M^2G: A Monitor of Monitoring Systems with Ground Truth Validation Features for Research-oriented Residential Applications

Meiyi Ma*, Ridwan Alam†, Brooke Bell§,
Kayla de la Haye‡, Donna Spruijt-Metz§, John Lach† and John Stankovic*
*Department of Computer Science, University of Virginia, Charlottesville, VA, USA
†Department of Electrical and Computer Engineering, University of Virginia, Charlottesville, VA, USA
‡Keck School of Medicine, University of Southern California, Los Angeles, CA, USA
§USC Center for Economic and Social Research, Los Angeles, CA, USA
Email: *{meiyi, stankovic}@virginia.edu, †{ridwan, jlach}@virginia.edu, ‡{delahaye}@usc.edu §{brookebe,dmetz}@usc.edu

Abstract—**Research in the area of internet-of-things, cyber-physical-systems, and smart health often employ sensor systems at residences for continuous monitoring. Such research-oriented residential monitoring systems (RRMSs) usually face two major challenges, long-term reliable operation management and validation of system functionality with minimal human effort. Targeting these two challenges, this paper describes a monitor of monitoring systems with ground-truth validation capabilities, M^2G. It consists of two subsystems, the Monitor^2 system and the Ground-truth validation system. The Monitor^2 system encapsulates a flexible set of general-purpose components to monitor the operation and connectivity of heterogeneous sensor devices (e.g. smart watches, smart phones, microphones, beacons, etc.), a local base-station, as well as a cloud server. It provides a user-friendly interface and supports different types of RRMSs in various contexts. The system also features a ground truth validation system to support obtaining ground truth in the field. Additionally, customized alerts can be sent to remote administrators and other personnel to report any dysfunction or inaccuracy of the system in real time. M^2G is applied to three very different case studies: the M2FED system which monitors family eating dynamics [1], an in-home wireless sensing system for monitoring nighttime agitation [2], and the BESI system which monitors behavioral and environmental parameters to predict health events and to provide interventions [3]. The results indicate that M^2G is a comprehensive system that (i) requires small cost in time and effort to adapt to an existing RRMS, (ii) provides reliable data collection and reduction in data loss by detecting faults in real-time, and (iii) provides a convenient and timely ground truth validation facility.**

Index Terms—**Residential Monitoring System, Ground Truth Validation, Reliability, Fault Monitoring**

I. Introduction

With the advent of various sensing, computation, and communication technologies, we have many systems aiming at 24/7 residential monitoring for physiological, psychological, behavioral, environmental, and social information. In the areas of IoT, cyber-physical-systems, smart homes, and smart health, many systems have been developed by researchers for residential monitoring to detect activities of daily living (ADL), monitor health status or home environments, identify circadian activity rhythms (CAR), etc. Most of these research-oriented residential monitoring systems (RRMSs) need to be deployed in real homes for weeks or even months. However, once the system is deployed, to ensure reliable data collection, someone needs to keep monitoring the deployed monitoring system. Hardware faults, software crashes, network disconnections, human interference, and many other reasons can result in the deployed system being partly or completely inoperative with accompanying loss of valuable data. In many of these systems, once the system is deployed in a home, it is either not monitored, or a system administrator manually remote monitors the application monitoring system. In addition, often there is limited or no support to obtain ground truth in real-time during the deployment phase. This leads to two common, but non-trivial classes of challenges for an RRMS:

- How can an RRMS detect that the deployed system is operational in real-time and react to minimize the loss of data? Can a general system be developed to provide a comprehensive monitoring capability with minimum effort and be applicable to many types of RRMSs?
- How can an RRMS obtain ground truth during the deployment period without using after-the-fact surveys, which are prone to subjective errors, and without using intrusive devices such as cameras?

A few systems have addressed these challenges, but in narrow application specific scopes and are not comprehensive enough for other systems to adopt [4], [5], [6]. For example, monitoring and validation of sensing systems have been well studied for safety related autonomous systems, such as nuclear power plants, aircraft, trains, medical, power plants and chemical plants. The primary purpose of these systems is to find undesired or not permitted process states and to take appropriate actions to maintain the operation of the system and to avoid damage or accidents. These types of monitoring systems have very high safety requirements and thus are very expensive. Another example, are the monitoring systems for large-scale distributed systems, network and services aimed toward fault detection and diagnosis, decision support, and maintaining and optimizing the overall system performance. These monitoring strategies are designed particularly for optimizing the network and are not well suited to monitor other components (e.g. devices, the base station) found in RRMS. Furthermore, these monitoring approaches are very specific to the intended application and often do not support run-time ground truth collection.

With these considerations in mind, M^2G is designed to feature the following characteristics:

- *Generality*: M^2G is a general package which can be adopted by any RRMS with any architecture and for any application with minimal effort and time. All the parameters for monitoring and ground truth validation are adjustable by users via user-friendly interfaces.
- *Comprehensive*: This system monitors devices, servers, connectivity, and software running on any component. And for different components, it can employ different strategies for validation.
- *User-friendly*: M^2G can be used by both researchers and, once deployed, by residents without any knowledge of the underlying monitoring techniques.
- *Real-time*: Monitoring and validation can be provided in real time against various faults, violations, and loss of data. Also ground truth validations can be performed in a timely manner.

M^2G is a user-friendly, real-time, and automated system for operation monitoring and system ground truth validation of RRMSs. It is comprised of two subsystems, a monitoring system and a ground truth validation system. The monitoring system *Monitor*^{2} incorporates a comprehensive list of monitoring tasks to ensure that an RRMS is running properly and to quickly inform host system administrators of any (potential) errors. In order to increase the accuracy of any detection or inference made by the RRMS, the validation system with *Ground Truth Validation* and *System Operation Validation* is designed to supplement *Monitor*^{2}. This can help researchers obtain the ground truth and improve the application in run-time. To the best of our knowledge, M^2G is the first reusable monitoring and validation system designed particularly for RRMSs.

The major **contributions** of this paper are: a general software framework that can easily be added to RRMSs, thereby relieving every new RRMS researcher from developing their own application-specific monitoring and validation system; a demonstration of the its value by applying it to multiple RRMSs with very little time and code modification; showing how it prevents RRMSs from losing data; providing a real-time ground truth facility to help improve application performance; and making the monitoring and validation system open and accessible to the community.

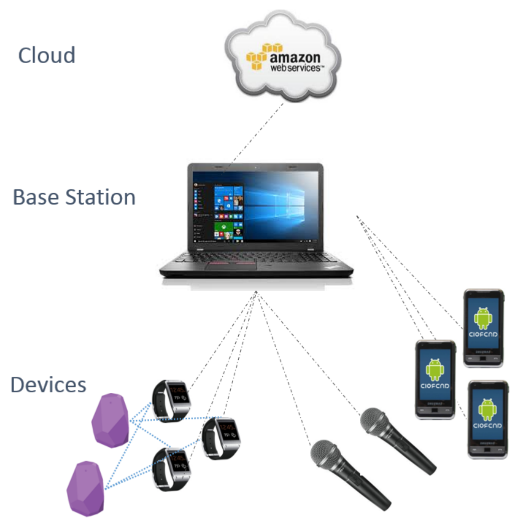


Fig. 1: Major Components of Residential Monitoring Systems

II. Overview of Research-oriented Residential Monitoring Systems

A research-oriented residential monitoring system (RRMS) usually consists of three layers: (i) sensors and actuators continuously collecting data or performing some actions in the residence or toward the residents; (ii) a base-station running programs or acting as a local server to process the data for detection, prediction, or inference; and (iii) a cloud server to store, process, and display the data to remote users or clients (See Figure 1). The connectivity in RRMSs usually lies between devices, devices and base-station, base-station and cloud, and sometimes directly between devices and cloud server. Devices communicate with other devices or the base station using interfaces such as Wi-Fi, Bluetooth, Z-wave, X-10, Serial or USB ports. They may also communicate with the cloud directly using a network interface. Meanwhile, the base station or the cloud acquires data from devices using either API modules or directly accessing ports of the program.

For instance, one of the case studies in this paper, Monitoring and Modeling Family Eating Dynamics (M2FED) [1], consists of multiple smart watches, smart phones, microphones, beacons, a laptop as the base station, and a cloud server. Apps run in the smart watches and smart phones to collect the data of users' arm movement and usages of the phone, respectively. Three server programs run in the base station to receive the data from the phones, the watches, and microphones. The main program runs continuously in the base station to process the data for detecting eating, speaker ID, and mood, as well as to upload the data from different sub-systems to the cloud. Upon detecting certain circumstances at the base station, queries are sent to users to obtain their subjective inputs and to act as reminders. Note that many RRMSs are built with a similar architecture or slight variants of it. M^2G is designed to operate in such systems to add reliability and validation in performance.

III. *Monitor*^{2} SYSTEM

To monitor if an RRMS is operating properly, a comprehensive monitoring of the application level monitoring system, *Monitor*^{2} is designed. *Monitor*^{2} provides a user-friendly interface to customize monitor settings (see Figure 2), where users can choose components to monitor and customize monitor options such as the frequency, the file path, the database type, the cloud URL and the notification email addresses. In addition, there is a real-time monitor display to show the status of the system, a monitor log to record all results and an alarm system to inform users of any anomalies. Monitor components are distributed at three places, the **base station** (the laptop), **devices** and the **cloud** to check the status of the hardware, the software and the connectivity between them.

A. Base Station

In order to understand if the base station is operating properly, the minimal requirements for the base station monitoring include (i) the power is plugged in or the battery has enough power, (ii) the disc memory should be enough to store the data, (iii) RRMS's processes are running, and (iv) data are uploaded (if applicable).

Focusing on the above requirements, the power, disc memory, processes, and data are monitored by *Monitor*^{2} on the base station continuously. Users choose which components to monitor and set up the configuration shown in Figure 2.

1) Power: When the system is deployed in a real home, the charger of the laptop may be disconnected. As a result, the base station will shut down and even lose all the data. Monitoring the power and battery level of the laptop is an important component.

To monitor the power with *Monitor*^{2} see (1) in Figure 2, users set up the monitor frequency of the laptop battery and add it to the monitor list. The power of the base station is monitored through the charger (plugged in or not) and the battery level. If the charger is unplugged and the battery level is lower than 80%, a notification email is sent to the researcher, who will remind the user to charge the laptop.

2) Disc Memory: Some sensors such as microphones and cameras can consume a significant amount of disc memory. It is often valuable to monitor the usage of disc memory and alert researchers ahead of time if the base station is about to run out of disc memory. Similar to setting up the power monitor, users can choose the monitor frequency for disc memory and add it to the monitor list. A notification is sent if the remaining disc memory is lower than 10%.

3) Processes: RRMS processes, such as algorithms detecting user behaviors and programs receiving and uploading the data usually run continuously on the base station. Monitoring that processes are alive is an important way to check the status of RRMSs.

Monitor^2 is able to monitor different types of processes such as .exe, .jar, .py, etc. each with its own monitoring frequency. Users can add as many as processes to monitor via (3) in Figure 2. Once *Monitor*^{2} detects that any monitored process has stopped, it notifies the researcher immediately. For advanced setup, users can also set processes to be restarted automatically.

4) Data: In RRMS, sensors usually collect and upload the data to the base station using files or a database. Monitoring the uploading status of these files/database is an indirect way to check the status of devices.

Users can customize data monitoring at (4) and (6) in Figure 2, which are for file checking and database checking, respectively. File checking requires file name, path, and frequency. Any number of files can be monitored each with their own frequency. *Monitor*^{2} provides monitor functions for three common databases (SQL Sever, MySQL and SQLite). Users can choose one of them and fill in the information of the database and tables and add it to the monitor list.

B. Devices

There are usually two types of devices, smart devices (e.g. smartphones and smart watches), and standard sensors (e.g. a temperature sensor, a contact sensor) in RRMSs. To check if a device is working properly, the minimal requirements include (i) the device is on and has enough power to support it over a required period, (ii) the apps (if any exist) in the devices work properly with enough memory, and (iii) the data is collected and uploaded properly.

In the file monitor section, we described how to monitor requirement (iii). For (i) and (ii), *Monitor*^{2} develops apps for smartphones and smart watches (with an Android OS) to monitor the power usage and processes. Apps also check and upload battery status files to the base station. By checking the battery status files, *Monitor*^{2} sends notifications when the battery levels of devices are low. Configuration is similar to the file monitoring, as shown at (4) in Figure 2.

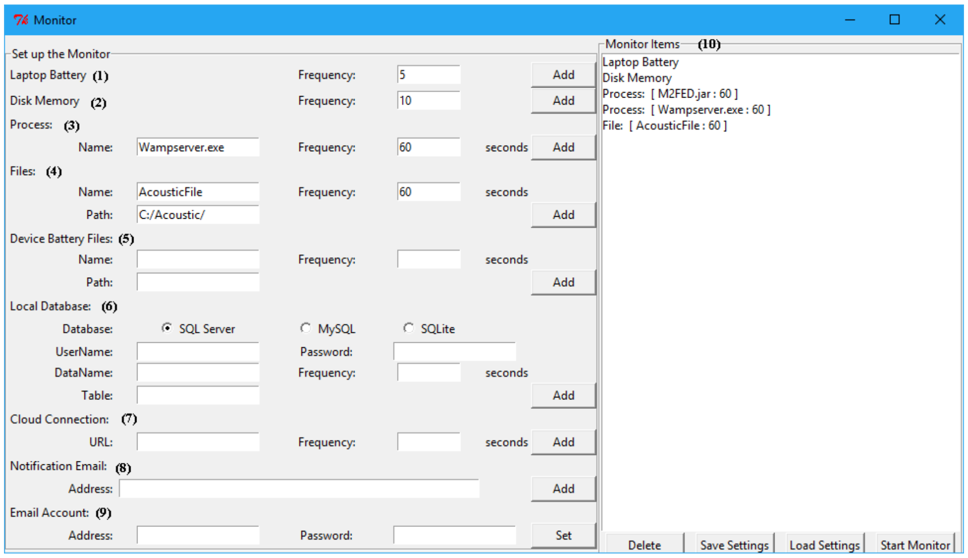
C. Cloud

Key potential errors in the cloud include (i) access denied, (ii) EC2 stops working, and (iii) uploading fails. Targeting these problems, three strategies are used to monitor the cloud.

To start with, in Figure 2 users add the cloud URL and its monitoring frequency to the monitor list. (i) The laptop sends a Ping including a query to the cloud, which goes to the scripts and then to the database in the cloud. Then it sends back to the base station a message indicating the situation in the cloud. If the connection breaks, the base station sends a notification to the user. (ii) Monitoring programs running in EC2 ensures that EC2 instances work well. For example, Amazon cloud provides the subscription of EC2 monitoring, which can be used directly here. (iii) The code is injected to the receiving data scripts in the cloud, which can check if the data is uploaded successfully. If not, it sends an email to the researcher directly.

D. Notifications

As mentioned above, *Monitor*^{2} sends email notifications of anomalies. (8) and (9) in Figure 2 are used to set email addresses to receive and send notifications. Users can add multiple emails to receive notifications and set up one email to send.



Laptop Battery (1) Frequency: 5 Add
Disk Memory (2) Frequency: 10 Add
Process: (3)
Name: Wampserver.exe Frequency: 60 seconds Add
Files: (4)
Name: AcousticFile Frequency: 60 seconds
Path: C:/Acoustic/ Add
Device Battery Files: (5)
Name: Frequency: seconds
Path: Add
Local Database: (6)
Database: [✓ SQL Server] MySQL SQLite
UserName: Password:
DataName: Frequency: seconds
Table: Add
Cloud Connection: (7)
URL: Frequency: seconds Add
Notification Email: (8)
Address: Add
Email Account: (9)
Address: Password: Set

Laptop Battery
Disk Memory
Process: [M2FED.jar : 60]
Process: [Wampserver.exe : 60]
File: [AcousticFile : 60]

Fig. 2: M^2G configuration interface on the laptop

IV. Ground Truth Validation System

The Ground truth validation system of M^2G includes two parts, *Ground Truth Validation* and *Data Validation*. Ground Truth Validation is a *human-in-the-loop ground truth collection* system with a server to generate and send questions with pre-defined logic and an app running on the smart phone to interact with users. In this process, M^2G also collects *reinforcement ground truth* to help researchers find the reasons for a false alarm. *Notifications* are triggered to alert researchers in a timely manner.

A. Human-in-the-loop Ground Truth Collection

A server with a friendly GUI and associated instructions is provided for users to set up their questions. The server is programmed in PHP, but it can be requested using URL request easily from a browser or by other programs with any language. The example request format is *https:// localhost/ ema/ema.php?q=*{*"id": "123", "c": "startsurvey", "suid": "1", "server": "http://192.168.0.100/ema/ema.php", "androidid": "922b94ecca15ed9d", "alarm":"true"*} . Questions can be generated either right after the event happens or at a scheduled time depending on the context of the RRMS. After the request is sent, the user receives the question on the app on their phones. For example, for an eating monitoring system, a ground truth query can be triggered immediately when the algorithm detects an eating event. The question can be set as "Are you eating now? Yes/No". Or it can also be generated every evening with question like "Did you eat at 8:00 a.m. this morning?". If the answer is yes, the eating detection algorithm is accurate for this time. Otherwise, it is a false detection. In this way, the accuracy of the algorithm over some deployment period is obtained.

B. Reinforcement Ground Truth

Besides validating if events happen or not, we can also obtain related information of the event as a reinforcement to the ground truth. In the case of false detections, the researcher also would like to know the reason for the false detection, which may be more valuable than the false detection itself. M^2G supports more personalized queries based on the user's answer. Following the above example, when the user answers "no" to the eating query, further questions can be issued such as "What were you doing? (Drinking water? Smoking? Brushing Teeth?)". In this way, researchers know which action causes a false detection and can improve the detection algorithm accordingly.

C. Notification

The ground truth is stored in the MySQL database in the cloud, so that researchers can log in and look at it at any time. Moreover, the notification system alerts researchers when a false detection is found.

D. Data Validation

Besides ground truth validation, M^2G also provides a basic, but essential data validation for the RRMSs. After receiving the data from sensors at the base station, while monitoring the data file, M^2G provides various types of checkers to validate the data. Users choose the proper data validation checker for their files. Nine checkers and examples of how to use them are shown in Table I. These data checkers indicate the abnormal nature of the data and alert users as needed, but are not necessarily errors.

V. Evaluation

M^2G is evaluated by applying it to three significantly different RRMSs: (i) Monitoring and Modeling Family Eating Dynamics (M2FED) which supports complicated sets of devices, processing and interventions for multiple family members, (ii) an in-home sensing system for Monitoring Nighttime Agitation and Incontinence [2] which directly addresses a medical issue and is used only at night, and (iii) a Behavioral Environmental Sensing and Intervention (BESI) [3] which is

TABLE I: List of Data Validation Checkers

Checker	Description	Example
Existence	If no data is collected successfully, some systems do not record and upload any data while some others record it as *NaN* or *Null*, which cannot be observed without reading the data.	The data collected from bed sensor (pressure pad) shows NaN.
Range	There is a threshold of the valid value for a sensor. If the data recorded is out of range, M^2G will report it to the researcher.	Room temperature should be in the range of (60 - 85).
Same Value	If the data recorded the same value all the time, it indicates potential faults with sensors or not being used by users.	Sensors such as microphones, accelerator usually are very sensitive and do not sense the same value all the time while using.
Character checks	Data may show in a wrong field in a data file during the collection and uploading processes. It checks if only expected characters are present in a field.	The filed recording the room temperature should only contains numbers.
Batch totals	Checking for missing records or counting the number of updating times.	For event-based data collection, there is no routine of data updating, but some of them has the number of updating. In these cases, check the number of updating at the end of the day.
Check digits	Validating the number of digits for some data that consists of a fixed number of digits.	Device ID, e.g. Android ID of the smartphone and watch.
Format check	Checking that the data is in a specified format (template).	Dates have to be in the format DD/MM/YYYY.
Uniqueness check	Checking that each value is unique.	For data that requires unique ID, such as event ID, check if repeated number is used.
Consistency check	Checking across fields to ensure data in these fields corresponds	If activity detected is cooking, then location should be in the kitchen.

decentralized and senses behavioral activities using wearables and monitors environmental parameters with in-home sensors. The wide diversity of these three systems demonstrate the generalizability and effectiveness of M^2G.

Further, to show the utility of M^2G at both early development stages and during real deployments, it is installed and evaluated in both pre-deployment and real deployment situations for 15 days and 7 days, respectively. The incontinence detection system was deployed in 13 patients homes for about 2 weeks each in 2014 without a comprehensive monitor and ground truth verification system. In this paper we describe how M^2G would work in that system by emulating the system using the real deployment data. BESI was deployed in real residences of dementia patients.

In each of these three case studies, M^2G is incorporated with the set of monitoring tasks shown in Table II and evaluated by the metrics described in the next section.

A. Metrics

We evaluate the performance of M^2G and show how it helps to improve RRMSs using the metrics: (a) deployment time, (b) amount of code modified, (c) the number of errors detected by different monitor components, and (d) the amount of the data saved for the system comparing to the situation of not using M^2G. To be noted, the results may vary when people with different background deploy a system. In this paper, all deployments are conducted by knowledgeable graduate students. For ground truth assessment we provide a qualitative description of M^2G's value for each of the case studies.

TABLE II: Monitor List for M2FED, Incontinence and BESI

Component	Monitor	M2FED	Incontinence	BESI
Phone	Battery Monitor	✓		
	App Monitor	✓		
Watch	Battery Monitor	✓	✓	✓
Base Station	Battery Monitor	✓	✓	✓
	Process Monitor	✓	✓	✓
	Disc Memory Monitor	✓	✓	✓
	Database Monitor	✓	✓	
	File Monitor	✓	✓	✓
Cloud	Connection Monitor	✓	✓	
	EC2 Monitor	✓	✓	
	File Uploading Monitor	✓	✓	
	Database Uploading Monitor	✓		

B. Operational Steps

Typically, deploying an RRMS is complex and time-consuming. However, integrating M^2G into the system is straightforward by following these steps.

Step 1: Simply list the monitoring tasks required of the RRMS by using the GUI tool, see table II.

Step 2: Set up monitoring and validation plans (e.g., frequency, etc.) following the GUI interface, as shown in Figure 2. (For advanced configurations, modify the code if needed.)

Step 3: Install the apps to smartphones and watches if needed.

Step 4: Install the ground truth validation system into smartphones and the laptop. Set up questions to obtain the ground truth and enforcement information in the server and the query type (i.e. immediate event triggered or periodical query).

Step 5: Start to monitor the system.

C. M^2G for M2FED

M^2G is applied to M2FED [1] in both a pre-deployment and a real deployment, which last for 15 days (4/15/2017 - 4/30/2017) and 7 days (7/5/2017 - 7/11/2017), respectively.

1) Installation Time and Code Modification for Both Scenarios: For both pre- and real deployments, the installation time and code modification metrics are the same. First, we do not need to modify code for applying it to M2FED. The installation time is 40 minutes, which is used for installing apps to the devices and setting up the monitor components.

2) Performance - Pre-deployment: In pre-deployment the M2FED system is not completely debugged and many errors are detected and reported. Some observations in using M^2G for pre-deployment are:

- Devices: Battery levels are monitored all day. M^2G sends notifications of the battery level in the morning every day and also when the levels are lower than 20%. Within 15 days, there were 13 days that watches and phones were not turned on in the morning, emails were sent to researchers, who notified users to turn them on. Also, battery levels were lower than 20% for 5 times on phone

1, which was detected and notified immediately. Without M^2G, in the worst case, 13 days of the watch data may have been lost if the system was only checked manually once per day.
- Base Station: As shown in Table 1, five components are monitored on the base station. In day 14, the laptop was unplugged without it being noticed for 2 hours. Users were notified and restored the power of the laptop in time before losing any data. Otherwise, it would have shut down and the operation of the whole system would have ceased. As a result, M^2G saved 10 hours of the data of all components running on the laptop. In addition, five processes errors were detected on average daily when six processes are running in total. By telling researchers the time and frequency of the process crashes, it helps them to improve the stability of programs. Meanwhile, file and database monitoring only records the file updating status without notification, because they are event-based data collections. After the experiments, researchers use the log to check the data uploading time with the event time to examine if the component works well. For example, there were 5 times when a user uses the phone but no data is uploaded to the server during that time. We found the connection issue of the app through the log. Also, there were 6 times when an eating event is detected, but no EMA data is updated in the database, which helps find an errors.
- Cloud: Most of the time the connection of the cloud worked well. Connection errors happened 52 times, but all were recovered quickly. Because it is a pre-deployment, it did not upload much data to the cloud. However, the log of cloud connection and Internet status helps researchers to select a better uploading time and frequency for the real deployment.

Overall, M^2G played a very important role in the pre-deployment of M2FED. It helped finding problems in the system and saving a large amount of human effort and time. For example, it detects that smartphones and watches consume power quickly and cannot last for a day without charging, which informed researchers that they need to improve the design to save additional energy in the smart devices.

3) Performance - Actual Deployment: The second deployment applying M^2G to M2FED is a real residential deployment, where the system is deployed in a home with three family members for 7 days. Results are summarized in Table III, which includes the errors detected for each component with M^2G and the amount of data saved comparing with the system without M^2G. Here we assume the system without M^2G is manually checked either just once, twice and three times per day at a fixed time. Figure 3 shows the daily results.

In this deployment, the system was checked with a frequency of 30 min. If an error happens and was not fixed in time, the same error will be detected in the next time interval. In this case, when counting the number of errors, if it happens multiple times in consecutive intervals it is counted as 1 error.

- Comparing three the major parts of the system, i.e., devices, the base station and the cloud, devices have the largest number of errors, which basically happens every day. These errors are caused by the low power in the morning (i.e., users don't charge the phone or watch the previous night) or during the day (e.g., energy is consumed and the device is about to turn off). After receiving the notification, alerts notify the user in time to charge their devices. As a result, it saves up to 24-hours of device data compared with a manually method.
- On the base station, to start with, the laptop ran out of disk memory for 5 times because it keeps storing large amounts of acoustic raw data. Without a timely notice, it could lose many hours of data. In M2FED, files are uploaded to the base station based on events, for example, eating files are only uploaded when watches detect the user is eating. Therefore, it is hard to differentiate an error or no event when no file is uploaded. M^2G only provides the latest status of the file without any error message in this case. However, in other deployments, if files are generated or uploaded with a fixed frequency or timestamp, M^2G can also detect file errors and save the related data.
- A cloud error only happened once and recovered by itself quickly, which did not cause losing data. However, in some deployments, if there is a real time usage of the data on the cloud, a monitor is also very important.
- Daily results are shown in Figure 3. The number of errors increases over time, especially for the base station. There is only 1 error on the first day, but 5 errors on the last day.

4) Ground Truth Validation: M^2G applies an EMA survey system to M2FED to obtain the ground truth of algorithms, including eating detection and mood detection. When an eating event or mood event are detected, an EMA is sent to the corresponding user's phone immediately. The questions include (1) "were you just eating" and (if not) (2) "what did you do if not eating". Not only the ground truth of true or false, but also the real event for false positive assessments can be obtained. Meanwhile, if a mood is detected, user receives an EMA asking if they were happy (neutral, or sad, angry).

In the real deployment, M2FED detected 5 eating events and

TABLE III: Errors detected of each component with M^2G and the amount of data saved comparing with the system without M^2G (Assuming the system is manually checked once, twice and three times)

	Component	Error	once (h)	twice (h)	three times (h)
Device	Phone (*2)	14	336	168	112
	Watch (*2)	14	336	168	112
Base Station	Power (*1)	0	0	0	0
	Disk Memory (*1)	5	240	60	40
	Processes (*6)	2	48	24	16
	Files (*7)	1	24	12	8
	Databases (*1)	5	240	60	40
Cloud	Connectivity (*1)	1	24	12	8
	EC2 (*1)	0	0	0	0

2 of them were confirmed by the user by answering the EMA. The other three were caused by other gestures like waving the watch. "Happy" moods were detected 21 times and 15 of them were confirmed. Without M^2G, these events cannot be validated in real time. With such short activities, it would be prone to error in a recall based system, especially asking them to remember the exact time of the event.

D. M^2G for Incontinence

The incontinence system is designed to detect incontinence and its relationship to sleep agitation. The system consists of three main layers, i.e. the sensing layer, the base station, and a cloud-based web server with an associated database. In the sensing layer, there are two smart watches monitoring sleep agitation, a wetness sensor to measure incontinence, and an audio system for speech patterns of patients at night. There is also a two layer layer simple monitoring module in the base station and cloud to detect failure. It was deployed in 13 real families from 5/8/2014 to 8/6/2014 (each patient participated for 1 to 2 weeks during this period).

The monitoring module only logged the states of the laptop (including the system memory, battery, and connectivity to the Internet) and the status of the cloud, and only sent notification to the user when the laptop power is low. From the data obtained from the real deployments, the simple monitoring module is not effective enough. There were still days of losing data because of late detection of the problem. Also, the ground truth (i.e., the wetness event) was collected by caregivers every morning, which took extra time.

Therefore, via emulation we installed M^2G into the incontinence project, which helped the system detect the errors in a timely manner and thus save a large amount of the data. The results are shown in Table IV, which compare the performance from the real deployment (with a simple monitoring module) and the emulated with M^2G, including the number of errors detected and the amount of the data lost per hour (here we are ignoring the time it takes to fix the error).

1) Installation Time and Code Modification: To implement M^2G into the incontinence application with the components listed in Table II, and following the steps in Section V.B, the total modification and deployment time is about 40 min.

2) Performance: In the emulation, three types of errors are detected. With a best case analysis, assuming that a researcher

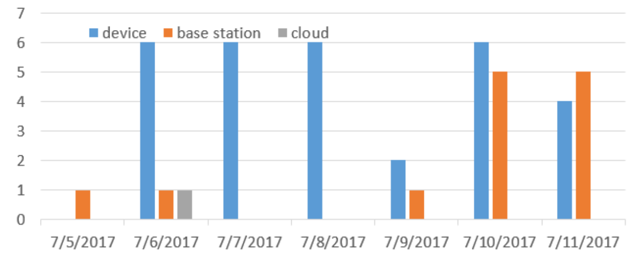


Fig. 3: Number of errors detected per day (1/5/2017 - 7/11/2017)

TABLE IV: Errors detected of each component with M^2G and the amount of data saved comparing with the system without M^2G on Incontinence Project

	Component	Error	Real data lost without M^2G (h)	Data lost emulated with M^2G (h)
Devices	Left Wrist	5	59	0 - 2.5
	Right Wrist	4	47	0 - 2
	Acoustic	0	0	0
	Wetness	0	0	0
	Bed-motion	5	118	2.5
Base Station	Home Controller	0	0	0
Cloud	Connectivity	49	0	0

fixes the problem immediately after the error is notified, we evaluated M^2G through the amount of the data lost.

- The smart watch data. Without M^2G, there is no monitoring of the smart watch in the deployment. As a result, 106 hours (left hand 59 hours and right hand 47 hours) of watch data was lost. For example, Participant 002 lost two days of watch data because the watch had no power. First and third-night data of Participant 004, and the first-night data of Participant 007 were lost because the watches were not worn properly. Both errors can be detected by M^2G and users can be notified immediately. The lost data due to the power issue can be completely avoided with a quick response because M^2G starts to alert users when the power is lower than 20%. Abnormal data can also be noticed by M^2G and as a result it detects the second error quickly. As a result, in best case (users charged the watch when they receive the notification), no data is lost, i.e. M^2G saved all 106 hours of the watch data.
- The bed sensor data. Without M^2G, the system lost 118 hours of bed sensor data because there is no timely notification of losing that sensor data. Take an example of Participant 007, who had 14 days of real deployment and the bed sensor data was lost for 3.5 days (25%). This error is quickly detected by the data validation component of M^2G. With a frequency of 30 min monitoring, these errors can be fixed in 2.5 hours, which is a significant improvement comparing with 118 hours.
- The connection to the Cloud. M^2G detects 49 cloud connectivity errors, which were not noticed by the original system. Although the connection recovered soon and no data was lost in this deployment, it is still a potential serious issue if there is a real time display in the cloud. In some cases when the data is too large, the laptop will delete the data after uploading or daily, and losing the cloud connectivity may cause losing data.

3) Ground Truth Validation: In the original project, ground truth is obtained by the caregiver by daily visits. This can be handled by M^2G, which can save at least 1 hour for the caregiver per day plus travel time.

E. M^2G for BESI

The BESI system is comprised of in-situ environmental sensors and on-body behavioral sensors [3]. A smart watch with an inertial sensor provides behavioral information. Temperature, barometric pressure, humidity, luminosity, and acoustic sensors are used as environmental sensors and are grouped together with a microcomputer (Beaglebone Black) in a room-level relay node. A base-station (laptop) is used for the purpose of remote access and system health monitoring. The BESI system also incorporates a tablet based android app and a second smart watch. All these components are connected as a distributed network. This system has been deployed at residences of dementia patients with a goal to empower caregivers by preventing agitation episodes [7]. The system incorporates many fault tolerance techniques to prevent data loss including on-node parallel sensing processes, watchdogs, and heartbeat messages to the base-station. These techniques address various hardware, software, and network faults which can be handled by automated restarting of the relevant processes [3]. Still there are many more faults that often go undetected and cause the system to lose data until the system is manually monitored at the next scheduled time. In such scenarios, an automated monitoring and notification system like M^2G is useful. M^2G helps to detect faults faster, notifies system maintenance to take action immediately, thus reducing data loss. In the BESI system, all sensors as well as the wearable device are connected with the room level relay nodes, and all the relay nodes send heartbeat messages about the sensor data to the base-station. These heartbeat messages are stored in organized file structures. *Monitor*^{2} reads these files periodically, keeps the log, and sends alerts if no or an erroneous heartbeat is received from any relay or sensor. The validation subsystem of M^2G helps detect the erroneous data from the heartbeat messages and allows reduction in fault recovery time and data loss.

1) Installation Time and Code Modification: The time needed for incorporating the M^2G system to the BESI system was minimal. The only modification required was related to sensing modality specific validation scheme implementation and assigning the specific location of the related files and directories. The total effort costed about 10 hours, for code change metric, it required about 50 lines of code to be modified. These metrics demonstrate that the M^2G system can be easily incorporated in a system with a distributed network architecture and with both in-situ and wearable sensors.

2) Performance: The BESI system has been deployed without the M^2G system for 7 month-long deployments at residences of dementia patients [3], [7]. During those deployments, the system was manually monitored by system administrators "twice-a-day", i.e. at 12 hours interval. For such manual monitoring, the administrator had to go over all the recent heartbeat messages to find possible errors and data inconsistency. This technique not only took time to monitor the system, but also requires expertise in the system functionality to find out possible faults. The average time to detect a fault depends on the frequency of monitoring, which can be up to 12 hours for the "twice-a-day" schedule. This often led to large amount of data loss. After incorporating the M^2G solution with the system, it was deployed for a month in a semi-controlled environment. During this period, the system administrator didn't monitor the system with a fixed schedule, rather only reacted to the alerts sent by the M^2G system. Also the alert messages contained the type of fault or error, hence the administrator didn't need to manually find the issue and resolve it. For each day of these deployments, we calculated the time required to detect and resolve a fault after occurrence or daily fault period (DFP) and the consequent daily data loss (DDL) and used the maximums of those as metrics for performance evaluation. The results are shown for each device and sensing parameters in Table V.

TABLE V: BESI performance in maximums of daily fault period (DFP) in hours and percentage of daily data loss (DDL) metrics by using M^2G against "Twice-a-day" monitoring.

Device	Parameters	"Twice-a-day"		With M^2G	
		DFP	DDL	DFP	DDL
Relay Node	Temperature	5	20.8	0.5	2.1
	Light	4	16.7	0.16	0.67
	Pressure	7	29.2	0.33	1.4
	Humidity	7	29.2	0.33	1.4
	Noise	5	20.8	0.5	2.1
Base Station	Power	2	8.3	0.25	0
	Disk Memory	0	0	0	0
	Processes	12	20.8	2	0
	Files	12	50	5	20.8
	Network	4	4.2	1	0
Wearable	Battery	3	12.5	0.25	0
	Motion	8	33.3	0.50	2.1

From the table, it is notable that the duration of faults and their consequences on data collection vary with the sources of the faults. For example, any relay sensor fault or smart watch app crash causes immediate data loss and the amount of loss (27%) is proportional to fault period (7 hours). On the other hand, power and memory faults on base-station and wearable have some buffer period to recover before causing any data loss, hence data loss is very low (2%). Different faults may be detected at the same time, but the recovery methods vary among the fault sources, hence the recovery time vary accordingly. For example, analog sensors (temperature and noise) cost more data loss (20%) than digital light sensors (10%). Also human factors impact the recovery time by some amount, as an alert in early morning may not be addressed as immediately as it would have been during day time. Finally, the addition of M^2G to the existing BESI system greatly reduced both fault period and data loss (by 90%).

3) Ground Truth Validation: An android app featuring a daily survey is hosted on the tablet. The caregiver of the dementia patient uses the app to provide detailed observations and behavioral information related to the agitation event. Thus ground truth information about the agitation event is acquired through the app. The caregiver is also provided with a smart watch which runs another app. That watch app facilitates easily marking any event as they occur, rather then going to fill out the tablet app immediately. This additional modality reinforces the validity of the ground truth. But both these approaches are passive and dependent on how the caregiver uses the apps. With M^2G, caregivers are prompted with queries to provide information about the agitation event when it is sensed by the system. This is a proactive approach, and will increase the reliability of the ground truth and reduce the possibility of missing event information.

VI. Related Work

Monitoring and validation systems have been studied and developed for many years and in many contexts. However, we are not aware of any existing general and reusable monitoring and ground truth validation platform for RRMSs.

A. Autonomic Systems

Monitoring has been regarded as a crucial component for autonomic systems, especially safety related systems, such as nuclear power plants, aircraft, and trains [4], [6]. Formal methods are usually applied to verify the system and its safety at design time. In addition, these systems usually monitor various output variables to continuously assess the system's safety. Model-based methods of fault-detection, use input and output signals and apply dynamic process models. These methods are based on parameter estimation, parity equations or state observers. Signal model approaches generate several symptoms indicating the difference between nominal and faulty status [6]. As an example case, procedures for automatic supervision of photovoltaic system (PV) systems is developed in [8]. Often these systems are large-scale and expensive and are not adoptable for RRMSs. Also, these are highly specific for particular applications, and their characteristics and goals are very different from those of RRMSs.

B. Smart Homes and Buildings

In this era of IoT, smart homes and buildings are built to improve residential living conditions through monitoring and affecting the environment. However, there is a limited number of systems that monitor and validate the deployed systems. Existing systems often monitor and optimize parts of the system. For example in [9], an HVAC sensor monitoring system is described which provides automatic fault detection of this specific application for smart buildings. It can automatically select representative features from sensors that are unique and relevant to the faults in a HVAC system. Similar examples may include those systems used in home security and power consumption. But comprehensive solutions for various sensing modalities are essential for RRMSs.

C. Existing RRMSs

Many RRMSs have been developed to monitor a residential environment and people activity [10], [11]. Some examples of these RRMSs include the case studies mentioned this paper [1], [2], [3]. Other typical examples may include systems for elderly population monitoring for health-care purposes [12], family interaction and residential environment monitoring for behavior estimation [13], etc. These systems attempt to monitor environment in residences and behavior of residents using a plethora of sensors. But most of these systems lack the required reliability and robustness for residential long-term deployments [11], [14]. With the improvement of sensors and detection algorithms, the functions of RRMSs are becoming more comprehensive. And continuous operation monitoring and ground truth validation of those systems are becoming more critical for their use in the real world.

VII. Conclusion

M^2G is developed to monitor the operation and verify the ground truth of RRMSs with minimal human effort. It provides a set of monitoring components to monitor the operation of sensor devices, the base station, the cloud and the connectivity between them. Moreover, it obtains ground truth for the verification of algorithms. It sends customized alerts to researchers and caregivers to report the dysfunction and inaccuracy of the system in real time, thereby minimizing loss of application data. Experimental results show its ability to comprehensively monitor three different types of RRMSs.

Acknowledgment

This work was funded, in part, by NSF under grants IIS-1521722, IIS-1418622, CNS-1646470 and CNS-1319302.

References

[1] D. Spruijt-Metz, K. de la Haye, J. Lach, and J. A. Stankovic, "M2FED: Monitoring and modeling family eating dynamics: Poster abstract," in *Proceedings of the 14th ACM Conference on Embedded Network Sensor Systems CD-ROM*. ACM, 2016, pp. 352–353.

[2] J. Gong, K. M. Rose, I. A. Emi, J. P. Specht, E. Hoque, D. Fan, S. R. Dandu, R. F. Dickerson, Y. Perkhounkova, J. Lach *et al.*, "Home wireless sensing system for monitoring nighttime agitation and incontinence in patients with Alzheimer's disease," in *Proceedings of the conference on Wireless Health*. ACM, 2015, p. 5.

[3] R. Alam, J. Dugan, N. Homdee, N. Gandhi, B. Ghaemmaghami, H. Meda, A. Bankole, M. Anderson, J. Gong, T. Smith-Jackson, and J. Lach, "BESI: reliable and heterogeneous sensing and intervention for in-home health applications," in *Proceedings of the IEEE International Conference on Connected Health: Applications, Systems and Engineering Technologies*. IEEE, 2017, in press.

[4] J. Ma and J. Jiang, "Applications of fault detection and diagnosis methods in nuclear power plants: A review," *Progress in nuclear energy*, vol. 53, no. 3, pp. 255–266, 2011.

[5] I. Lee and O. Sokolsky, "Medical cyber physical systems," in *Design Automation Conference, 47th ACM/IEEE*. IEEE, 2010, pp. 743–748.

[6] R. Isermann, *Fault-diagnosis applications: model-based condition monitoring: actuators, drives, machinery, plants, sensors, and fault-tolerant systems*. Springer Science & Business Media, 2011.

[7] R. Alam, J. Gong, M. Hanson, A. Bankole, M. Anderson, T. Smith-Jackson, and J. Lach, "Motion biomarkers for early detection of dementia-related agitation," in *Proceedings of the 1st Workshop on Digital Biomarkers*. ACM, 2017, pp. 15–20.

[8] S. Silvestre, A. Chouder, and E. Karatepe, "Automatic fault detection in grid connected PV systems," *Solar Energy*, vol. 94, pp. 119–127, 2013.

[9] Y. Guo, J. Wall, J. Li, and S. West, "Real-time HVAC sensor monitoring and automatic fault detection system," in *Sensors for Everyday Life*. Springer, 2017, pp. 39–54.

[10] U. Varshney, "Pervasive healthcare and wireless health monitoring," *Mobile Networks and Applications*, vol. 12, no. 2-3, pp. 113–127, 2007.

[11] H. Alemdar and C. Ersoy, "Wireless sensor networks for healthcare: A survey," *Computer Networks*, vol. 54, no. 15, pp. 2688–2710, 2010.

[12] N. K. Suryadevara and S. C. Mukhopadhyay, "Wireless sensor network based home monitoring system for wellness determination of elderly," *IEEE Sensors Journal*, vol. 12, no. 6, pp. 1965–1972, 2012.

[13] C. Bi, G. Xing, T. Hao, J. Huh, W. Peng, and M. Ma, "Familylog: A mobile system for monitoring family mealtime activities," 2017.

[14] T. W. Hnat, V. Srinivasan, J. Lu, T. I. Sookoor, R. Dawson, J. Stankovic, and K. Whitehouse, "The hitchhiker's guide to successful residential sensing deployments," in *Proceedings of the 9th ACM Conference on Embedded Networked Sensor Systems*. ACM, 2011, pp. 232–245.