MIOT Framework, General Purpose Internet of Things Gateway using Smartphone

https://doi.org/10.3991/ijoe.v14i02.7326

Toni Tegar Sahidi(✉), Achmad Basuki, Herman Tolle
Universitas Brawijaya, Malang, Indonesia
sahidi@student.ub.ac.id

Abstract—The Internet of Things (IoT) is an integrated system consists of several components linking each other. A framework is often used to fasten the development of an IoT system. This paper presents MIOT, a general purpose gateway framework which utilizes a Smartphone as the main component to handle communication between IoT device and the Internet. MIOT provides a bidirectional end-to-end communication gateway for Smartphone to link the IoT device and the Internet. For evaluation purpose, two implementations of IoT prototypes are built: an environmental monitoring and a remote controller (RC) car. The implementation of both prototypes demonstrates faster development times and quick response times in data transfer. MIOT framework helps reduces time, effort, and learning overhead on building and deploying IoT systems. The use of Smartphone as the gateway in MIOT offers a trade-off between easiness, latency, and reducing IoT devices workload.

Keywords—Internet of Things Framework; Gateway; Smartphone; WebSocket (keywords)

1 Introduction

Internet of Things (IoT) is an ecosystem where daily Things are connected to the Internet to gain more benefit. Atzori [1] mentioned that 'Things' in IoT can take a form of sensors, actuators, or both. However, IoT is considered as a complex system. Building such one is not an easy task. Zanella et al. in [2] said that the development in IoT often obstructed by the newness of IoT, complexity, and also very few ***'best practices'*** in building one. IoT device's heterogeneity, connectivity, and real-time communication [3] are some problem which often needs to be tackled to develop an IoT system. Such complex system will hinder IoT development by non-technical skills, and also IoT deployment by an ordinary user (with no or little technical expertise).

To ease the development of an IoT system, IoT developer often uses a framework, a collection of some coordinated component to make it reusable then fasten in building a system [4]. There had been existing IoT frameworks available. Most of these are

focused on data integration and collaboration. Only a few address the communication issue and none which supports real-time communication nor mobility. In this research, a general purpose communication framework for IoT is built named MIOT (Mobile Internet of Things). MIOT make use of Smartphone as the main component. The Smartphone is used as the gateway for IoT device, providing encapsulation and IP stack service.

The use of Smartphone in IoT ecosystem is not a new idea. As presented by Kamilaris and Pitsillides in their Survey [5], most research and products use Smartphone only as the controller for IoT device. This common Smartphone-IoT solution put Smartphone as the controller for IoT device and used in user (human) side. Meanwhile, the framework in this research, put the Smartphone into the 'Things' side which aimed to help 'Things' to be connected and communicate with the Internet**.** In this case, the Smartphone is used as the gateway (relay) for IoT device.

The idea of Smartphone as the gateway to help Things connect with the Internet was first coined by Golchay et al. [6] in 2011, then by Zachariah et al. [7] and Roy Want et al. [8] in 2015. Those researchers proposed an architectural concept where Smartphone can be used as a bridge or gateway for IoT devices. However, they only discussed the idea and there still no proof of concept on how it should be carried out.

Although there are some real-world commercial product that uses Smartphone as the gateway in IoT, it should be emphasized that most of them are product specific gateway. In most product, the Smartphone-gateway approach works only on particular product and system only. This approach means new IoT product will require new Apps and only works in a specific closed system, despite that these products use the same communication technology (Bluetooth). Those are the case that mentioned by Zachariah et al. [7] happened in Smartwatch from Apple, Motorolla, and Samsung. An approach which further said, that this closed siloed system hinder the growth potential of IoT. There still no 'general purpose' and ready to use Smartphone based gateway which could ease and speed up IoT development by researcher and developer. MIOT Framework as presented in this paper is an attempt to realize the concept of using a Smartphone as a general purpose gateway for IoT devices.

The justification regarding the use of Smartphone in IoT ecosystem is based on three considerations. First, in many IoT cases, IoT needs a gateway (relay) system to communicate their data with the Internet. One notable example is the Things which based on RFID or Bluetooth which not based on IP based communication. In such system where a gateway is required, Smartphone can fill in as a gateway device. Secondly, Smartphone is considered a complete computer system. It is equipped with a various method of communication, either in low level (e.g., Bluetooth) and higher level (IP based communication through WiFi, 3G, etc.). [9]. These features could be leveraged to build a relay system for IoT. The third consideration, with more than two billions Smartphone sales per year worldwide [10], make Smartphone as one notable ubiquitous device. Smartphones are widely available, and many are already inexpensive, in this way make Smartphone easily replaceable. Combined with mature technology and lot of supporting technologies make Smartphone as a good candidate in supporting IoT development.

Putting the Smartphone into the IoT device side aimed for several benefits. First, it will ease and speed up developer in building IoT system, especially that require real-time communication capabilities. Second, the ubiquitousness of Smartphone will help people with few or no technical skill to be able to build or deploy IoT system. MIOT will help the ordinary user do basic customization in deployment phase without doing programmatical changes to the IoT device. Thus, it will improve usability for a non-technical user. Third, Smartphone will also help on device mobility in case IoT device need to be mobile or located in outdoors in a remote area where no cable Internet connectivity existed.

Wearable technologies, like Smartwatch and Smartband, is an area where Smartphone based IoT schema gain popularity. As explained by Rawassizadeh et al. [11], limited power, constrained I/O interface and weak computing power in Wearable device often force the industry to create the complementary Smartphone apps. In many cases, the Smartphone Apps also sync the wearable data into their cloud server. However, it should be noted that: (1) The Smartphone Apps primary function is still to 'control' the Wearable device. (2) The Smartphone is still more on 'user-side' rather than 'Things-side.' And (3) All wearable still follow closed-siloed approach with paradigm one wearable only works for one specific App, and in some cases, for one particular Smartphone. Different from the wearable approach, MIOT is designed to be as general purpose as, and as Things-centric as possible. MIOT is also intended to allow more customization (data sources and server destination), and support bidirectional (send and receive) real-time data communication.

MIOT Framework consisted of three components. (1) MIOT Apps in the Smartphone, (2) MIOT Server on the server, and (3) MIOT Controller to provide a control interface for the user. MIOT Framework uses WebSocket [12] as a protocol for real-time communication. To picture how MIOT Framework is applied in an IoT environment is like this: An IoT devices with Bluetooth connectivity is connected to MIOT Apps (Smartphone). MIOT Apps then set to communicate with MIOT Server. A user uses a MIOT Controller to connect to MIOT Server. MIOT Server will then manage all communication between MIOT Apps and user, enable a user to send/receive data to IoT devices.

In this paper, MIOT Framework is presented as one best-practice in building IoT system. This paper is written as follows. Section 2 discusses some related works on IoT Framework, Smartphone, and IoT Gateway. Section 3 presents the MIOT Framework component and architecture. Section 4 shows the applicability of MIOT Framework by implementing into two usage scenarios. Performance Evaluation of MIOT Framework is presented in Section 5. More discussion about the result and some insight are presented in Section 6. The overall research is concluded in Section 7.

2 Related Work

The use of the Smartphone for IoT is not entirely new. Kamilaris and Pitsillides [5] wrote a survey that presents many research and products related the usage of

Smartphone in IoT. In most case, the Smartphone still used as the controller, or as "user-side" sensing device.

In term of the Smartphone as IoT framework, there is research by Gray et al. [13] that build SPHERES. SPHERES is a web service framework for IoT that uses Smartphone's sensors as the source of data. However, the framework only still in architectural concept only. There is no implementation or real product available yet. Moreover, SPHERES only use the Smartphone as the source data by using Smartphone's internal sensor and does not provide connectivity with IoT devices. Mentioning research on the real-time framework for IoT itself is still rare. One that can be referred is research by Zhang et al. [14]. In their study, they build a framework to handle real-time communication in the manufacturing environment. However, just like SPHERES did, the framework is still in architectural concept only. The research also did not mention on using Smartphone.

The concept of how the Smartphone can be utilized as IoT gateway can be dated back in 2011 where Golchay et al. [6] propose an idea on Smartphone act as service gateway between IoT devices and cloud services. In 2015, Roy Want et al. [8] also explained some architectural concept where Smartphone can be used either as bridge or gateway for IoT devices to communicate to the Internet. Further research by Jun Li et al. [15] even put the concept "Smartphone as the gateway in IoT" in Future Internet Architecture (FIA) affirming that this concept has a promising future. However, their research did not much focused on Smartphone based IoT, but more on FIA benefit instead. This concept was further confirmed with research by Zachariah et al. [7] that deeply discuss on IoT-gateway problem and challenge. Their research explained that constrained IoT device is not expected to support full IP Stack, thus need to offload this functionality to a more sophisticated device like Smartphone. Further, their research also discusses one notable problem: that most Smartphone based IoT-gateway product still use closed-siloed approach. The Smartphone based gateway only works for specific products, with a specific system, and even specific Smartphone only. This close approach is restricting the growth potential of IoT. In their research, Zachariah et al. encourage an open-general purpose-Smartphone based gateway system for IoT, which use Bluetooth Low Energy (BLE) as the communication system.

Despite how Golchay et al. [6] Want et al. [8], and Zachariah et al. [7] explained on how the Smartphone could be further used in IoT, there still no implemented framework for IoT gateway which already available, functional, general-purpose, and Smartphone-based yet. Most framework still not for general purpose, and only work in closed specific system only. It is concluded that there yet no research on general purpose, Smartphone-based IoT gateway Framework which ready to use. Even more, the framework which can handle real-time communication between IoT devices and server.

3 MIOT Framework

MIOT Framework serves three main features, that are: (1) Bridging communication between IoT devices to the server through Smartphone, (2) Manage communica-

tion in the server, and (3) Provide a front-end interface for the user to communicate with IoT device. MIOT Framework is designed for IoT developer, who need to deploy real-time communication infrastructure quickly.

The architecture of MIOT Framework is shown in Fig. 1. The Framework consists of three part: an Android app (MIOT Apps), NodeJS [16] server code (MIOT Server), and a web page as the front end for the user (MIOT Controller). Please note that MIOT Apps and IoT Hardware are inseparable as one IoT device.

Seen from IoT stack perspectives, MIOT Framework is implemented on the device, server, and user level. IoT developer only needs to build their IoT devices, which will communicate with MIOT Apps. The rest will be seamlessly handled by MIOT Framework. The implementation diagram of MIOT Framework is shown in Fig. 2.

A scenario of use is as follows. IoT developer, connect their IoT devices through MIOT Apps. MIOT Apps then is set to connect to MIOT Server. A user using MIOT Controller will connect to MIOT Server, then be able to communicate with IoT devices.

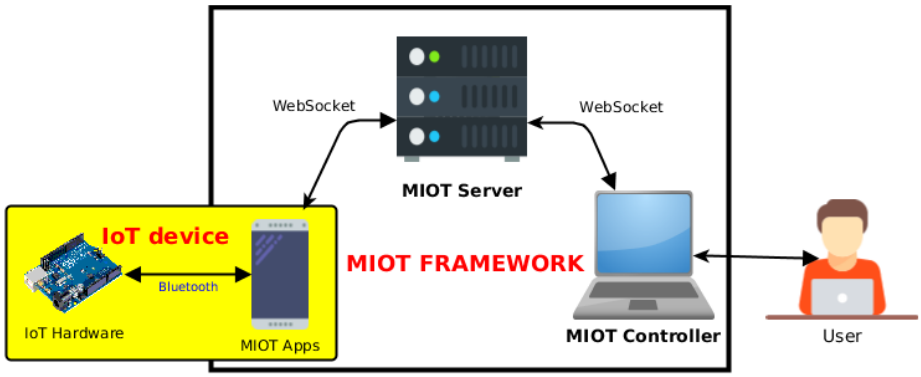


Fig. 1. MIOT Framework Architecture

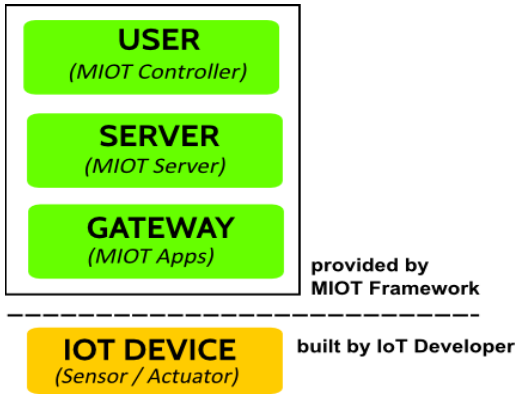


Fig. 2. Implementation Diagram of MIOT Framework

MIOT framework is designed to be as modular as possible. Each component can be utilized independently even without using other MIOT component. However, each component uses the same data structure called MIOT Data Structure to manage the communication.

3.1 MIOT Data Structure

WebSocket protocol support both String (utf8) and binary data format [12]. Proposed MIOT Framework uses String data format as its primary communication data since it will be easier to handle and more intuitive in use. All data, which communicated in the MIOT Framework, is first encapsulated into a JavaScript Object Notation (JSON) format [17]. JSON is a widely-used data format consisted of a key-value pair in each data set. MIOT Data Structure encapsulates a data along with sender information about data itself. The JSON Structure for MIOT encapsulation is shown in Fig. 3.

```
{
   “type”   : ID|MESSAGE,
   “text”   : STRING_TEXT,
   “id”     : ID_OF_SENDER,
   "group": GROUP_OF_SENDER,
   “time”   : TIME_DATA_SENT
}
```

Fig. 3. Implementation Diagram of MIOT Framework

The key "type" is used to define the kind of the message. Type can have one of two value, that is "id"-type, or "message"-type. The "id"-type is used by the sender to inform client identity to the server on first connection only. The "message"-type is used to send the real data and employed by both MIOT Server and MIOT Apps. The next keys, "id" and "group" are used to give information about who sends the message and on which group the message should be forwarded. By default, all connected user and Things, which have the same group name as defined in the message, will also receive the message. The key "time" contains information about when the data is sent. It has a value of "unsigned long"-type number that defines a UNIX timestamp.

The main part of data is set in "text" key-value that contain the main payload of exchanged data between all endpoint in MIOT. The user may set this value up to 1024 Bytes length.

This MIOT data structure is used between framework component only. Meanwhile, the communication between MIOT Apps and IoT devices through Bluetooth is done as is without any pre-formatting or encapsulation. If IoT developers want to send more complex data structure, they can first encapsulate the data using their format, then put the formatted data as the text value. Once the data has arrived at the other

end, IoT developer can configure the program to extract the data again to access his complex structure message.

All encapsulation process is done in the background. IoT developers only need to deal the "text" value as the data, and other keys-value as the data-source information.

3.2 MIOT Server

MIOT Server function as a rule manager between all communicated message. It decides which message should be forwarded to which group based on "group"-key carried in the MIOT data structure. It should be noted that the use of MIOT Server is not mandatory. IoT Developer may use their server code, as long as it follows MIOT data Structure used by other MIOT components.

MIOT Server code is written in NodeJS environment [16]. The Server consists a package folder with one main JavaScript file. MIOT Server is designed to be as simple as possible, with few step to run it. Once NodeJS environment is installed in IoT developer system, running the MIOT Server merely is one command away. Since MIOT Server is written in JavaScript, all code is also highly readable and editable. All configuration and setting were put on the top of the files to ease user's configuration. Some example of user's configuration such as choose to use the database, set preferred port, etc.

MIOT server also supports save to database function. If this function is enabled, then all received data from any ends (user, or device), will then save into the database. A time stamp about when a piece of data arrived into the server is also recorded into the database.

3.3 MIOT Controller

To minimize IoT developer's effort, MIOT Framework also equipped with MIOT Controller. MIOT Controller is a web page designed for a user who wants to control an IoT device (send and or receive data). By using MIOT Controller, a user can connect to a server, receive any data from IoT devices in the same group, and also send data to all connected peers in the same group.

MIOT Controller is only a single HTML page. It implements a chatting layout to improve 'intuitive' usability. This layout is chosen since the communication is basically like chatting, only, in this case, user 'chat' with IoT device rather than with the human.

There are two kinds of MIOT Controller: standard-chat controller, and auto-key-send controller. In a standard-chat controller, the user will need to press Enter keys each time he wants to send data to the server, enabling the user to send longer data like words or sentence. In the auto-key-send controller, every single key pressed by the user will immediately be sent to the server without the need to press Enter, enabling the user to send a fast-paced data to the server.

IoT developers can customize MIOT Controller for their needs by modifying specific functions on receiving data and sending data. All other functionality like con-

necting, disconnecting, encapsulation and extraction are done internally by MIOT Controller. IoT Developers -even though they can- no need to modify these features.

Just like MIOT Server, the use of MIOT Controller is not mandatory either. IoT developer may use their code that suits their requirements.

3.4 MIOT Apps

The main component of MIOT Framework is an Android application (apps) named MIOT Apps. The apps function as a communication gateway between IoT device -which use lower level communication (e.g., Bluetooth)- into Internet infrastructure. MIOT Apps will manage communication between MIOT Server through the Internet connection, and an IoT device through Bluetooth.

The structure of MIOT Apps is shown in Fig. 4. Inside MIOT Apps, there are two running thread, one handle WebSocket communication, and the other handle Bluetooth communication. Eventbus library [18] is used to connect the thread. The apps work by listening to two sides, any incoming packet in one side will be automatically forwarded to the other side. When the data is transmitted from one end to another, the data is encapsulated, or extracted depend on where the data is forwarded.

IoT developer only needs to configure information about MIOT server (address, port, id, and group), pair the apps with a Bluetooth devic e, and then start the service. Once the app is set and connected, it will listen to any data from either WebSocket or Bluetooth. Once data is received in one endpoint, it will automatically be forwarded it to the other end. MIOT Framework works 'passively' since it does not periodically check the availability of data on both ends like request-response communication infrastructure.

MIOT Apps is also responsible for data encapsulation and data extraction for data from the server. Please note that there is no encapsulation done for communication with Bluetooth. Once data is received from MIOT Server, it will be extracted. Only the "text" part then are sent to Bluetooth.

The current version of MIOT Apps still supports only one channel communication (a pair between one MIOT server and one IoT device). It means that MIOT Apps can handle communication between one server and one Bluetooth device.

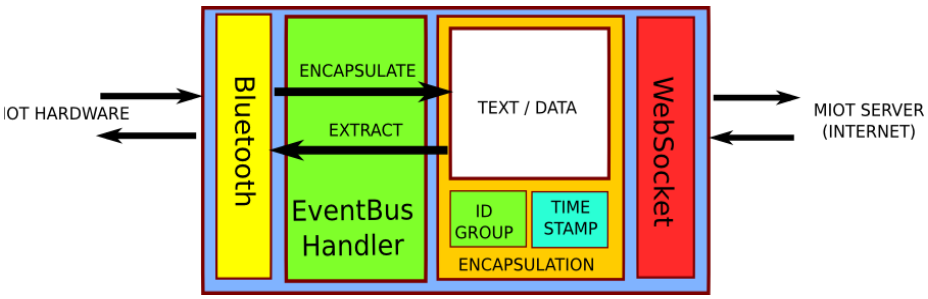


Fig. 4. Architecture of MIOT Apps

4 Implementation and Setup

MIOT is designed to enable IoT device (1) send and (2) receive data from the Internet. Send and receive are the basic functionality in IoT involve sending data from the sensor and receiving data to be handled by the actuator. On the other hand, most products related to Smartphone based IoT gateway only enable IoT device sending data but not otherwise. Unlike the other, due to WebSocket capabilities, MIOT able to do both send and receive in one single product. To verify these functionalities, two IoT usage scenarios which often used in IoT application were built.

First, an IoT for Environment monitoring system is created. An IoT device equipped with sensors will send data about air environment. Including temperature, humidity, and levels of Carbon Monoxide (CO) gas. In second usage scenario, a Remote Control (RC) Toys Car is modified so that it can be controlled through the Internet.

In both usage scenario, Arduino [19], equipped with HC-05 Bluetooth module, is used as the main component for IoT devices. Please note that Both usage scenarios use the same MIOT server and the same Smartphone with the same MIOT Apps.

These two scenarios are chosen to verify two basic IoT functionality offered by MIOT. These both usage scenarios are also intended to show how quick and easy to implement and deploy IoT system using MIOT Framework. The reusability and general purpose of MIOT will help IoT developer to gain more focus on IoT hardware rather than the overall system.

4.1 Environment Monitoring System

The Environment Monitoring System collect data about temperature, humidity, and CO gas levels. Two sensors, DHT22, and MQ7 sensors are used. DHT22 is a sensor to gather data on humidity and temperature and MQ7 to collect data on CO gas level. The hardware used for Environment Monitoring System is shown in Fig. 5a. The Arduino is programmed to fetch data from these sensors every 10 seconds, encapsulate them into a single data with JSON format, and then send it to Bluetooth. This encapsulation help to keep track which data is temperature, humidity, or CO level. The data is sent through HC-05 Bluetooth Module to Smartphone (MIOT Apps), then forwarded to MIOT Server.

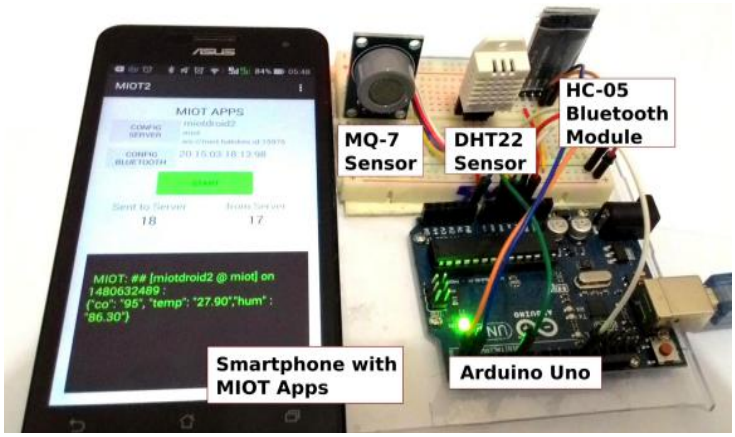
4.2 Internet Controlled RC Toys Car

This usage scenario uses an RC Toys Car that has two Direct Current (DC) motors. DC motors are electric motors which its rotation direction can be adjusted by switching the position of positive charge. In RC cars, the DC motors are placed in back and front of RC car. The back DC Motor is used to move the cars go front and back, while the front DC motor is used to drive the front wheels left and right. The prototype of Internet Controlled RC Toys Car system is shown in Fig. 5b.

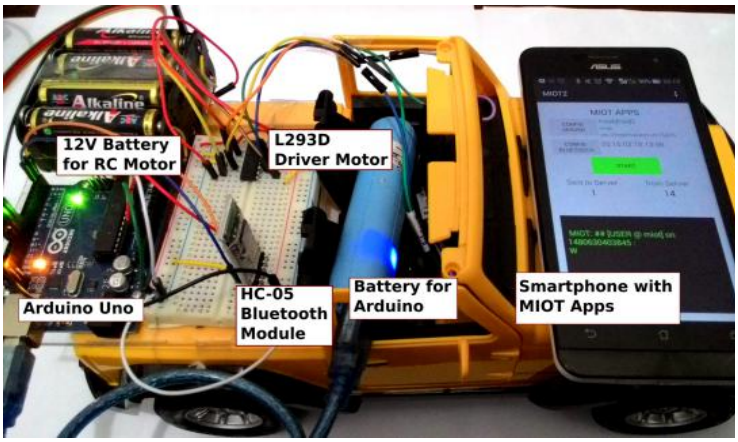
The Arduino is equipped with an L293D motor driver that function as a switch for electric current. It can control two DC motors and can switch its positive (+) and

negative (-) charge to motors based on input from Arduino. The Arduino is programmed to turn on and off the DC motors and also set which cable has + and - when receiving a particular character. The 'A,' 'S,' 'D,' and 'W' character are used as the key controller since it is often used in a conventional game key to defining the two-dimensional movement. Once a key character is received, the program will determine which motor and which + and - charge it is.

MIOT Controller with auto-key-send is used without any modification. This controller will directly send any character pressed on the keyboard to MIOT server. Despite the Arduino and MIOT Controller, the implemented of MIOT Apps and MIOT Server is the same as Environment Monitoring System. Based on conducted experiments, the movement of RC Toys Car can be controlled through the Internet using MIOT Framework with reasonable control response.



(a) Environment Monitoring System



(b) Internet Controlled RC Toys Car

Fig. 5. Usage Scenario Example of MIOT Framework.

5 Performance Evaluation Overview

Despite MIOT Framework purpose on easier IoT development and deployment, this design will affect overall communication performance. Moving IP Stack from IoT device to Smartphone is a possible -easiness trade off- on IoT development and deployment. Two experiment is conducted to give an overview of how this kind of framework affects overall network performance. First, a system with MIOT Framework is measured on latency performance. Second, a system without MIOT is also built then measured on latency performance. This second system called "Common Approach" as a 'comparison' with MIOT approach. Although this is not an apple-to-apple comparison, as previously stated, these experiments are intended to give an overview of how MIOT may differ from Common Approach (without Smartphone).

To measure the latency, a system is built as follows. MIOT Server is modified to sent periodic data to MIOT Apps. The data is just 1 Byte length which contains one character. As the IoT hardware, Arduino Uno is used. The Arduino has programmed to send-back any input it receives from the network, without any processing at all. It is like a schema where the Arduino is ping-ed and bounces back the data. Using this schematic, MIOT Server will send data to Arduino. Arduino will then send-back (reply) immediately to its sender (MIOT Server). The modified MIOT Server is programmed to record the time it sent the data and time it received the returned data. By looking at the differences between the two, the round trip time between MIOT Server to Arduino is obtained.

In the first experiment, Smartphone with MIOT Apps is set to forward data from and to both edges (MIOT Server and Arduino). The Arduino Uno is equipped with HC-05 Bluetooth module as the communication interface. The architecture of latency measurement using MIOT is shown in Fig.6. The round trip time between MIOT Server - MIOT Apps - Arduino - MIOT Apps - MIOT Server is named $T_{total.}$ The network latency between MIOT Server and MIOT Apps is also measured using standard ping packet and called as T_1. Since using MIOT Apps includes Bluetooth communication, the time required for the packet to travel from MIOT Apps - IoT Hardware - back to MIOT Apps is also measured and named as T_2. To measure T_2, MIOT Apps is modified, so it records the time between send to Bluetooth and receives from Bluetooth, then calculate the difference. Based on T_{total}, T_1, and T_2, the experiment can estimate the latency that caused by MIOT Apps (T_{Apps}).

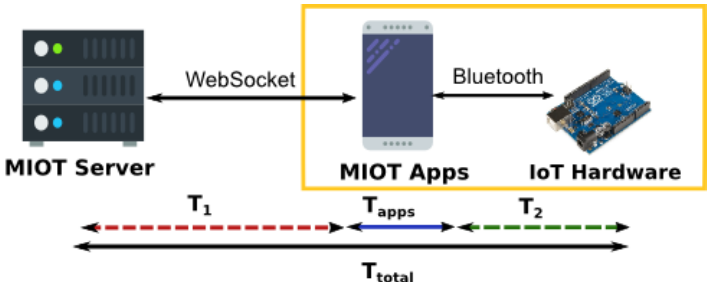


Fig. 6. Latency Measurement using MIOT Framework

Using 2 seconds interval between each measurement, and 626 times packet, the result of the experiment is shown in Table 1. Based on the table, MIOT Apps have estimated latency $T_{Apps} \approx 18.326$ ms. It should be noted that the calculated T_{Apps} includes measurement of two-way communication. It should also be noted that different experiment (e.g., different Smartphone or network) may result in different result.

Table 1. Latency Measurement for MIOT Framework

Measurement	N data	Average (ms)	Standard Deviation (ms)
T_{total}	626	146.67	44.78
T_1	525	60.506	37.39
T_2	727	67.838	19.48
TApps (T_{total} - T_1 - T_2)		18.326	

In the second experiment, Common Approach, an ESP8266 WiFi module is used as the replacement for MIOT Apps (Smartphone). ESP8266 is a programmable WiFi communication module and commonly found in many IoT system. The architecture of Common Approach system is shown in Fig. 7. The Arduino and ESP8266 are programmed with WebSocket library and use MIOT Data Structure to enable communication with MIOT Server. The latency of data being sent from MIOT Server - ESP8266 - Arduino - ESP8266 - MIOT Server again is calculated and named this as T_{total}. The network latency between MIOT Server and IoT Hardware is measured using standard ping program and named this as T_1. Using T_{total} and T_1, the amount time needed by ESP8266 and Arduino to receive and send back the data can be measured and is named as T_2. The interval for each data is 2 seconds, the same as the previous experiment. The result of the test is shown in Table 2.

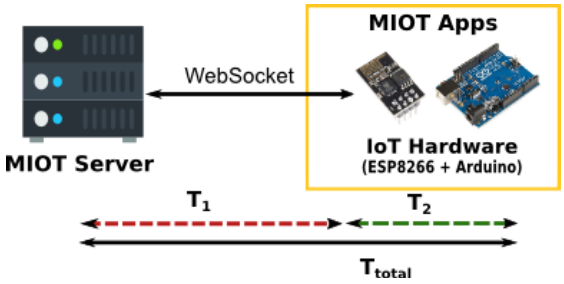


Fig. 7. Latency Measurement using Common Approach

Table 2. Latency Measurement for Common Approach

Measurement	N data	Average (ms)	Standard Deviation (ms)
T_{total}	727	242.87	223.655
T_1	623	4.56	7.556
T_2 (T_{total} - T_1)		**238.31**	

Based on the experiment shown in Table 1 and Table 2 the performance evaluation between MIOT and Common Approach on total latency (T_{total}) is compared. The comparison is shown in Table 3

Table 3. Performance Comparison of MIOT and Typical IoT Approach

Experiment	**Average of Total Latency (T_{total})**
MIOT Approach *(2 seconds interval)*	146.67
Typical Approach *(2 seconds interval)*	242.87

Based on Table 3, it is shown that MIOT Approach has smaller latency than Common Approach concerning the experimental conditions.

6 Discussion

This paper shows that MIOT Framework can be used as best practices in building IoT system. The general purpose-ness of MIOT can be used to build various IoT system, especially which require real-time communication. The use of Smartphone as a pure concept 'Smartphone for the Things' may seem a bit unusual, even exaggerated at first. But on the other hand, several benefits are shown in MIOT Framework. First, it will allow IoT device mobility due to mobile Internet connection features in Smartphone. Second, it will add usability for a non-technical user in the deployment phase. Such scenario is since the IoT hardware communicates on Bluetooth, the communicated data can be configured to target any desired MIOT Server. Thus if there's some change regarding server address, group, or users, the change can be done easily through Smartphone user interface, rather than change the IoT program code. In this term, the communication on IoT devices can be built as generic as possible, which it just emit and receive Bluetooth data. The MIOT Apps which provide IP based communication will then decide which data goes to which server.

Using MOT Framework can reduce time and effort in both developing and implementing. IoT Developers now can focus on developing devices that send or receive data through Bluetooth, or the control interface for the user. In the case of usage scenario in Section 4A, IoT developer will need only to build their devices and (may) modify MIOT controller to display their data. They do not need to build a server program or adding more modules and code for their IoT devices to connect to the Internet. In the case of usage scenario in Section 4B, IoT developers only need to build their IoT devices. All of the rest communication infrastructure already handled by MIOT. This is very handy since IoT devices are often devices with low computational abilities. Saving a few lines of codes can mean much in term of program features, speed, and also power usage.

This might be the reason behind the measurement result in section V. As can be seen in Table 1 and Table 2; the MIOT approach unexpectedly has better latency

compared to Common Approach. This better latency is questionable since there is an additional path (Smartphone) along the communication path. But after seeing the processing time in both experiments (T_{Apps} + T_2 in MIOT, and T_2 in Common Approach), it can be seen that the higher delay in Common Approach is caused by high processing time (T_2). In this case, the much larger code on Arduino and ESP8266 due to IP Communication, WebSocket, and JSON Encapsulation, are suspected to contribute significantly [20]. Please note that this lower latency for MIOT happens in experiments that use Arduino Uno. If more sophisticated IoT hardware is used, e.g., Raspberry Pi, the result may differ. However, further research is needed on this.

As an additional benefit, MIOT Framework will also reduce developer's overhead in learning. Just to be noted that both in usage scenario, there are at least four different system used which are : (1) Build IoT device which is Arduino based. (2) Build Android apps which is Java Android-based. (3) Build NodeJS server which uses JavaScript server based. And (4) build a web page, which uses JavaScript, to connect to the server. Learning those four skills will take much overhead, and make development much slower for a single person. By using MIOT Framework, the learning curve can be cut in half. In this way, it can be said that MIOT Framework could reduce time and effort in building an IoT system.

One may criticize that this Smartphone-based IoT design will add cost to overall IoT system. But on the other hand, it could also be seen that it could save money when the IoT device is used temporarily. Meanwhile, The user can also employ obsolete, unused Smartphone that already outdated in recent technology, but still able to run MIOT Apps. Thus, can help lengthen the operating age of a device. Future potential in this design is seen where Smartphone (MIOT Apps) not only forward data back and forth, but also harness more Smartphone features, mainly computing potential and its sensor and actuator. The Smartphone computing power will sure add more benefit when compared to IoT hardware computing capabilities. Smartphone internal sensor, like GPS and accelerometer, could be used to give information about device location and movement.

The massive amount of Smartphone available in society will sure also increase the IoT adaptability. Its availability will come in handy in case the gateway system needs to be replaced. It should be noted that sensors and actuators are already massive, but not the gateway system. Rather than custom build IoT relay communication system - which might be hard to find and be replaced -, the replacement of Smartphone is just as easy as buying new ones, in any mobile store. Overall, the Smartphone as used in MIOT Framework will sure open more research opportunities and could increase IoT adaptability.

In this research, Bluetooth is used as the main communication interface between IoT hardware and Smartphone. Bluetooth is chosen for several reasons. First, it is already one mean of popular communication method. Second, in many IoT prototyping hardware (like Arduino), programming with Bluetooth is simpler and require less code.

One may argue that the use of WebSocket and JSON in MIOT framework is already common in IoT research. Indeed, those technologies are chosen as the primary framework element to increase adaptability by users due to their familiarity. Web-

Socket is already proven in web-based real-time communication. The underlying TCP protocol used in WebSocket ensure data and communication reliability (prevent any data loss). The research in [21] even shows that WebSocket outperforms more popular IoT-specific protocol (e.g., CoAp and MQTT) in term of performance and energy consumption. JSON is an already familiar data structuring that is human readable, efficient, and also programmable friendly. JSON is used so that any IoT developer can extend or modify the framework for their needs easily.

6.1 Smartphone Ping Latency Anomaly

Observing on T_1 in Table 1 and Table 2, something intriguing is found. The ping latency (T_1) from MIOT Server to Smartphone is much higher compared with T_1 in Table 2. It is 60.506 ms vs. 4.56 ms. The ping latency in Table 1 also has very high jitter. This condition is questionable since, in both experiments, T_1 is just a ping time between MIOT Server and IoT device and considering that both experiments performed on the same controlled network.

To check whether this is not caused network problem, the same test as in Section 5A is repeated, and yet the result is still consistent with Table 1. As another comparison, a ping test between two laptop computer is performed in the same network, and it shows small ping latency similar to T_1 in Table 2. This test eliminates the possibilities of a network problem. To check the possibilities of malfunctioned Smartphone, another Smartphone is used, and the experiment was repeated, yet the result is still consistent with Table 1, high latency with high jitter, This means that this very high ping latency is not caused by the Smartphone problem.

Curious about this ‘anomaly' a deeper reference is needed. Surprisingly there are only a few references mentioned about this high and unstable ping latency in Smartphone's WiFi. One notable lead is the discussion forum in [22]. Some users found similar high latency when they ping their Smartphone. Some user report that this high latency did not happen when the Smartphone acts as a tethering device or when Smartphone is doing active network activity such as browsing or downloading files. Another user in the forum "conclude" that it might be caused by Smartphone power saving features to make the WiFI network idle when not heavily in use, thus contributing to higher time to ‘wake' the WiFi.

One notable reference to these power saving features is found in research by Elnashar and El-Saidny [23]. They discuss such power saving features that called Discontinuous Reception (D/RX) and mainly involve user inactivity timer (UIT). These features (D/RX and UIT) are mainly researched regarding mobile cellular networks like 3G and LTE. However, the experiment conducted in this research -which use WiFi- seems to have similar behavior.

Based on those reasons, T_1 and even overall latency might be reduced if the test is performed when Smartphone WiFi is in the active state, or when the interval in the test is much reduced. It seems that 2 seconds interval for each packet is considered beyond the Smartphone's WiFi UIT constraint, thus make the WiFi idle and increase the latency. However, further research is needed to confirm this statement.

6.2 Some Consideration

Despite its features, and easiness MIOT Framework can bring, there is three consideration that should be taken into consideration.

Latency. Putting a Smartphone between IoT and server give more simplicity. However, it also adds more latency. Embedding WebSocket technology, rather than the request-response system like polling and long-polling, could help reduce the latency of communication [24]. However, the addition of Smartphone (MIOT Apps) as communication gateway will add more latency. Based on the experiment on Section V, it is shown that MIOT Apps contribute to ≈ 13.145 ms latency. Such number may seem huge compared to a typical switch or router, which usually serves in metrics of a microsecond. It is comprehensible since Smartphone are not designed to function like switch/router. But on the other hand, by looking at how the number compared to the total time needed to propagate between server to the device, 13.145 ms can be considered as quite insignificant. Although the degree of significance sure will depend on the system design and requirement.

Security. MIOT Framework is built with a bare minimum configuration to ease the deployment. As a trade-off, current version of MIOT Framework still did not equip with much security features. It means anyone who knows the MIOT Server address, port number, and also the name of the group will be able to listen or send data to the group. To address this problem, the name of the group should be set as if it is a secret phrase so that those only know the phrase will be able to tap or listen to the data. And last, as the other alternative, IoT developers may and should modify both MIOT server, or MIOT Controller to adapt to their security needs.

Concurrency. MIOT Server process incoming packet based on first come first serve mechanism. It does not consider the time stamp value that embedded in the message. MIOT Server does not care whether the later message has earlier time stamp than the current processed message, it will process the first message that comes. This concurrency consideration should be noted especially when data integrity is a key factor in the system.

7 Conclusion and Future Works

MIOT Framework is a promising framework and can be used as best practices in building IoT system. The general-purpose, simple configuration, bi-directional real-time communication, and Smartphone based are features that can ease and speed up IoT development and deployment. MIOT Framework can reduce time, effort, and learning overhead. MIOT framework is an example where the Smartphone can be helpful for developing an IoT system by acting as service gateway between IoT device and Internet. MIOT should be used as one of ‘best practices' to develop IoT system especially for the researcher with few technical capabilities or those who need a quick and working system available. Particularly for a system that shows control and communication between a user and IoT devices in real-time.

A vast applicability of MIOT Framework is seen in the future where it could take place to speed up IoT development, both in research or deployment phase. MIOT

Framework should also be developed to harness the Smartphone power, especially Smartphone sensors and or actuator. Further research is also required to see the comparison performance and trade-off cost between this approach and more another common approach.

8 References

[1] L. Atzori, A. Iera and G. Morabito, "The Internet of Things: A Survey," Computer Networks, vol. 54, no. 15, pp. 2787-2805, 2010. https://doi.org/10.1016/j.comnet.2010.05.010

[2] A. Zanella, N. Bui, A. Castellani, L. Vangelista and M. Zorzi, "Internet of Things for Smart Cities,"IEEE Internet of Things Journal, vol. 1, no. 1, pp. 22-32, 2014. https://doi.org/10.1109/JIOT.2014.2306328

[3] J. Stankovic, "Research Directions for the Internet of Things," IEEE Internet of Things Journal, vol. 1, no. 1, pp. 3-9, 2014. https://doi.org/10.1109/JIOT.2014.2312291

[4] M. Baker, "What is a Software Framework? And why should you like 'em?", CImetrix, 2016. [Online]. Available: http://info.cimetrix.com/blog/bid/22339/What-is-a-Software-Framework-And-why-should-you-like-em. [Accessed: 13- Apr- 2016].

[5] A. Kamilaris and A. Pitsillides, "Mobile Phone Computing and the Internet of Things: A Survey," IEEE Internet of Things Journal, vol. 3, no. 6, pp. 885-898, 2016. https://doi.org/10.1109/JIOT.2016.2600569

[6] R. Golchay, F. Le Mouël, S. Frénot and J. Ponge, "Towards Bridging IoT and Cloud Services: Proposing Smartphones as are Mobile and Autonomic Service Gateways," UbiMob, pp. 45-48, 2011.

[7] Zachariah, T., Klugman, N., Campbell, B., Adkins, J., Jackson, N. and Dutta, P. (2015). The Internet of Things Has a Gateway Problem. Proceedings of the 16th International Workshop on Mobile Computing Systems and Applications - HotMobile '15. https://doi.org/10.1145/2699343.2699344

[8] R. Want, B. Schilit and S. Jenson, "Enabling the Internet of Things," Computer, vol. 48, no. 1, pp. 28-35, 2015. https://doi.org/10.1109/MC.2015.12

[9] Soukup, Paul A., S. J. (2015). Smartphones. Communication Research Trends, 34(4).

[10] "Gartner Says Global Devices Shipments to Grow 2.8 Percent in 2015", Gartner Newsroom, 2015. [Online]. Available: http://www.gartner.com/newsroom/id/3010017. [Accessed: 28- Dec- 2016].

[11] R. Rawassizadeh, B. Price and M. Petre, "Wearables," Communications of the ACM, vol. 58, no. 1, pp. 45-47, 2014. https://doi.org/10.1145/2629633

[12] "RFC 6455 - The WebSocket Protocol", Tools.ietf.org, 2016. [Online]. Available: https://tools.ietf.org/html/rfc6455. [Accessed: 25- Jul- 2016].

[13] M. Gray, "Spheres: A Web Services Framework for Smartphone Sensing as a Service," 2015 9th International Conference on Next Generation Mobile Applications, Services and Technologies, 2015. https://doi.org/10.1109/NGMAST.2015.36

[14] Y. Zhang, G. Zhang, J. Wang, S. Sun, S. Si and T. Yang, "Real-time information capturing and integration framework of the Internet of Manufacturing Things," International Journal of Computer Integrated Manufacturing, vol. 28, no. 8, pp. 811-822, 2014. https://doi.org/10.1080/0951192X.2014.900874

[15] J. Li, Y. Zhang, Y. Chen, K. Nagaraja, S. Li and D. Raychaudhuri, "A Mobile Phone-Based WSN Infrastructure for IoT over Future Internet Architecture," 2013 IEEE International Conference on Green Computing and Communications and IEEE Internet of Things

and IEEE Cyber, Physical and Social Computing, 2013 https://doi.org/10.1109/GreenCom-iThings-CPSCom.2013.89
[16] Node Foundation, "Node.js", Nodejs.org, 2016. [Online]. Available: https://nodejs.org/en/. [Accessed: 25- Jul- 2016].
[17] Ecma International, "The JSON Data Interchange Format, 2013", 2013.
[18] "EventBus for Android™ by greenrobot", Greenrobot.github.io, 2016. [Online]. Available: https://greenrobot.github.io/EventBus/. [Accessed: 02- Nov- 2016].
[19] "Arduino - Home", Arduino.cc, 2016. [Online]. Available: https://www.arduino.cc/. Accessed: 26- Jul- 2016].
[20] "ESP8266WebServer to send larger amounts of data - Everything ESP8266", Esp8266.com, 2017. [Online]. Available: http://www.esp8266.com/viewtopic.php?f=29&t=2417. [Accessed: 19- Jan- 2017].
[21] D. Mun, M. Dinh and Y. Kwon, "An Assessment of Internet of Things Protocols for Resource-Constrained Applications," 2016 IEEE 40th Annual Computer Software and Applications Conference (COMPSAC), 2016
[22] "iPhone Wifi high latency in ping times," MacRumors Forums, 2017. [Online]. Available: http://forums.macrumors.com/threads/iphone-WiFI-high-latency-in-ping-times.477141/. [Accessed: 16- Jan- 2017].
[23] A. Elnashar and M. El-Saidny, "Extending the Battery Life of Smartphones and Tablets: A Practical Approach to Optimizing the LTE Network," IEEE Vehicular Technology Magazine, vol. 9, no. 2, pp. 38-49, 2014. https://doi.org/10.1109/MVT.2014.2311571
[24] V. Pimentel and B. Nickerson, "Communicating and Displaying Real-Time Data with WebSocket,"IEEE Internet Computing, vol. 16, no. 4, pp. 45-53, 2012 https://doi.org/10.1109/MIC.2012.64

9 Authors

Toni Tegar Sahidi is with the Research Group of Information-Centric Network, Faculty of Computer Science, University of Brawijaya, Malang, INDONESIA (email: sahidi@student.ub.ac.id).

Achmad Basuki is with the Research Group of Information-Centric Network, lecturer in Network major, Faculty of Computer Science, University of Brawijaya, Malang, INDONESIA (email: abazh@ub.ac.id).

Herman Tolle is with the Research Group of Multimedia, Game & Mobile Technology, lecturer in Mobile, Multimedia, & Game major, Faculty of Computer Science, University of Brawijaya, Malang, INDONESIA (email: emang@ub.ac.id)

Article submitted 21 June 2017. Resubmitted 11 November 2017. Final acceptance 05 February 2018. Final version published as submitted by the authors.