An IoT System for Autonomous, Continuous, Real-Time Patient Monitoring and Its Application to Pressure Injury Management

Sam Mansfield, Eric Vin, Katia Obraczka
Baskin School of Engineering
University of California, Santa Cruz
smansfie@ucsc.edu, evin@ucsc.edu, katia@soe.ucsc.edu

***Abstract*—In this paper, we introduce PIMAP, an IoT-based system for continuous, real-time patient monitoring that operates in a fully autonomous fashion, i.e. without the need for human intervention. To our knowledge, PIMAP is the first open system that integrates the basic patient monitoring workflow including sensed data collection, storage, analysis, and real-time visualization. PIMAP's open design allows it to easily integrate a variety of sensors (custom and off-the-shelf), analytics, and visualization. Other novel features of PIMAP include its deployment flexibility, i.e., its ability to be deployed in different configurations depending on the specific application needs, setting, and resources, as well as PIMAP's self-profiling and self-tuning capabilities. While PIMAP can be applied to various patient monitoring applications and settings, in this paper we focus on the unsolved problem of preventing pressure ulcers, or pressure injuries. We describe how PIMAP's design addresses autonomous, continuous, real-time operation to sense, store, analyze, and visualize patient data from a variety of off-the-shelf as well as custom sensors. We present our current PIMAP prototype as well as different PIMAP configuration scenarios, e.g. cloud-based or edge-based deployment options. We also evaluate PIMAP's performance under different workloads and demonstrate its use collecting wearable pressure sensor data in real-world scenarios from patients with high risk of forming pressure injuries.**

***Index Terms*—Cloud computing, Software architecture, Digital Health, Electronic healthcare, Internet of Things**

1. Introduction

We present a novel framework for autonomous, continuous, real-time patient monitoring that can be used in different settings, e.g. hospital ICUs, clinics, skilled nursing facilities, and homes. Our research is motivated by our collaboration with UC San Francisco and UC Davis medical researchers and their longstanding work on understanding and treating complex wounds, in particular pressure ulcers, also known as pressure injuries.

Pressure injuries are open, deep wounds that develop from the bone outwards caused by a combination of prolonged closure of capillaries and lymphatic vessels, ischemia, reperfusion, and tissue deformation [1] [2] [3]. They require significant treatment and interventions to avoid further complications such as infection. Pressure injuries remain a complex, unsolved, and elusive problem in healthcare. There is general consensus on when they will form, e.g., prolonged pressure on an area of the body (e.g., the sacrum), lack of movement from the patient, and best practices on how to prevent them from forming, e.g., turning the patient periodically. Pressure injuries are considered "never events", they should never occur in healthcare settings [1] and, as an incentive to reduce their occurrence, cost for their treatment in the U.S. are not reimbursable. Yet, in the U.S., pressure injuries still affect 2.5 million patients a year at a cost of $11 billion [5].

There are two well known problems in pressure injury prevention. One, periodically turning patients, which is the current best practice in healthcare facilities [6], is not trivial. Two, because pressure injuries start their formation from the bone [1], by the time one can visually see the injury, it is past the point of prevention, meaning assessing pressure injury risk is not trivial.

In a recent literature survey on the state-of-the-art in pressure injury prevention [7], it was found that sensor-based patient monitoring is a promising approach to assess risk in an objective and patient-centric fashion. Objective metrics can then be used by healthcare clinicians to focus efforts on the highest-risk patients. However, current patient monitoring solutions are either commercial systems that are limited to what the commercial entity offers and/or are one-off solutions that are not discussed in detail and are not released for use by other researchers. In addition visualization of the sensor data is usually not discussed at all even though this is a key aspect of using the sensed data.

To fill this gap, we designed, tested, and evaluated the Pressure Injury Monitoring And Prevention (PIMAP) system framework. PIMAP targets both medical research and clinical use and it is not meant to be a diagnostic tool, but rather a system to improve quality of care and support clinicians and caregivers.

One of PIMAP's main novel features is its holistic approach to patient monitoring which integrates the basic

1. More than 90% of pressure injuries are a secondary condition, meaning the patient was being treated for a different condition when the pressure injury formed [4].

patient monitoring workflow components, namely: *Sense* which collects sensor data, *Store* which stores sensed data as well as analytics, *Analyze* which processes sensor data using different algorithms, and *Visualize* which visualizes sensed data and resulting analytics.

While PIMAP was originally motivated by pressure injury management, its system model is generalizable to a wide range of patient monitoring use cases. PIMAP can be utilized by researchers to monitor a variety of conditions as it can incorporate new sensors (custom and off-the-shelf), different analytics, while leveraging existing sensing, storage, analysis, and visualization technologies. In addition PIMAP can be deployed in different configurations (e.g., edge and/or cloud) depending on the specific application requirements and settings.

In summary, the power of PIMAP is twofold. One, any new sensing, storage, analytics, and visualization technology can be incorporated without putting the burden on the medical researcher or clinician. Two, PIMAP's ability to operate autonomously and continuously enables clinicians to continuously visualize patient sensing data and analytics in real-time.

The remainder of this paper is organized as follows. In Section 2 we discuss related work and how our contribution fits into existing research. In Section 3, we discuss the motivating factors behind our focus on preventing pressure injuries and additionally demonstrate how PIMAP can be used to stratify patients based on risk of forming a pressure injury in a real-world scenario using data collected in a clinical trial. In Section 4, we discuss PIMAP's design, main functional components. and workflows. Section 5 describes PIMAP's current implementation and in Section 6, we discuss how we evaluated PIMAP. Section 7 concludes the paper with some directions for future work.

2. Related Work

Due to its interdisciplinary nature, our work can be classified under different areas in the research literature including: connected health, wireless body area networks (WBANs), ubiquitous healthcare, remote monitoring, ehealth, patient monitoring, internet of healthcare things (IoHT), mHealth, and telemedicine. In this section, we highlight related works that focus on general patient monitoring frameworks. The survey presented in [7] provides a more complete and detailed description of related work specific to pressure injury prevention.

In order to investigate whether the 6LoWPAN specification, which is an adaptation layer that allows IPv6 packets to be sent over the low-power physical and MAC layer 802.15.4 standard, is able to achieve the necessary throughput to support medical applications, a system consisting of Sensor Units, Patient Unit, Remote Processing Unit, and Server is proposed in [8]. While the paper concludes that 6LoWPAN is able to achieve the necessary throughput for common case medical applications, the proposed setup is not released for use by other researchers and does not include visualization capabilities. Since the system was put together with a specific goal, it is not clear whether it could be extended to accommodate different sensing technologies and analytics.

A system application to store historical common biometric sensor data that can be analyzed offline was proposed to address the need to perform historical analysis on medical sensor data [9]. The framework relies on proprietary medical devices to gather the sensor data in the clinic and Kafka [10] to store the data for historical analysis. The analysis presented was performed by downloading data after collection from the data storage server.

Several smartphone solutions have been proposed in the literature. For example, HealthSense is a smarthphone application developed to facilitate clinical trials [11]. It uses a web portal to setup a trial, enroll participants, create survey questions, compute, and store data. The focus is on a clinical workflow tool that can support both questionnaires as well as common sensor information that can be gathered from a smartphone. Another example is an Android interconnection layer entitled TIROL [12] which was proposed to collect medical sensor data from a myriad of sensors. The paper highlights that no health data protocol or standard has prevailed and typically each vendor has its own method of generating data. TIROL was developed to address this issue and was designed such that any standard can be implemented, but is abstracted by the interconnection layer so that the overarching application is agnostic to the protocol used to gather the data.

A medical sensor data collection application entitled $p^2Health$ uses a smartphone app, ModMedApp, to collect data from vendor servers, Bluetooth or Ant+ sensor devices, and questionnaires. ModMedApp stores the collected data into a cloud-based server that clinicians and patients can interact with [13].

The brain scan community appears to be the furthest ahead in terms of developing open source software to analyze different aspects of brain activity. In one work the authors propose an analysis software, MNE Scan, that can both gather and analyze brain activity data in real time [14]. The authors mention that there is other bran scan software available in addition to their own that is open source and capable of real-time analysis. The software framework is based on sensor plugins, algorithm plugins, and a display manager. The software is able to gather and display data in real-time. We did attempt to adapt MNE Scan to meet the needs of pressure injury monitoring at one point in time, but the software is very specific to brain scan monitoring and also based on our investigation is not network based, meaning the software is run locally on one computer that can physically connect to the brain scan device. There is a plugin per device connection.

Another work in the brain scan community discussed some of the difficulties of connecting data related to neurological analysis [15]. The authors mention that the brain scan community has software that can collect the data, but connecting all of this data across multiple sites is still elusive and a working group to develop an architecture to do so is in development.

Collecting and analyzing ventilator data is another specific application example that is the focus of the work presented in [16]. The framework relies on a very specific workflow and consequently it is not clear how easy it would be to adapt this system to other types of sensor data or other workflows.

As our focus is on pressure injury prevention we have selected three notable systems that show interesting results in reducing pressure injuries using sensor information. However they all use proprietary software. One work used a bed sized array of pressure sensors to continuously read and display pressure information to clinicians and found that patients were more effectively repositioned [17], but the same group conducted a randomized controlled trial using the same sensor and software and found that there was no pressure injury reduction in the group using this sensor [18].

A randomized controlled trial using a wearable inertial measurement unit, a sensor that detects movement, and scheduled turning found that patients using this sensor and associated software had significantly fewer pressure injuries and turning compliance was higher [19]. Turning compliance is how close the clinicians were able to turn patients to a desired periodic schedule.

Our work is complimentary to the existing literature and, to the best of our knowledge, PIMAP is the only open source patient monitoring system that operates autonomously, continuously, and integrates sensing, data collection, storage, data analysis, and visualization in a single system. It allows different sensors, off-the-shelf and custom, to seamlessly connect to the system and can integrate various analytics and visualizations. Additionally, PIMAP is designed to be deployed in both centralized and distributed configurations in order to cater to the needs of different deployment settings, including edge, cloud, and hybrid deployments.

3. Case Study: Pressure Injuries and Real-Time Risk Stratification Using PIMAP

Our main motivation for designing a reliable, easy to use patient monitoring system for the clinic comes from the difficulty in preventing pressure injuries. In this section, we provide some explanation as to why pressure injuries are still a serious problem in healthcare.

Pressure Injuries

Pressure injuries, recently standardized from the term pressure ulcers or decubitus ulcers by the National Pressure Ulcer Advisory Panel [20], are classified colloquially as "never events", meaning they should never occur in health care settings and yet in the U.S. there are over 2.5 million patients affected every year at a cost of $11 billion [5]. More than 90% of pressure injuries are a secondary condition, meaning the patient was being treated for a different condition when the pressure injury formed [4].

Biomechanically, pressure injuries are caused by prolonged pressure to an area of the body. Through a combination of prolonged closure of capillaries and lymphatic vessels, ischemia, reperfusion, and tissue deformation the affected tissue dies [1] [2] [3]. Typically this occurs at the bony prominences, such as the sacrum or heels in a patient lying down. The result is an open wound that descends to the bone, which must be further treated to avoid infection.

Pressure injuries have an impact on quality of life as they cause severe pain, treatments increase discomfort and pain, and impact the social life of the patient [21] [22]. Pressure injuries are generally developed while patients are being treated for a different condition, but the resulting pressure injury can affect treatment options [22].

The current pressure injury standard of care is to evaluate a patient's risk of developing a pressure injury based on the Braden Scale [23] which assigns a risk designation (or ranking) based on a healthcare personnel assessing factors such as the patient's activity, age, nutrition, etc. The Braden scale is well studied [24], but has come into question as it has not proven to be effective at predicting pressure injuries due to its inherent subjectivity in assessing patient risk [24].

The combination of the Braden Scale not being effective at predicting pressure injuries and that additional measures beyond nursing interventions may be needed [25] to reduce pressure ulcer prevalence motivates PIMAP's design as a tool to persistently and autonomously monitor patients and help assess their risk of developing pressure injuries. As proof-of-concept, we show how PIMAP can be used to stratify patients in real time according to their risk of acquiring pressure injuries.

Real-Time Risk Stratification Using PIMAP And Data From A Clinical Trial

Objective Mobility [26] was proposed as an objective metric to quantify a patient's mobility using data from a novel pressure bandage. Mobility or the lack thereof has been associated with the risk a patient has of developing a pressure injury. The novel pressure bandage contains a four by four grid of pressure sensors and was used in a clinical trial to collect data on five patients with high risk of forming pressure injuries based on the Braden Scale [23]. Trial enrollment criteria was that a patient must score a 1 (the lowest score) for activity, mobility, and friction/sheer on the Braden Scale, i.e., patients with the highest risk of developing a pressure ulcer.

We use the pressure bandage data collected during its clinical trials to demonstrate how PIMAP can be used to objectively assess pressure injury risk. We fed the pressure data to PIMAP as if it were being collected in real time. PIMAP then calculated patient Objective Mobility over time and displayed the results in real-time.

While in the original trial five patients were monitored at non-overlapping times, we instead simulated the patients as if they were being monitored simultaneously and having their risk assessed in real-time. The original pressure bandage patient data has gaps when the bandage became disconnected (this is often intentional if the patient needs to be moved to a different location). In order to present data that is closest to reality we ran the experiment for 7 hours as

TABLE 1. OBJECTIVE MOBILITY FOR REAL-TIME RISK STRATIFICATION

A		B		C		D		E		Average	
PID	Movements Per Minute	PID	Movements Per Minute	PID	Movements Per Minute	PID	Movements Per Minute	PID	Movements Per Minute	PID	Movements Per Minute
4	0.05	1	0.00	5	0.03	4	0.00	1	0.00	4	0.10
5	0.09	3	0.13	2	0.25	3	0.05	4	0.01	1	0.12
1	0.16	4	1.68	4	0.28	1	0.07	5	0.25	5	0.19
3	0.19	2	2.41	1	1.03	5	0.42	3	0.40	2	0.27
2	1.22	5	3.02	3	1.46	2	1.06	2	1.57	3	0.32

this was the amount of time that all patients had consistent pressure bandage data.

We present the results in Figure 1. Figures 1a, 1b, 1c, 1d, and 1e display the amount of movement (the higher the spike, the stronger the movement) over the length of the experiment. Figure 1g displays The Movements Per Minute metric for all patients over the length of the experiment. The Movements Per Minute metric is a real-time risk assessment of which patients are moving the least, regardless if the movement was clinic-assisted. The y-axis is inverted such that the patient at the top of the graph is the most at risk of forming a pressure injury as they are making the least amount of movements. As can be seen the risk changes over time and there is no one patient that is always most at risk, which is how the status quo Braden Scale assesses patients. The patient data used had an enrollment criteria that all patients must score a 1 (the lowest score) for activity, mobility, and friction/sheer on the Braden Scale. Even though all patients had similar Braden Scale risk scores our metric is able to further distinguish patients in real-time based on their movement.

In Figure 1g we label five moments in time, labelled A-E. In Table 1 we rank the patients at each respective moment in time based on the Movements Per Minute metric. We demonstrate that a real-time risk assessment can highlight at any moment in time which patient is most at risk. This is invaluable to clinicians in a busy clinical setting as the healthcare staff can focus their efforts on the patients that are most at risk and not waste time blindly rotating a patient periodically that is moving on their own. In addition in Table 1 we highlight the average Movements Per Minute over the length of the experiment by patient. It is clear that this this fixed value does not provide the insight that the risk a patient has of forming a pressure injury changes over time. The real-time assessment that we present does not need clinician interference and can be used in addition to any tools or standard of care.

Figure 1f presents the end to end latency from when the data was sent in this experimental setup to when the data in Figures 1a, 1b, 1c, 1d, and 1e were displayed. These metrics are set with a five sample delay per patient at one sample/s. The average latency of this experiment to display the movement data across all patients is $3.05s$, which includes the time it takes to process the data and the time it takes to send the data over the network. See Table 4 for the configuration and Sections 4, 4.2, 5 to fully understand the configuration.

In an ideal scenario with zero network and processing latency we would anticipate a $2s$ latency from the five sample delay that it takes to calculate and visualize the data. For purposes of explanation let us assume that PIMAP-Analyze-Objective-Mobility, which is described in Section 5.3, receives sample 1 at time $0s$, sample 2 at time $1s$, sample 3 at time $2s$, sample 4 at time $3s$, and sample 5 at time $4s$, at which point the movement metric is created stored, retrieved, and visualized. Retaining the ideal scenario criteria that there is zero network and processing delay sample 1 would have latency of $4s$, sample 2 would have a latency of $3s$, sample 3 would have a latency of $2s$, sample 4 would have a latency of $1s$, and sample 5 would have a latency of $0s$. When we average these results the resulting latency is $2s$. From this experiment we see on average a $1.05s$ network and processing latency.

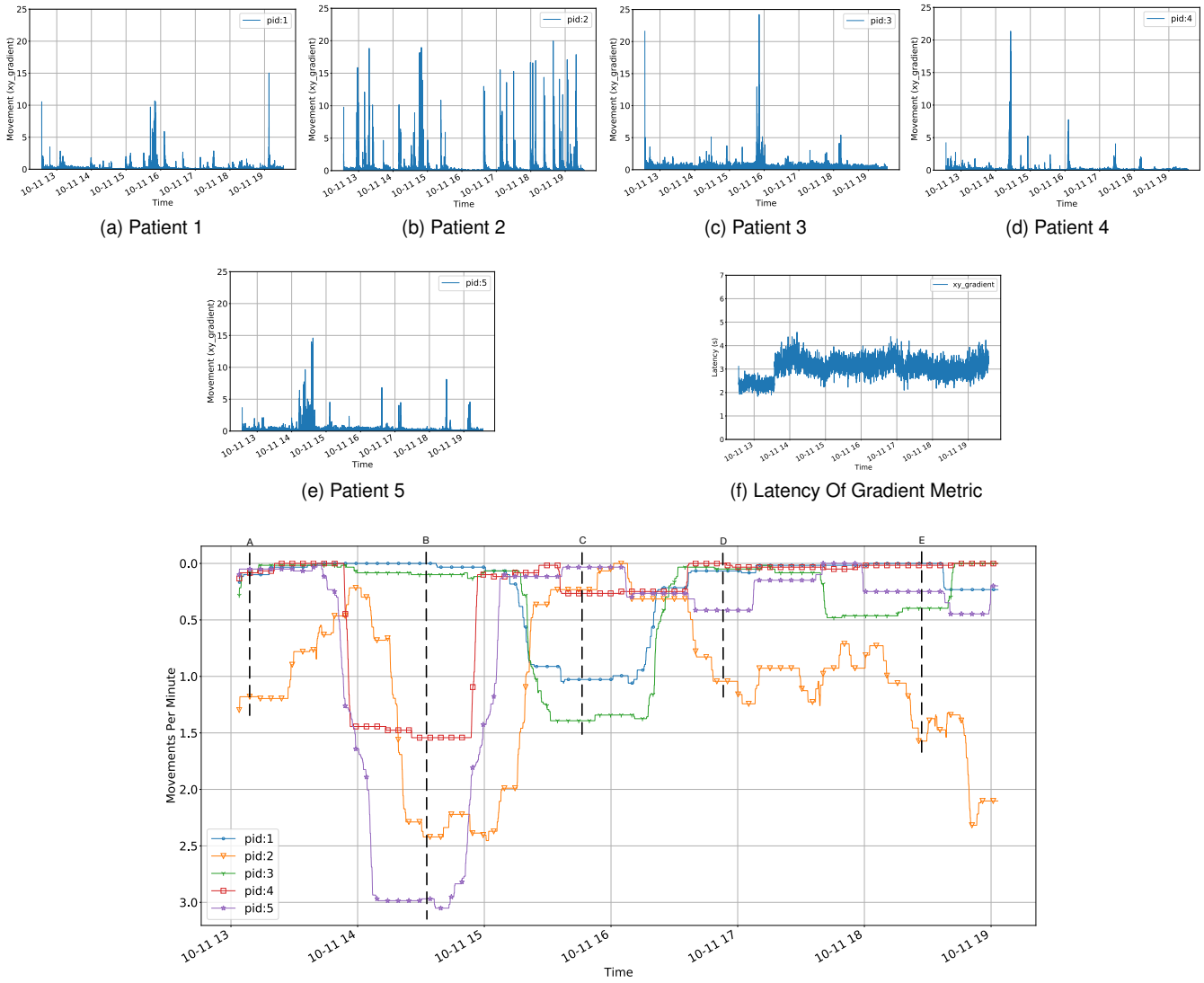
We also see in Figure 1f a clear increase in latency approximately an hour into the experiment as Movements Per Minute metrics begin to be generated. The metric Movements Per Minute is calculated after 3,600 movement metrics per patient are calculated and is a sliding window, so after a 3,600 gradient metric delay (approximately a one hour delay) every new movement metric a new Movement Per Minute metric is calculated.

4. PIMAP Design

There are a wide variety of applications for patient monitoring from services as non-critical as location monitoring to critical applications such as pressure injury monitoring. Nevertheless, PIMAP's premise is that the typical data flow for most patient monitoring applications is essentially the same with a few variations.

Data generally starts from a sensor device, for example a GPS sensor or a pressure sensor, and is sampled periodically. This data is then typically stored somewhere for future analysis. Data is then read from this storage and analyzed as the raw data is often hard to interpret, and then stored back into storage. Finally the analyzed data is read from storage and visualized, which may be for clinicians or for a report to correlate this information with the condition being monitored. This general workflow is illustrated in Figure 2a.

There are variations to this data flow such as not using storage and instead going from the sensed data to analysis and analyzed data to visualization. However, it is generally better practice to store the data for historical analysis. Otherwise this data is lost and the entire experiment must be redone to perform new analysis.



(a) Patient 1 (b) Patient 2 (c) Patient 3 (d) Patient 4

(e) Patient 5 (f) Latency Of Gradient Metric

(g) Movements Per Minute

Figure 1. Objective Mobility Real-Time Risk Stratification

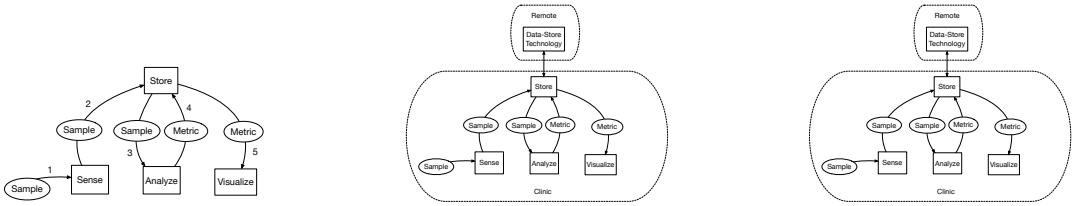
PIMAP is designed with these concepts in mind and concentrates on four components appropriately named based on the general data flow discussed prior: PIMAP-Sense, PIMAP-Store, PIMAP-Analyze, and PIMAP-Visualize. All data passed between components we define to be lists of PIMAP-samples, PIMAP-metrics, and/or PIMAP-commands, which are all self-contained, meaning all data needed to process a PIMAP data type is contained within itself.

Given the sensitive nature of patient data, PIMAP is designed with both security and privacy in mind. Mechanisms to ensure secure and privacy-preserving operation are discussed in Section 4.4.

4.1. PIMAP Data Types

We define three types of data that are passed through the PIMAP system: PIMAP-samples, PIMAP-metrics, and PIMAP-commands. All three pieces of data are self-contained, which means that they can be interpreted on their own and they do not need some sort of exchange or handshake to interpret.

Typically PIMAP-samples contain raw sensor data. For example a pressure sensing device could send data to a PIMAP-Sense component, e.g. via UDP, at which point PIMAP-Sense creates a PIMAP-sample that contains the raw pressure values. A PIMAP-metric is data generated by analyzing PIMAP-samples. A PIMAP-command can be used to actuate on a sensing device, e.g. to change its sampling frequency.



(a) *General patient monitoring data flow*

(b) *Clinic-Sense* data flow scenario

(c) *Cloud-Sense* data flow scenario

Figure 2.

4.2. PIMAP Functional Components

Examining the general data flow in a patient monitoring application we develop four main components as mentioned earlier in Section 4: PIMAP-Sense, PIMAP-Store, PIMAP-Analyze, and PIMAP-Visualize.

In this section we define the four main PIMAP components. Each of these components are abstractions and are not associated with a specific technology. For example PIMAP-Sense could be sensing UDP packets or Bluetooth packets. Our goal is to enable PIMAP to integrate different underlying technologies in a seamless fashion while still maintaining the same structure and functionality.

If it is not entirely clear why we would want to treat these components as abstract objects we will provide an example. The PIMAP-Store is probably the simplest of these objects from an interaction perspective. If you write to it the data is stored and you can read back any data that was previously written. But, in practice a data-store is not trivial to implement, the simplest data-store could be a text file, but it is not immediately clear in what format data should be written and how to retrieve data quickly. What if we want to switch to using a database? By treating the data-store as an object, which at this point in time is a well-established practice, all that we need to know is that we write either PIMAP-samples or PIMAP-metrics and we read PIMAP-samples or PIMAP-metrics. The implementation can change, but the interface will not. To make this even more concrete for the current implementation we rely on Kafka [10], which is a distributed data-store based on the publish-subscribe model, but when interacting with the PIMAP-Store the developer does not need to know how to interact with Kafka as this is all abstracted when using the PIMAP-Store object.

4.2.1. PIMAP-Sense. PIMAP-Sense is how PIMAP interacts with sensor devices. For example if a new novel sensor is created and is sampled via some sort of microcontroller as is typical, in all cases this data must be sent somewhere. One option is to save it locally and then transfer this data to a computer for post-analysis. But, a more favorable practice is to send the data to a computer in real-time over a network or physical wire for real-time analysis. PIMAP-Sense is the component of PIMAP that gathers and/or senses this sensor data.

PIMAP-Sense can accommodate a variety of sensing devices communicating over different network protocol standards. For example an actual implementation of PIMAP-Sense is PIMAP-Sense-UDP, which on each call reads UDP packets that are sent to its interface. One can imagine running a clinical trial with several devices on several different patients sending data via UDP to an endpoint. Using PIMAP PIMAP-Sense-UDP is this endpoint. It is even possible to have many different PIMAP-Sense components running in parallel and in many different locations. In addition PIMAP-Sense can send PIMAP-commands back to the sensor device (this is in exploratory development).

The core interaction of PIMAP-Sense is that a call or query to PIMAP-Sense, PIMAP-Sense.sense() returns a list of PIMAP-samples sent to this endpoint and subsequent calls return any new PIMAP-Samples sent. A PIMAP-command is sent using PIMAP-Sense.send(list of PIMAP-commands).

4.2.2. PIMAP-Store. PIMAP-Store is responsible for data access in the PIMAP system. PIMAP-Store can store PIMAP-samples, PIMAP-metrics, or even PIMAP-commands. To store PIMAP-data only the PIMAP-data itself needs to be provided to PIMAP-Store, but to retrieve data either the $sample\_type$, $metric\_type$, $command\_type$ must be provided. PIMAP itself is basically a publish subscribe model as we are dealing with time series data. So when data is retrieved from PIMAP-Store data is returned in order of the timestamps.

To store data, it is simply PIMAP-Store.store(list of PIMAP-samples/metrics) and to retrieve PIMAP-Store.retrieve($sample\_type$, $metric\_type$, or $command\_type$), which returns a list of PIMAP-samples or PIMAP-metrics respectively. It is not guaranteed that all data will be retrieved in one call, but instead to retrieve all data PIMAP-Store.retrieve() should be continuously called until PIMAP-Store.retrieve() returns an empty list.

PIMAP-Store is not coupled to a particular technology and any implementation of PIMAP-Store must adhere to the simple guidelines as provided. This may make some data-store technologies more or less difficult to implement.

4.2.3. PIMAP-Analyze. PIMAP-Analyze creates PIMAP-metrics from PIMAP-samples. Analysis is often specifically associated with a specific type of PIMAP-sample. This is unavoidable as there are very few types of analysis that can be performed on all types of data that is also useful.

TABLE 2. PIMAP PROFILE OF THREE COMPUTERS

	iMac 2010	System 76 Oryx Pro	Raspberry Pi 2
PIMAP-Sense-UDP.sense()	3,555	49,000	1,217
PIMAP-Store-Kafka.store()	38,780	100,200	3,527
PIMAP-Store-Kafka.retrieve()	102,700	90,840	4,275
PIMAP-Analyze-Objective-Mobility.analyze() In	3,588	10,000	15.20
PIMAP-Analyze-Objective-Mobility.analyze() Out	7,192	20,010	35.00
PIMAP-Visualize-Plt-Graph.visualize()	6,653	1,329	338.0
	PIMAP-samples/metrics a second		

For example Objective Mobility analysis is performed on Pressure Bandage PIMAP-samples. It is nonsensical to perform Objective Mobility Analysis on other types of PIMAP-samples.

From experience with using PIMAP-Analyze there are several different common ways to convert PIMAP-samples to PIMAP-metrics. The most straightforward is a one to one conversion, meaning one PIMAP-sample can be analyzed and used to create a PIMAP-metric. We say this is the most straightforward because in this scenario PIMAP-Analyze has no memory, if you feed in ten PIMAP-samples you will get ten PIMAP-metrics.

Another type of analysis that is also somewhat common is a many to one ratio of PIMAP-samples to PIMAP-metrics, for example if one is analyzing the amount of movements per minute of many PIMAP-samples. In this type of analysis a history or state is kept and then using a time window PIMAP-metrics can be generated using multiple PIMAP-samples.

A less common scenario, but one that we employed in our analysis, is a many PIMAP-sample to many PIMAP-metric ratio with a time delay. We employed this strategy when analyzing the gradient. You need multiple PIMAP-samples to calculate the gradient, but when calculated each PIMAP-sample used in calculation has a gradient value. In this analysis a time window is used to gather PIMAP-samples and then PIMAP-metrics are generated when this threshold is reached.

The varieties of analysis that can be supported is rich and the only guideline we enforce for the PIMAP system is that PIMAP-metrics must be generated, but this can be a one-to-one relationship, many-to-one, or many-to-many with a time delay. And there may be other types of analysis that we have not yet discovered that are also supported.

The primary interaction with PIMAP-Analyze is PIMAP-Analyze.analyze(list of PIMAP-samples), which returns a list of analyzed PIMAP-metrics.

4.2.4. PIMAP-Visualize. PIMAP-Visualize is the component of PIMAP that gives feedback to the clinician/researcher/developer. PIMAP-Visualize takes in either PIMAP-samples or PIMAP-metrics and based on instantiation displays the data. A common type of visualization is displaying the time series data, where the x-axis is the $timestamp$ and the y-axis is the data being displayed. But, other types of visualization are also supported, such as heat maps.

From experience it tends to be more useful to display PIMAP-metrics than PIMAP-samples as the reason why PIMAP-metrics are generated is because the PIMAP-samples are often hard to interpret, but an example of where this may not be the case is data such as room temperature.

Visualizing time series data is common to most types of data, but there may be cases where specialized visualization is necessary and this is supported with PIMAP-Visualize. For example to visualize a graph one could use PIMAP-Visualize-Graph and to visualize a heat map one could use PIMAP-Visualize-Heat-Map. The only guideline is that a PIMAP-Visualize component must take in a list of PIMAP-samples or PIMAP-metrics and visualize these in some way. The main interaction of PIMAP-Visualize is PIMAP-Visualize.visualize(list of PIMAP-samples/metrics).

4.3. PIMAP Workflows

The interaction among PIMAP's components are governed by three main PIMAP workflows:

1) Sense PIMAP-samples and store the PIMAP-samples (sense and store).
2) Retrieve PIMAP-samples, analyze the PIMAP-Samples, and store the respective PIMAP-metrics generated by PIMAP-Analyze (retrieve, analyze, and store).
3) Retrieve PIMAP-metrics and visualize the PIMAP-metrics (retrieve and visualize).

The three workflows are separate, but interlinked, and can be run in parallel and in multiple distributed configurations. For example we present two scenarios, which we entitle Clinic Sense and Cloud Sense to demonstrate two of many scenarios in which PIMAP can be configured.

The *Clinic-Sense* scenario, illustrated in Figure 2c, has one location, the clinic, which could be a hospital room. In this scenario one or more sensor devices send sensor data to a PIMAP-Sense component located in the clinic. The sense and store workflow, the retrieve, analyze, and store workflow, as well as the retrieve and visualize workflow are all located in the clinic. PIMAP is run entirely in the clinic, but makes no assumptions about the underlying technologies at work, for example although PIMAP-Store is running in the clinic, the underlying data-store technology, say Kafka, could be running remotely.

The *Cloud-Sense* scenario, illustrated in Figure **??**, has two general locations, the clinic and the Cloud and are

accessible to each other via a network. In this scenario one or more sensor devices are located in the clinic and send data to a PIMAP-Sense component. The sense and store workflow, the retrieve, analyze, and store workflow are located in the Cloud, and the retrieve and visualize workflow is located in the clinic. This scenario allows for more resources and power to be applied to sensing, analyzing, and storing/retrieving data by utilizing the Cloud. The visualize component would be in the clinic so that clinicians can see the patient's data. To address data security and privacy issues in this data flow scenario, PIMAP-Sense would incorporate data security and integrity mechanisms such as encryption

4.4. Privacy and Security

Privacy and security are obviously very important issues to consider when designing any software system and especially when it comes to handling healthcare information, even if sensor information by itself is a is not self-identifiable to a patient. The current implementation of PIMAP as of this writing has not handled any patient identifiable information, but we expect that in the future we may change and will require private information to be stored and transferred securely. As such, PIMAP will be able to integrate standard security and privacy mechanisms such as authentication and end-to-end encryption.

5. PIMAP Implementation

PIMAP was designed to be easily extensible to accommodate different sensing devices, analytics, and visualization methods. As proof-of-concept, we implemented the following PIMAP component instances in Python: PIMAP-Sense-UDP, PIMAP-Store-Kafka, PIMAP-Analyze-Objective-Mobility, and PIMAP-Visualize-Plt-Graph, each of which is described in detail below. PIMAP components can adapt to application requirements, e.g. to accommodate both low and high throughput scenarios as well as server and network load.

In Section 6 we will evaluate how PIMAP performs under the different conditions and workloads.

To enable PIMAP to dynamically adapt to different application requirements, system conditions, and evaluate these limitations we developed a profiling methodology. We profile PIMAP by analyzing the throughput each component can handle in isolation. The profile is not an absolute limit, but instead an estimate. In the discussion of the implementation decisions made for each component we will also discuss our methods on profiling each component. All throughputs are reported in PIMAP-samples/metrics per second.

In addition we realize there is a benefit for the PIMAP system to monitor itself so that it can adapt to different situations. For this reason each component has a parameter, system-samples, that can be set to generate PIMAP-samples that report on information relevant to each component.

5.1. PIMAP-Sense-UDP

User Datagram Protocol (UDP) is part of the Internet protocol standard and therefore we leverage existing tools to make the PIMAP-Sense-UDP component. UDP is a non-reliable communication protocol and therefore is not appropriate for scenarios where every piece of data is critically important. We recognize this and in future development plan to support reliable communication protocols such as TCP. The PIMAP-Sense-UDP component is a multi-process server that listens on a given host and port. The amount of processes is user configurable, but for evaluation purposes we use three server processes as we did not find a benefit to increasing this number on the systems we profiled. We demonstrate this configuration's adaptivity in Section 6.1.

To profile PIMAP-Sense-UDP we run one process that sends PIMAP-samples as quickly as possible. In a separate process we initialize PIMAP-Sense-UDP to output system-samples that report the throughput and call PIMAP-Sense-UDP.sense() as quickly as possible.

We monitor the generated system-samples and average the reported throughput over the length of profiling. The results of profiling can be seen in Table 2.

5.2. PIMAP-Store-Kafka

We leverage Kafka [10] in our initial PIMAP-Store implementation. Kafka is a publish-subscribe data model, where a producer publishes data to a topic and a consumer can subscribe to a topic to eventually receive every message that was published to the given topic. The PIMAP-Store component is divided into two interfaces, PIMAP-Store.store() and PIMAP-Store.retrieve(), which fits naturally into Kafka as PIMAP-Store-Kafka.store() corresponds to a producer and PIMAP-Store-Kafka.retrieve(topic) corresponds to a consumer. Kafka has the added benefit that data can be distributed across multiple sites (multiple brokers). As PIMAP is written in Python we leverage the confluent-kafka API [27]. A Kafka broker must be setup independently of PIMAP, but this is relatively easy for a developer to setup or a paid cloud-based service can be used instead.

PIMAP-Store-Kafka uses the $sample\_type$ or $metric\_type$ as the Kafka topic and the PIMAP-sample/metric as the Kafka value. We create a consumer per topic requested. Kafka consumers have two parameters that greatly affect the throughput, the number of messages and timeout. When a consumer requests a topic from Kafka it will return after the given number of messages is reached or a given timeout is reached. To make PIMAP-Store-Kafka.retrieve() adaptive we fix the timeout to $100ms$ and decrease the number of messages parameter if a timeout occurs, otherwise we increase the number of messages parameter. We demonstrate how this configuration can dynamically adapt as system conditions change in Section 6.1.

To profile PIMAP-Store-Kafka.store() we initialize PIMAP-Store-Kafka to output system-samples and store as many PIMAP-samples as possible in a single process.

We monitor the generated system-samples and average the reported throughput over the length of profiling.

To profile the PIMAP-Store.retrieve() interface we retrieve the samples that were sent previously in the profile of PIMAP-Store.store(), monitor the system-samples generated and average the throughput reported over the length of profiling. The results of profiling can be seen in Table 2.

5.3. PIMAP-Analyze-Objective-Mobility

Objective Mobility [26] has been proposed as a metric to quantify patient mobility based on pressure readings collected by a custom wearable pressure bandage. Using the four by four grid of pressure sensors embedded into the bandage Objective Mobility reports an approximate angle of the patient on a bed and based on the angle variations in time calculates the amount of movements a patient makes.

PIMAP-Analyze-Objective-Mobility is written in Python and takes advantage of numpy's [28] libraries as well as Python's map utility to perform parallel calculations as often as possible. To enable PIMAP-Analyze-Objective-Mobility to dynamically adaptive to current workload and network conditions we added an aggregation buffer that adjusts based on a timeout. If a timeout occurs we increase the aggregation buffer until we reach a maximum buffer length based on a maximum processing delay tolerance. If aggregation increases past the maximum processing delay tolerance we cut the aggregation buffer in half. Section 6.1 reports on PIMAP-Analyze's ability to dynamically adapt to the underlying system dynamics.

To profile the analyze component we send as many pressure bandage PIMAP-samples as possible in a single process. We monitor the system samples generated and average the throughput over the length of profiling The results of profiling can be seen in Table 2.

5.4. PIMAP-Visualize-Plt-Graph

For our initial visualize component we leverage the matplotlib [29] Python library, a common library used to display data. We focus on graphing data over time. Often when we visualize data in this way we can observe phenomena that would otherwise be unnoticed when looking at a singular value. An example of this is the difference between a time-lapsed photograph and video of the night sky. In the time-lapsed photograph the stars appear as streaks in the sky, whereas a video depicts the stars moving across the sky (of course the reality is the stars are actually not moving from the vantage point of the Earth, but that is besides the point).

To make PIMAP-Visualize-Plt-Graph adaptive we use an aggregation buffer similar to PIMAP-Analyze-Objective-Mobility, a limit on the amount of data that can be displayed, and an update period, which determines how often data is displayed. If the time to process data is above a threshold we decrease the size of the aggregation buffer otherwise we increase the aggregation buffer. If the time it takes to visualize data is greater than the update period we downsample the data to be displayed. This will decrease the resolution, but the only alternative is to increase the update period. We demonstrate this configuration is adaptive in Section 6.1.

To profile the visualize component we visualize as many PIMAP-samples as possible in a single process. We monitor the system-samples generated and average the throughput over the length of profiling. The results of profiling can be seen in Table 2.

TABLE 3. SETUP FOR THROUGHPUT EXPERIMENTS

Application	Location run
Low throughput pressure bandage data, 1 sample/s via UDP	iMac 2010
Medium throughput pressure bandage data, 100 sample/s via UDP	iMac 2010
High throughput pressure bandage data, 2,000 sample/s via UDP	iMac 2010
PIMAP-Sense-UDP	iMac 2010
PIMAP-Store-Kafka	iMac 2010
Kafka	iMac 2010
PIMAP-Analyze-Objective-Mobility	iMac 2010
PIMAP-Visualize-Plt-Graph, update period 1s	iMac 2010
Network	Local network

6. PIMAP Evaluation

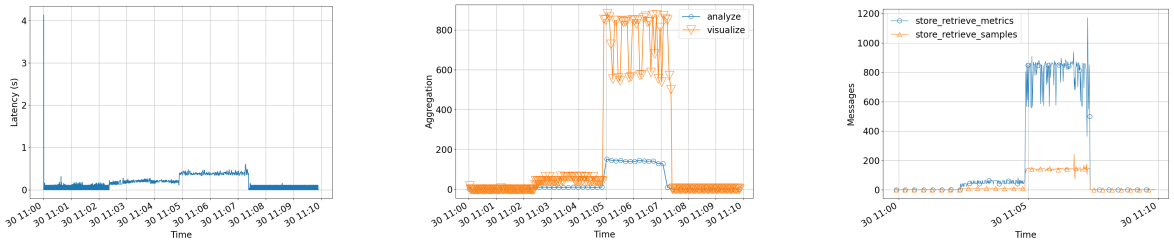
To evaluate PIMAP's features and performance, we ran a variety of experiments including: (1) we evaluate PIMAP's performance in low-, medium-, and high throughput scenarios to demonstrate that PIMAP has low end-to-end latency from the time data is sampled to the time data is visualized regardless of throughput; (2) we demonstrate PIMAP's ability to integrate new sensors, in particular a custom skin health sensor; (3) we connect PIMAP to a sensor network simulation platform (COOJA [30]); and (4) demonstrate how PIMAP can be used to analyze and visualize data in real-time by playing back data obtained from a wearable pressure bandage.

6.1. Low, Medium, And High Throughput Scenario With Low End To End Latency

To demonstrate PIMAP's ability to adapt to different throughput scenarios, we evaluate PIMAP's end-to-end latency, i.e., the time between when the data is sampled to the time it is visualized. We also monitor how PIMAP adapts to low, medium, and high throughput scenarios.

TABLE 4. OBJECTIVE MOBILITY EXPERIMENTAL SETUP

Application	Location run
Patient data (x5) sent at 1 sample/s via UDP	iMac 2010
PIMAP-Sense-UDP	iMac 2010
PIMAP-Store-Kafka	iMac 2010
Kafka	Remote Server
PIMAP-Analyze-Objective-Mobility	iMac 2010
PIMAP-Visualize-Plt-Graph, update period 1s	iMac 2010
Network	Internet



(a) End to End Latency (b) Aggregation (c) Retrieved Messages

Figure 3. Evaluating PIMAP's adaptiveness

We deploy all components of PIMAP locally on one computer (see Table 3 for the experimental configuration used). Based on the profiling information listed in Table 2 the limiting throughput is PIMAP-Sense, which is a little over 3,500 PIMAP-samples/s. Based on this limit we experiment with a low throughput scenario of 1 sample/s, a medium throughput scenario of 100 PIMAP samples/s, and a high throughput scenario of 1,000 PIMAP samples/s. We ran each experiment for ten minutes and calculated the average end-to-end latency for the low throughput scenario to be $20.0ms$, for the medium throughput scenario to be $203ms$, and for the high throughput scenario to be $411ms$. From these results, we observe that PIMAP's latency does not scale linearly with sample rate and is closer to logarithmic scaling, and even at a high sample rate of 1,000 samples/s, PIMAP stays below $500ms$.

To showcase how PIMAP can adapt to the dynamics of the application and underlying network, we ran an additional experiment where we dynamically adjust the sample rate. We use the same setup as in the previous experiment and for two and a half minutes we send at a low throughput of 1 sample/s, the next two and a half minutes we send at a medium throughput of 100 samples/s, the next two and a half minutes we send at a high throughput of 1,000 samples/s, and for the final two and a half minutes we send at a low throughput of 1 samples/s. The results can be seen in Figure 3 and it we can observe how the system is able to adapt as we increase sample rates.

6.2. Support For Different Sensor Types

An important consideration when designing PIMAP was to allow for various types of sensors. If we only support one type of sensor the system does not provide value to the majority of the community and as discussed in Section 2 the literature tends to provide such one off systems, which makes it difficult to reuse existing implementations. To support a large variety of types of sensors we provide very loose guidelines on what type of data can be sent in a PIMAP-sample. So far in our development we have not encountered data that we cannot encode in a PIMAP-sample or PIMAP-metric.

To reiterate the sensor information of a PIMAP-sample is stored in the *sample* field. In our implementation this field is a string representation of a dictionary. For example pressure bandage data with no pressure applied, used in Objective Mobility analysis [26], can be converted to a string and will look as follows:
"sample: {'pressure_bandage':'[[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0]]', 'pressure_bandage_units':'mmHg'}"

To unpack the sample one can use the built-in `ast` (Abstract Syntax Trees) Python library that can convert a syntactical grammar into its corresponding type. Another way data can be passed and converted is using the builtin `pickle` Python library, by pickling data when inserting as a *sample* and unpickling when reading the PIMAP-sample and analyzing.

To demonstrate PIMAP's ability to incorporate new sensors we setup an environment in which we gathered, analyzed, and visualized the impedance spectroscopy data from a single Sentinel bandage [31], which we obtained through our collaboration with UCSF. All tests were run on a single Linux laptop and on a local network. The Sentinel bandage generating impedance spectroscopy data was plugged into the laptop via USB and a custom PIMAP-Sense component was used to read the serial data and convert it to a PIMAP-sample. The PIMAP-samples were analyzed using a custom PIMAP-Analyze component that converted the data into a PIMAP-metric that can be displayed using a heat map. Finally we created a custom PIMAP-Visualize component that can display the heat map data.

7. Future Work And Conclusion

In this work we presented PIMAP, an IoT system for continuous, autonomous, real-time patient monitoring that can be deployed in a clinical and long-term care facilities as well as at home. As a holistic patient monitoring system, it integrates the basic patient monitoring workflow, i.e., sensed data collection, storage, analytics and visualization. We demonstrated how PIMAP can be use in real-world settings, in particular acquiring and processing pressure data from a custom pressure sensor, and based on the assessed mobility levels of the patients, determining their risks of developing pressure ulcers, and presenting that information to clinicians in real time. We also evaluated PIMAP's performance and showed that it yields adequate end-to-end latency in both low throughput and high throughput scenarios. We also show that PIMAP is able to integrate new sensors. PIMAP is planned to be released under an open source license.

References

[1] D. R. Berlowitz and D. M. Brienza, "Are all pressure ulcers the result of deep tissue injury? A review of the literature." *Ostomy/Wound Management*, vol. 53, no. 10, pp. 34–38, 2007. [Online]. Available: http://www.o-wm.com/content/are-all-pressure-ulcers-result-deep-tissue-injury-a-review-literature

[2] K. Agrawal and N. Chauhan, "Pressure ulcers: Back to the basics," *Indian Journal of Plastic Surgery: Official Publication of the Association of Plastic Surgeons of India*, vol. 45, no. 2, p. 244, 2012. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3495374/

[3] D. L. Bader and P. R. Worsley, "Technologies to monitor the health of loaded skin tissues," *BioMedical Engineering OnLine*, vol. 17, p. 40, Apr. 2018. [Online]. Available: https://doi.org/10.1186/s12938-018-0470-z

[4] C. A. Russo, C. Steiner, and W. Spector, "Hospitalizations related to pressure ulcers among adults 18 years and older, 2006: statistical brief# 64," 2006.

[5] N. Q. F. (NQF), "Serious Reportable Events in Healthcare—2011 Update: A Consensus Report," NQF Washington, DC, Tech. Rep., 2011.

[6] N. P. U. A. P. (US) and E. Haesler, *Prevention and treatment of pressure ulcers: quick reference guide*. Cambridge Media, 2014.

[7] S. Mansfield, K. Obraczka, and S. Roy, "Pressure Injury Prevention: A Survey," *IEEE Reviews In Biomedical Engineering*, pp. 1–3, 2019.

[8] F. Touati, A. B. Mnaouer, O. Erdene-Ochir, W. Mehmood, A. Hassan, and B. Gaabab, "Feasibility and performance evaluation of a 6LoWPAN-enabled platform for ubiquitous healthcare monitoring," *Wireless Communications and Mobile Computing*, vol. 16, no. 10, pp. 1271–1281, 2016. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/wcm.2601

[9] V. Baljak, A. Ljubovic, J. Michel, M. Montgomery, and R. Salaway, "A scalable realtime analytics pipeline and storage architecture for physiological monitoring big data," *Smart Health*, vol. 9-10, pp. 275–286, Dec. 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2352648318300485

[10] "Kafka." [Online]. Available: kafka.apache.org

[11] A. Curtis, A. Pai, J. Cao, N. Moukaddam, and A. Sabharwal, "HealthSense: Software-defined Mobile-based Clinical Trials," in *The 25th Annual International Conference on Mobile Computing and Networking - MobiCom '19*. Los Cabos, Mexico: ACM Press, 2019, pp. 1–15. [Online]. Available: http://dl.acm.org/citation.cfm?doid=3300061.3345433

[12] C. Stach, F. Steimle, and A. C. F. da Silva, "TIROL: the extensible interconnectivity layer for mHealth applications," in *International Conference on Information and Software Technologies*. Springer, 2017, pp. 190–202.

[13] S. Neubert, A. Geißler, T. Roddelkopf, R. Stoll, K.-H. Sandmann, J. Neumann, and K. Thurow, "Multi-Sensor-Fusion Approach for a Data-Science-Oriented Preventive Health Management System: Concept and Development of a Decentralized Data Collection Approach for Heterogeneous Data Sources," *International Journal of Telemedicine and Applications*, vol. 2019, pp. 1–18, Oct. 2019. [Online]. Available: https://www.hindawi.com/journals/ijta/2019/9864246/

[14] L. Esch, L. Sun, V. Klüber, S. Lew, D. Baumgarten, P. E. Grant, Y. Okada, J. Haueisen, M. S. Hämäläinen, and C. Dinh, "MNE Scan: Software for real-time processing of electrophysiological data," *Journal of Neuroscience Methods*, vol. 303, pp. 55–67, Jun. 2018.

[15] A. Rodriguez, P. Smielewski, E. Rosenthal, and D. Moberg, "Medical Device Connectivity Challenges Outline the Technical Requirements and Standards For Promoting Big Data Research and Personalized Medicine in Neurocritical Care," *Military Medicine*, vol. 183, no. suppl_1, pp. 99–104, Mar. 2018. [Online]. Available: https://academic.oup.com/milmed/article/183/suppl_1/99/4960031

[16] G. B. Rehm, B. T. Kuhn, J.-P. Delplanque, E. C. Guo, M. K. Lieng, J. Nguyen, N. R. Anderson, and J. Y. Adams, "Development of a research-oriented system for collecting mechanical ventilator waveform data," *Journal of the American Medical Informatics Association*, vol. 25, no. 3, pp. 295–299, Mar. 2018. [Online]. Available: https://academic.oup.com/jamia/article/25/3/295/4571786

[17] L. Gunningberg and C. Carli, "Reduced pressure for fewer pressure ulcers: can real-time feedback of interface pressure optimise repositioning in bed?" *International Wound Journal*, vol. 13, no. 5, pp. 774–779, 2016.

[18] L. Gunningberg, I.-M. Sedin, S. Andersson, and R. Pingel, "Pressure mapping to prevent pressure ulcers in a hospital setting: A pragmatic randomised controlled trial," *International Journal of Nursing Studies*, vol. 72, pp. 53–59, 2017.

[19] D. Pickham, N. Berte, M. Pihulic, A. Valdez, B. Mayer, and M. Desai, "Effect of a wearable patient sensor on care delivery for preventing pressure injuries in acutely ill adults: A pragmatic randomized clinical trial (LS-HAPI study)," *International Journal of Nursing Studies*, vol. 80, pp. 12–19, 2018.

[20] "National Pressure Ulcer Advisory Panel (NPUAP) announces a change in terminology from pressure ulcer to pressure injury and updates the stages of pressure injury | The National Pressure Ulcer Advisory Panel - NPUAP." [Online]. Available: http://www.npuap.org/national-pressure-ulcer-advisory-panel-npuap-announces-a-change-in-terminology-from-pressure-ulcer-to-pressure-injury-and-updates-the-stages-of-pressure-injury/

[21] Hopkins Alison, Dealey Carol, Bale Sue, Defloor Tom, and Worboys Fran, "Patient stories of living with a pressure ulcer," *Journal of Advanced Nursing*, vol. 56, no. 4, pp. 345–353, Sep. 2006. [Online]. Available: https://onlinelibrary.wiley.com/doi/full/10.1111/j.1365-2648.2006.04007.x

[22] Gorecki Claudia, Brown Julia M., Nelson E. Andrea, Briggs Michelle, Schoonhoven Lisette, Dealey Carol, Defloor Tom, and Nixon Jane, "Impact of Pressure Ulcers on Quality of Life in Older Patients: A Systematic Review," *Journal of the American Geriatrics Society*, vol. 57, no. 7, pp. 1175–1183, Jun. 2009. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1532-5415.2009.02307.x

[23] B. Braden and N. Bergstrom, "A Conceptual Schema for the Study of the Etiology of Pressure Sores," *Rehabilitation Nursing*, vol. 12, no. 1, pp. 8–16, 1987.

[24] J. P. Tran, J. M. McLaughlin, R. T. Li, and L. G. Phillips, "Prevention of Pressure Ulcers in the Acute Care Setting: New Innovations and Technologies," *Plastic and Reconstructive Surgery*, vol. 138, pp. 232S–240S, Sep. 2016. [Online]. Available: http://Insights.ovid.com/crossref?an=00006534-201609001-00030

[25] C. VanGilder, C. Lachenbruch, C. Algrim-Boyle, and S. Meyer, "The International Pressure Ulcer Prevalence™ Survey: 2006-2015," *Journal of Wound, Ostomy and Continence Nursing*, vol. 44, no. 1, pp. 20–28, 2017. [Online]. Available: http://www.ingentaconnect.com/content/wk/won/2017/00000044/00000001/art00005

[26] S. Mansfield, S. Rangarajan, K. Obraczka, H. Lee, D. Young, and S. Roy, "Objective Pressure Injury Risk Assessment Using A Wearable Pressure Sensor," in *2019 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*. San Diego, CA, USA: IEEE, Nov. 2019, pp. 1561–1568. [Online]. Available: https://ieeexplore.ieee.org/document/8982939/

[27] "Python Client for Apache Kafka." [Online]. Available: https://github.com/confluentinc/confluent-kafka-python

[28] "NumPy." [Online]. Available: numpy.org

[29] "Matplotlib." [Online]. Available: matplotlib.org

[30] F. Osterlind, A. Dunkels, J. Eriksson, N. Finne, and T. Voigt, "Cross-level sensor network simulation with cooja," in *Local Computer Networks, Proceedings 2006 31st IEEE Conference on*. IEEE, 2006, pp. 641–648. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4116633

[31] S. L. Swisher, M. C. Lin, A. Liao, E. J. Leeflang, Y. Khan, F. J. Pavinatto, K. Mann, A. Naujokas, D. Young, S. Roy, and others, "Impedance sensing device enables early detection of pressure ulcers in vivo," *Nature Communications*, vol. 6, p. ncomms7575, 2015.