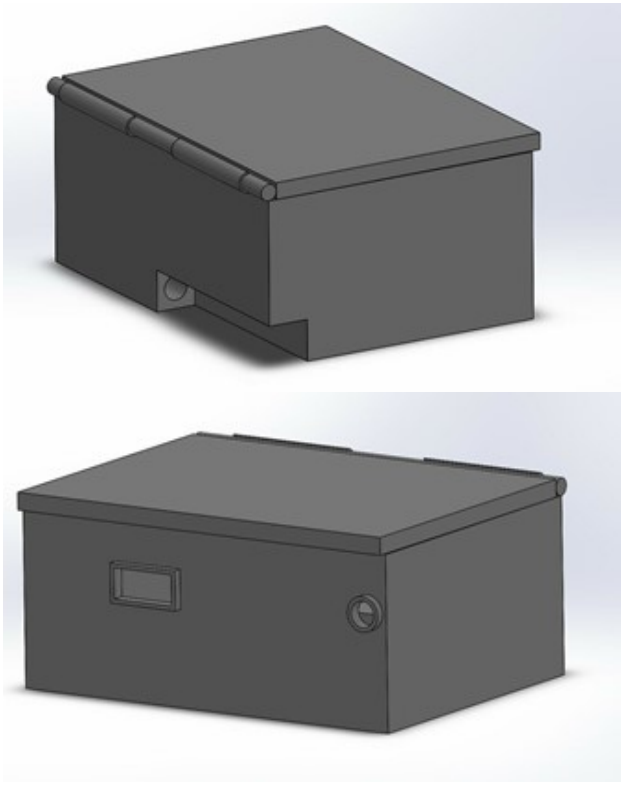

Fig. 2. Tracker Device Casing

A. Data Processing and Communications Module

Data Processing Module consists of TTGO-TCall ESP32 Module. This Module has integrated SIM800L module that will act as mobile modem to connect to telecommunications mobile network. It also includes IP5306 System-on-Chip (SoC) to power up the board using battery, and controls battery charging process. This board allows for conventional communication between module via GPIO pins that are available, and providing power pins to modules such as GPS Module.

B. Location Sensing Module

The functionalities of coordinate data acquisition for real time tracking and historical data collection is fulfilled by u-Blox NEO-6M Module. NEO6M module is a GNSS receiver with capabilities of receiving from US GPS satellite network. Compared to newer variants from u-Blox, notably the M8 Series receivers, the NEO-6M lacks the ability to receive data from Galileo and GLONASS satellite networks. However, as with the choice of SIM800L module for communication, the availability and price of the module are the key reasons on why

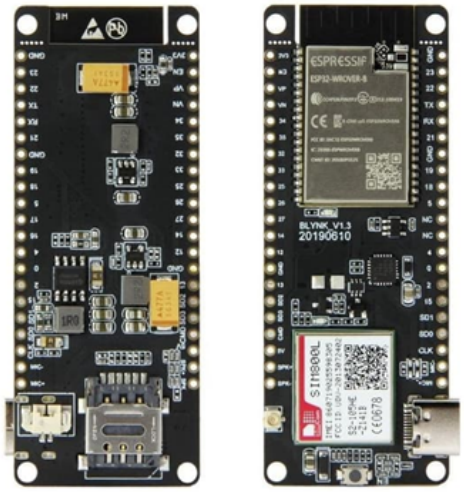

Fig. 3. TTGO T-Call ESP32 Board with integrated SIM800L Communications Module

NEO-6M is used. Communication between NEO-6M Module and ESP32 module are as follows:

- 'Tx' Pin on NEO-6M connects to pin 'GPIO14' of ESP32module.
- 'Rx' Pin on NEO-6M connects to pin 'GPIO12' of ESP32module.

C. Theft Detection Data Acquisition Module

Theft detection is possible when the system meet one of the two scenarios. In the first scenario, the system will detect theft if vehicle is moving from its original position where the user initially turns on the configuration. This scneario can be fulfilled from comparing location data that has been acquired.

In the second scenario, when the vehicle engine turns on. The second scenario, however, will need additional interfacing modules to gather more information. This information is the information of the car being on or off. To get this data, the device will interface with the vehicle with On Board Diagnostics (OBDII) Port. OBDII Port are mandatory features that can be found in cars. While different cars have different standards for their OBDII ports, where each port can send different types of data, it is crucial for this system to be used in types of cars. Therefore, detection of engine turning on is done by monitoring for impulses in the data lines of the individual pins, where the data pins will be active only once the engine of the car is turned on.

D. Backup Power Module

The system designed will have a backup power module in the form of rechargeable Lithium-Ion Batteries. These batteries need to be able to last a certain period of heavy usage in the case of the device being unplugged from the OBDII port. The system uses 2 3400 mAh Panasonic NCR18650B batteries, totaling the capacity to 6800 mAh. This will be connected via a proprietary connector available in the ESP32 module,

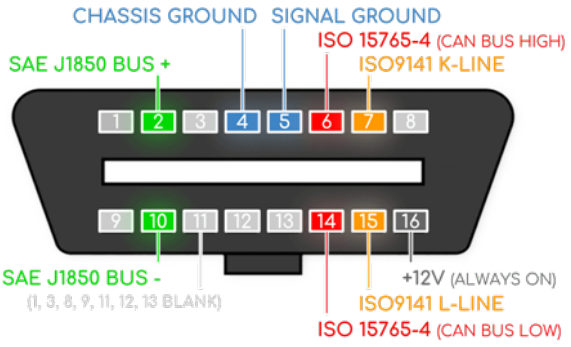

Fig. 4. Pinout Diagram of an OBDII Port

and charging will be managed by IP5306 SoC module as mentioned previously.

V. Cloud Subsystem

Google Cloud will be the provider of the Cloud infrastructure, where multiple solutions offered by Google Cloud will work together to perform the function of transporting data from Tracker to User Interface, and vice versa.

A. Google Cloud IoT Core

Cloud IoT Core allows for securely connecting and managing IoT devices to the Google Cloud Infrastructure, and acts as the connecting bridge of these devices to other Google Cloud Solutions [7]. Devices connect to IoT Core via registry that is created from Google Cloud Console. This registry, when devices are connected, will be able to show a full log of input and output of the device, or the CONFIG and STATE of the device in relation to the display of the device in the log.

B. Cloud Functions

Cloud Functions allows triggering of certain methods and functions via different solutions that are available at Google Cloud, where data from a solution can be used by another solution. One example of this is triggering a function from a change in Firestore Database, or triggering a change in Pub/Sub Bucket to activate certain functions [8]. In this case. Cloud function will be used for the following:

- Store telemetry sent from Tracker to Firestore Database, triggered by a new publish of Data from Pub/Sub Client of the Created IoT Core Registry
- Send configuration updates from User Interface to change the modes of the Tracker, triggered by an HTTPS request from the front end (i.e. press of a button available in the user interface)
- Pair a device with a user account in order for it to be accessed by said account, triggered by an HTTPS request from the front end
- Send push notifications based on change in Firestore document data, triggered by a change of data in the field of a Device Document in Firebase Firestore

C. Firebase Firestore

The database that we used for this project, divided into two main collections that are 'devices' and 'users'. The 'users' collection contain documents of all the registered users, containing email data and notification tokens to be used for receiving notifications. The name of the document uses the randomly generated user ID from Firebase Authentication. The 'devices' collection contain documents of all the registered and activated tracker devices, where the name of each document correspond to the name of tracker that is created when device is connected to Google IoT Core. This is to make sure that the data sent from a specific device will arrive to the correct document with the same name. The document itself contains several fields, consisting of real time longitude and latitude; owner of the tracker, which corresponds to the user ID that is generated and used by Firebase Authentication to identify a user; the fields 'Trigger' and 'ReqReal' containing values of either '0' or '1', which will be identified by the Tracker to be used to change the configuration of the device; the fields 'isConnect', 'isGPS' and 'baterai' corresponding to mobile data connection status, GPS data availability and battery level respectively; the fields 'Daya' and 'Pencurian' to identify whether the device has been plugged off from the OBDII port and to identify if the vehicle has been stolen, respectively. These data will be used from each side of the system (Tracker side and User Interface Side) by Cloud Functions and create a communication system between these two ends. Linked to the Tracker Document is a collection of historical data that will contain historical data documents. These historical data documents consist of latitude, longitude and timestamp data to differentiate themselves from each other.

VI. User Interface Subsystem

This subsystem will be responsible for the user's interaction, which includes account management functionalities such as creating and accessing accounts, receiving push notifications in regard to events such as theft detection and the unplugging of OBDII port, pairing trackers with an account, as well as a method in which to access data received from device to a presentable visual form. This subsystem will also allow the user to control different configurations of the device, in order to change the behavior of the device. The user interface is built as a Progressive Web App, essentially a website with a service worker to implement functions such as offline caching and notification event listening. Progressive Web Apps also allow the user to install the application as if the program is a native application for each respective device, currently supported in Windows 10, iOS and Android devices.

This subsystem consists of multiple pages, each performing a different task. Login page will be used for both creating a new account and login to an existing account. Dashboard page will be the main page to access telemetry Data such as real time location, availability of mobile network and GPS data, and backup power reserve percentage. Dashboard also allows initial pairing of device to the account. History page will be used to access historical data of each paired device.

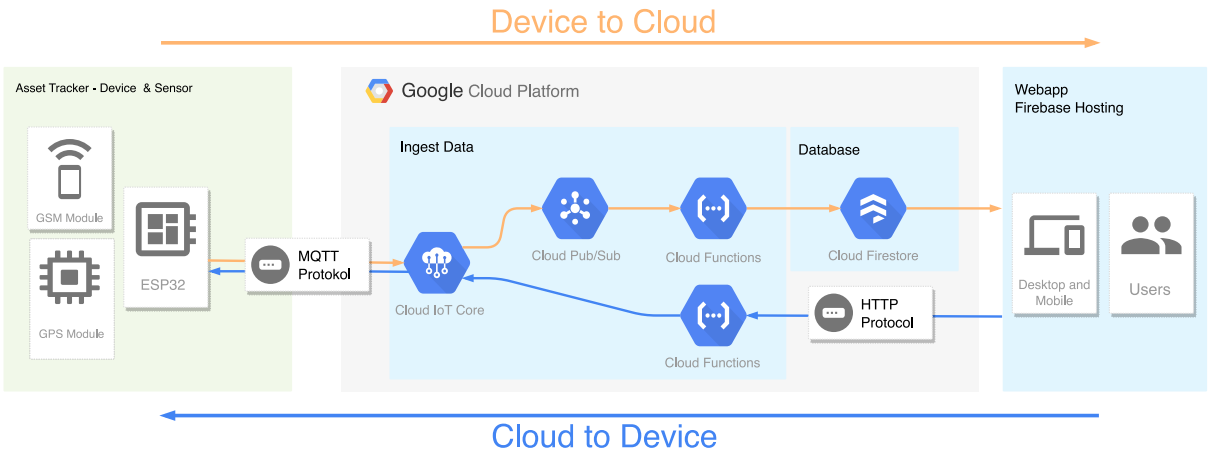

Fig. 5. Cloud Functions in Relation to Other Subsystems

All of the location data will be displayed via Google Maps and shown in relation with the current location of the user via their geolocation data.

A. *Firebase Authentication*

Firebase Authentication will be used to handle account management request, such as creating an account, verifying an account and accessing an already created account. Firebase Authentication allows a simple implementation of authentication as it is in the form of a ready-to-use function that can be called from the Firebase Authentication SDK. Firebase Authentication allows multiple methods of account creation, such as Google Account Login or Facebook Account Login. In this system, the user can create an account with an email, then set a password and lastly will be able to verify their account via email verification. Users who are able to verify their accounts can log in and access the Dashboard page where the user will be able to add/pair activated trackers. The figure below the design of the User Interface for register and login processes.

B. *Firebase Cloud Messaging*

Firebase Cloud Messaging allows the function of implementing push notifications to users, by working together with Service Worker of the Progressive Web App application, and Cloud Functions to send notifications such as a warning that the vehicle has been stolen, and a message regarding disconnect of device from OBDII port. Firebase Cloud Messaging uses tokens to trigger notifications, where the tokens will be given to a user when accessing the Dashboard page. This token will be saved in Database in the user's document section of the database. Notification will be triggered once an event happened, in this case a change on the fields of the Tracker document as mentioned previously.

C. *Firebase Hosting*

Firebase Hosting will be used to serve all the resources of the application, and thus accessible via domain. One of the key advantages of using Firebase Hosting is the ability to serve securely via HTTPS and therefore communication between the user and the server is encrypted. Firebase hosting also allows deployment of different versions of the application, which is useful in a situation where a new deployment can cause problems, and a quick 'undo' action is necessary to make sure the user's experience will not be interrupted by the problematic deployment.

D. *Google Maps API*

Google Maps API will be used to visually display the coordinate data received from the Tracker Device. Google Maps is implemented using Maps JavaScript API that is available to be used using an API key generated on Google Cloud Console. Alongside the location data of tracker will also be the location of the user, who will use the geolocation data of the device used to access the user interface (i.e. mobile phone or desktop device). Maps API will be used to display both real time location and historical location.

VII. Results

A. *Real time Tracking and Historical Data Acquisition*

Real time location data acquisition is possible through the Dashboard page. The user will first choose which Tracker Device. The chosen Tracker Device will then be selected, and the latest recorded location will be displayed in Google Maps interface, along with the location of the user accessing the application. The user will then be able to choose to turn on real time tracking to get the latest location data from a button element available in the Dashboard. Once it is selected, it will create a change in data in Firestore Database by changing the value of the 'ReqReal' field from '0' to '1', where it will be detected by a Cloud Function that is designed to detect changes in the document. This will then send a new configuration to the device, telling the device to turn on real time tracking, where it will provide an update at a rate of every 10 seconds, provided that GPS data is provided by the receiver.

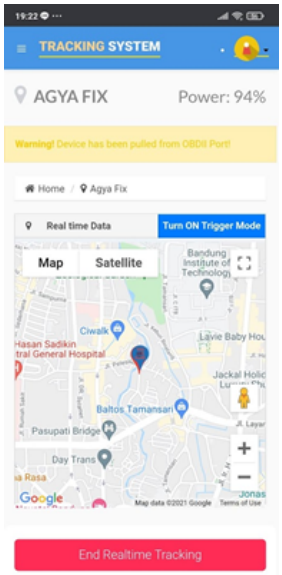

Fig. 6. Real Time Tracking Turned on in Dashboard

The tracker device will also send historical location data automatically without prompt, providing the user with extra precaution in the case that they forgot to enable theft detection mode. Historical data can be accessed in the History page of the application.

B. Theft Detection Simulation

Theft detection is possible by turning on theft monitoring configuration, by pressing the 'Turn ON Trigger Mode' button in the dashboard Page. Turning on this future will change the 'Trigger' field from '0' to '1', and this changed will be detected by a Cloud Function, and will instruct the device to change configuration and monitor for scenarios of car theft. When theft is detected, it will send a push notification the user's device, while also displaying a message to remind the user of the situation when they access the Dashboard page. A notification will appear as shown in figure 7 if a theft is detected.

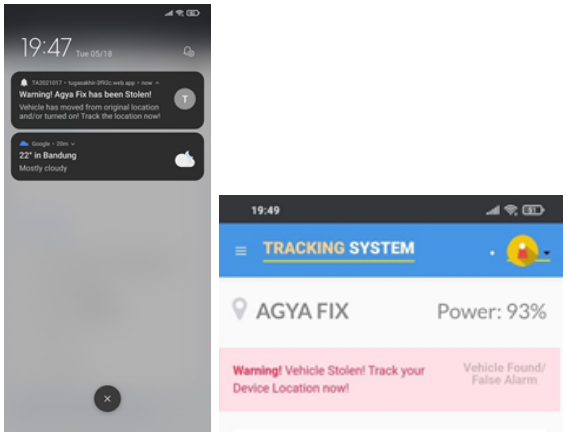

Fig. 7. Notification and Displayed Message when Theft is Detected

VIII. Conclusion

A system has been designed that has been able to incorporate GPS and GPRS technology to track location of passenger cars. The system is also able to detect an act of theft through 2 predefined scenarios, which are when engine is turned on and when the vehicle has moved 50 meters from its original position. The user is able to access all of the information given by the Tracker through a user interface designed as a Progressive Web App which enables features such as push notifications to be possible in a web-based application. Communication between Tracker and User Interface is fulfilled with Google Cloud Solutions, consisting of Google IoT Core, Cloud Functions, and Firebase Firestore. Implementation is possible in passenger cars with OBDII Port, where engine-turn on data and main power is acquired. Backup power is provided by a battery pack for situations where OBDII Port is unplugged, which will also be informed to the user via push notifications.

IX. Future Development

This system can be enhanced further by implementing the use of 4G LTE instead of GPRS once the components are available for a more economical price point. Any technology that can be used is to use Narrowband-IoT (NB-IoT) as a means of communications once the infrastructure is available to be used for general public in Indonesia. Implementing NB-IoT will allow for less power consumption and enable the device to be powered solely by external power source, like a battery pack, and enable more reliable mobile network connection. Another development would be offering this system to motorcycles, by changing the interfacing format to the vehicle from OBDII to other means.

References

[1] A. Mounika and A. Chepuru, "Iot based vehicle tracking and monitoring system using gps and gsm," *International Journal of Recent Technology and Engineering (IJRTE)*, vol. 8, no. 2S111, pp. 2399–2403, 2019.

[2] X. Lu, P. Wang, D. Niyato, D. I. Kim, and Z. Han, "Wireless networks with rf energy harvesting: A contemporary survey," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 2, pp. 757–789, 2014.

[3] I. M. Almomani, N. Y. Alkhalil, E. M. Ahmad, and R. M. Jodeh, "Ubiquitous gps vehicle tracking and management system," in *2011 IEEE Jordan Conference on Applied Electrical Engineering and Computing Technologies (AEECT)*, 2011, pp. 1–6.

[4] S. Lee, G. Tewolde, and J. Kwon, "Design and implementation of vehicle tracking system using gps/gsm/gprs technology and smartphone application," in *2014 IEEE World Forum on Internet of Things (WF-IoT)*, 2014, pp. 353–358.

[5] M. Ramadan, M. Alkhedher, and S. Alkheder, "Intelligent anti-theft and tracking system for automobiles," *International Journal of Machine Learning and Computing*, pp. 83–88, 01 2012.

[6] N. Pande, A. Somani, S. Prasad Samal, and V. Kakkirala, "Enhanced web application and browsing performance through service-worker infusion framework," in *2018 IEEE International Conference on Web Services (ICWS)*, 2018, pp. 195–202.

[7] G. Cloud. Cloud iot core overview. [Online]. Available: https://cloud.google.com/iot/docs/concepts/overview

[8] G. Developers. Cloud functions — google developers. [Online]. Available: https://developers.google.com/learn/topics/functions