Production Site Visualization Using Mobile Applications

Authors: *Yasuhiro Hara** and *Takahiro Kaneko**

1. Introduction

The GOT2000 Series (hereinafter “GOT”) provides a lineup of industrial touch panel interfaces developed based on an “Easy & Flexible” concept. These touch panels have garnered tremendous praise for connecting to a variety of FA devices to visualize the entirety of a production system. The market responded well to GOT Mobile (Fig. 1) as a remote Mitsubishi Electric GOT solution that can monitor and operate equipment connected on production sites remotely from computers, tablets, and other mobile terminals. This favorable reception has increased the demand for the Company to develop additional functionality.

However, GOT Mobile must always have a web browser running, otherwise users cannot quickly identify any on-site issues remotely from a computer or mobile terminal.

To provide even faster remote notifications about the status of production sites, Mitsubishi Electric released Pocket Mobile as a mobile app compatible with GOT Mobile in May 2021 so that users can not only receive notifications about alarms triggered on production sites

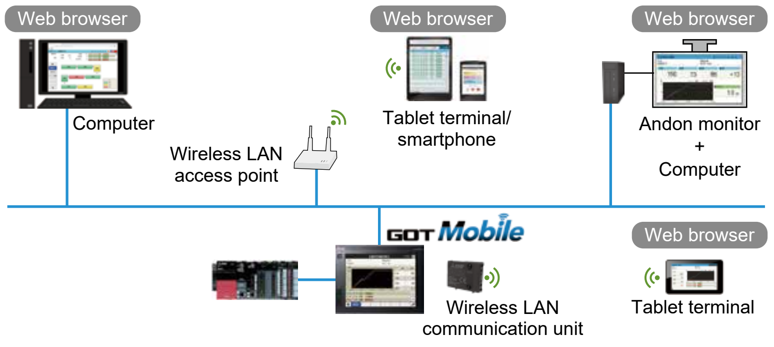

Fig. 1 GOT Mobile system configuration

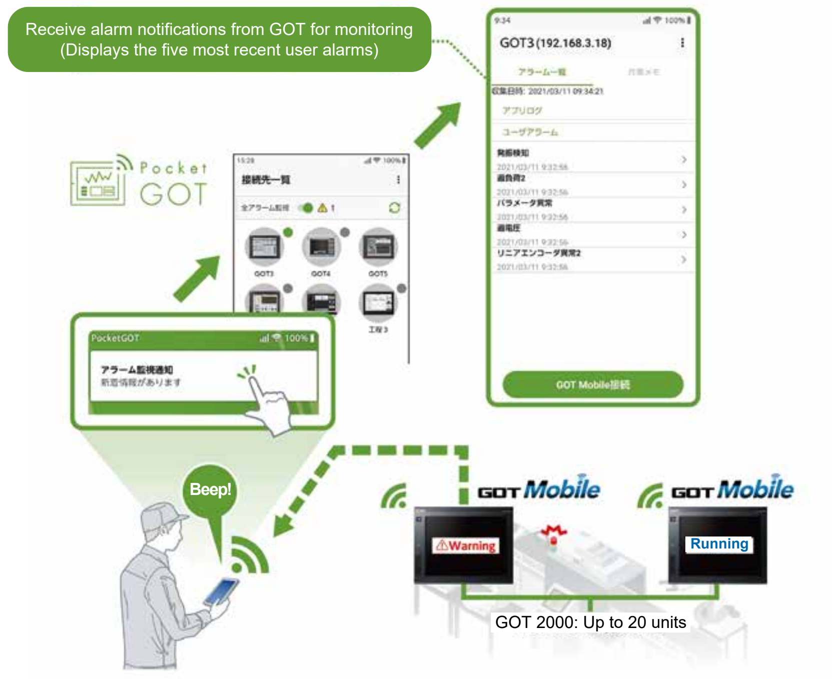

Fig. 2 Remote error identification

**Nagoya Works*

but also remotely monitor and operate equipment from a computer or mobile terminal.

This paper describes the Pocket GOT features as well as the technologies and innovations made to realize those functions.

2. Pocket GOT Features

2.1 Remote error notifications

Pocket GOT is a free Android*1 mobile app available on the Google Play store. Pocket GOT can notify remote users about alarms triggered on a production site sent from an on-site GOT via a mobile terminal using sound, vibrations, and warning banners. Therefore, operators no longer need to have a web browser running on their mobile terminal to quickly learn about any on-site production problems (Fig. 2).

*1 Android is a registered trademark of Google LLC.

2.2 Verification of equipment statuses and alarm logs

Users can display a dedicated GOT Mobile screen by launching a web browser using GOT Mobile from the Pocket GOT app linked to the Mitsubishi Electric GOT Mobile remote solution. In this screen, users can view alarm logs and on-site information related to alarm notifications via a remote mobile terminal (Fig. 3).

2.3 Consolidated working memo display

Users can send working memos (daily and other reports) that include pictures created on a mobile terminal with Pocket GOT installed to the GOT. FA Application Package iQ Monozukuri Process Remote Monitoring(1) makes system configuration without any expert knowledge easy while enabling a shift to IoT production equipment. By using this FA application package, users can aggregate working memos from the GOTs installed on each line of the production site. This enables employees to display all of the working memos in a form format on an office computer as well as share pictures, images, and the registered text information (Fig. 4).

3. Pocket GOT Development

3.1 Application configuration

Pocket GOT runs as three different types of mobile applications: a native application that runs independently as an application installed on a device, a web application that runs in a web browser without requiring any installation, and a hybrid application that does require installation but runs the main functions in a web browser.

Pocket GOT must always run as a background process on mobile terminals to receive alarms triggered on production sites on those mobile terminals. Because the web and hybrid applications that run in a web browser

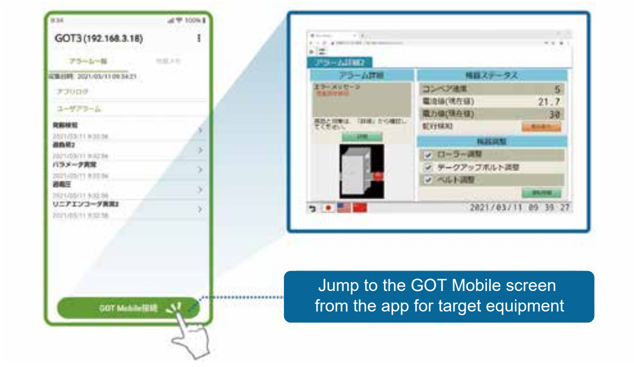
GOT3 (192.168.3.18)
Jump to the GOT Mobile screen from the app for target equipment

Fig. 3 System status and alarm history

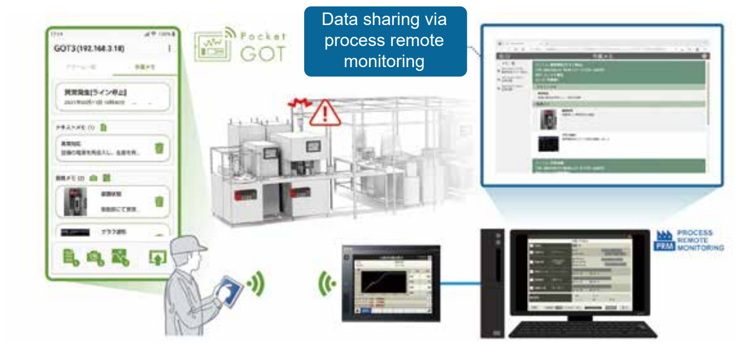
Data sharing via process remote monitoring
Pocket GOT
GOT3 (192.168.3.18)
PROCESS REMOTE MONITORING

Fig. 4 Collective display of working memos

cannot run functions requiring the application to run in the background, Mitsubishi Electric adopted a native application with no limitations on background functions (Table 1).

3.2 Development environment

Users need support for platform-agnostic environments (iOS*2/Android) on mobile terminals used to install Pocket GOT. Therefore, Mitsubishi Electric adopted Xamarin*3—Microsoft's cross-platform development environment—as the development environment for Pocket GOT to take advantage of Visual C#*3 and Visual Studio*3 knowledge and the success of its other products.

*2 iOS is a registered trademark of Cisco Technology, Inc.
*3 Visual C#, Visual Studio, and Xamarin are registered trademarks of the Microsoft Corporation.

4. Technologies and Innovations to Realize Functionality

4.1 System configuration

This section describes the Pocket GOT system configuration. Pocket GOT consists of an alarm notification function to remotely notify users of abnormalities at production sites (Fig. 5[1]) and a working memo function to create inspection reports about production sites (Fig. 5[2]). The alarm notification function transmits data via the HyperText Transfer Protocol (HTTP) for server-client communications using the same port. The proven communications library built into GOT Mobile sends alarm requests via an HTTP connection (Fig. 5[3]). GOT Mobile operates as a function to receive these requests and notify users about the details of any alarms (Fig. 5[4]). The working memo function operates as a feature to send data that consists of text and images recorded on production sites to the GOTs using the File Transfer Protocol (FTP) (Fig. 5[5]).

Any user who has installed the FA Application iQ Monozukuri Process Remote Monitoring (Fig. 5[6]) to collect and visual data from multiple systems on a computer can acquire working memo data from GOTs (Fig. 5[7]) and collectively display all of the working memos in the HyperText Markup Language (HTML) format (Fig. 5[8]).

4.2 Alarm notification methods

There are two alarm notification methods: local notifications to display notifications issued internally by the mobile application on a mobile terminal and online notifications to receive and display notifications issued from a cloud server via an internet connection on a mobile terminal. Local notifications benefit from minimal

Table 1 Comparison of Mobile Application Format

Selection point	Native application	Web application	Hybrid application	Priority
Functional limitations of terminal*1	No limitations	Limitations	Some limitations	High
Network connection	Not required	Required	Sometimes required	↑
System Configuration	Simple configuration	Complex configuration	Complex configuration	
Operational stability	High	Platform dependent	Platform dependent in some circumstances	
Universality of terminal	Low	High	Low	
Universality of program	Low	High	High	Low

*1 Camera, communication, and other functions

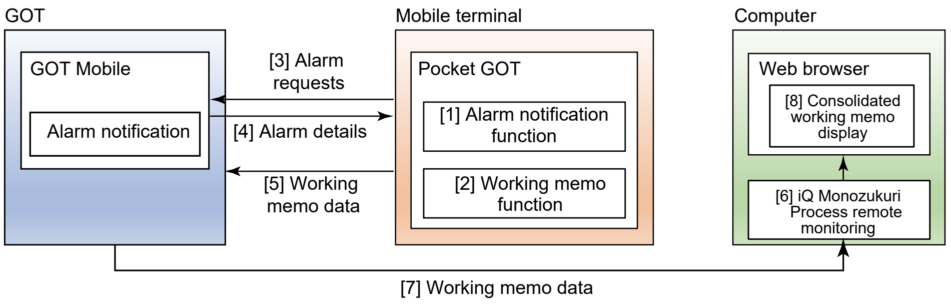

Fig. 5 Pocket GOT system configuration

notification functions to offline terminals but suffer from a high-level of complexity to ensure operation during the sleep state. Online notifications benefit from a simple implementation of notifications via background processes, but suffer from an inability to provide notifications to offline terminals without an internet connection in addition to the complex configuration of the notification server (Fig. 6).

Pocket GOT can send notifications to offline terminals and incorporate methods to realize local notifications that are easy to implement. This sequence sends requests for the status of alarms to the GOTs. The GOT receives this alarm data and generates an internal Pocket GOT notification if there is a new alarm (Fig. 7).

4.3 Segmentation of communication processes for mobile terminals in sleep mode

Pocket GOT runs as a foreground and a background application when the mobile terminal is in an active state. As a foreground application, the mobile terminal prioritizes the operation of the mobile app, which in turn has essentially no operational limitations. As a background application, the mobile application runs but the mobile app screen is not displayed. In this state, there are operational limitations to the mobile app because the mobile app has a lower operational priority than other applications that are displayed on screen. The mobile terminal also severely limits the operations of the mobile app if in sleep mode.

Taking into account the various risks resulting from these operational limitations, Pocket GOT utilizes a system that operates as a foreground process even when running in the background or when in the sleep mode. However, the mobile terminal does have an issue of delays in processing regular alarm requests from Pocket GOT to the GOTs.

Therefore, Pocket GOT receives data for only the latest alarm from a GOT for alarm requests sent to the GOTs. By changing the internal process to determine the necessity of a notification process comparing the current alarm with the previous alarm data, Pocket GOT reduces processing in the sleep mode and retains the periodicity. This segmentation of the communication process realizes a framework that can regularly execute

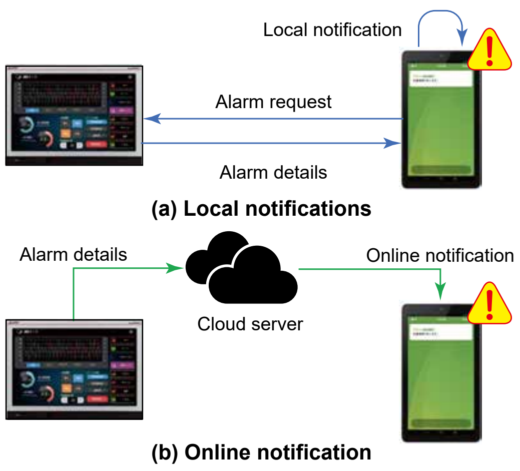

Fig. 6 Local and online notifications

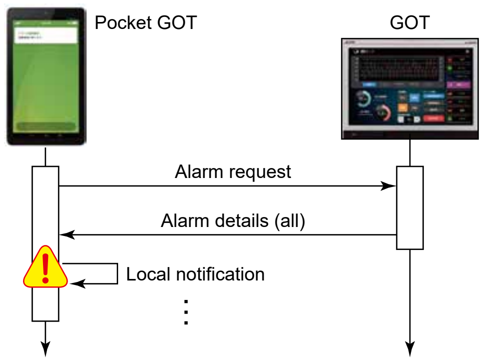

Fig. 7 Sequence of local notifications

Pocket GOT processes in the background, even when in sleep mode (Fig. 8).

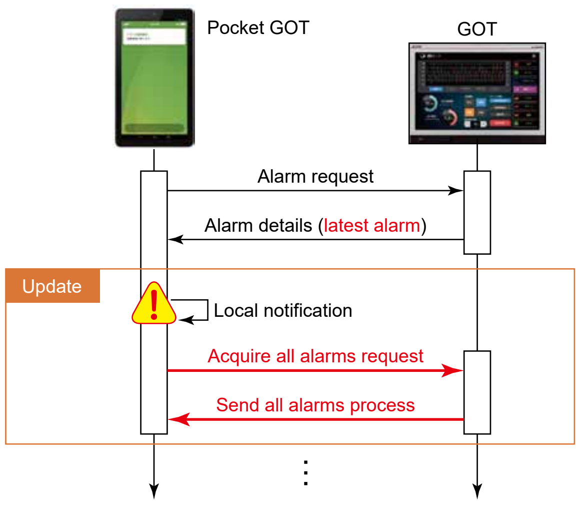

Fig. 8 Segmentation of communication processes

5. Conclusion

This paper has described the ways in which manufacturers can easily adopt the Mobile App Pocket GOT as a simple and low-cost solution that more rapidly visualizes any on-site issues that arise through links to the GOT2000 series, GOT Mobile, and iQ Monozukuri Process Remote Monitoring.

In the future, Mitsubishi Electric will pursue further usability and broader functionality for users.

Reference

(1) Okojima, S., et al. : FA Application Package “iQ Monozukuri PROCESS REMOTE MONITORING,” Mitsubishi Denki Giho, 94, No. 4, 236-239 (2020)