St. Paul University Surigao
St. Paul University System
Surigao City, Philippines

SAFER: AN IOT-BASED REAL-TIME LIGHT AND HEAVY EQUIPMENT MONITORING SYSTEM FOR OPEN-PIT MINING OPERATIONS

A Draft Thesis Manuscript Presented to

The Faculty of the College of Engineering

St. Paul University Surigao

Surigao City

In Partial Fulfillment

of the Requirements for the Degree

BACHELOR OF SCIENCE IN MINING ENGINEERING

By:

Basil Eric C. Rabi

James M. Paje

John Kenneth C. Velonta

Richard Banog

April 2024

TABLE OF CONTENTS

CHAPTER 1

THE PROBLEM AND ITS BACKGROUND

INTRODUCTION

The Role of Data in the Mining Industry

Data is the life-blood of any company, regardless of which industry this may come from, as it provides vital insights that can help in making informed decisions. Due to the sheer volume of industrial data that could be extracted at any given time, it is often a big undertaking to even attempt to make a data pipeline that is scalable to the highly defining characteristics of industrial data; high dimensionality and process dynamics, and large yet redundant [1].

The mining industry is no exception when it comes to large amount of industrial data prime for extraction. For example, mining companies, whether surface operations or otherwise, utilize large fleets of equipment for various purposes. This alone can generate thousands, if not millions of rows of real-time data per day depending on the parameters that the company decides to obtain from these equipment. The advent of using information technologies in big industries have led authors to coining different terminologies such as *Smart Mining*, *Industry 4.0*, and *Digital Revolution* among others [2].

One emerging technology that has the potential to revolutionize the mining industry is the (Internet of Things (IoT)). IoT is a concept that enables seamless communication between electronic devices and is already widely used in various fields such as environmental monitoring, healthcare, and transportation [3]. By leveraging IoT, real-time information can be gathered and companies can improve the reliability of their data collection by automating tasks and minimizing human errors. Access to real-time information also allows

making decisions and taking actions in a more prompt manner.

While there is a growing sentiment that data is a very important aspect of an organization's operation, it is not a surprise that the Philippines, a developing country, is still trying to catch-up when it comes to applying data engineering, analytics, and science practices.

In Philippine surface mines, it is still common to see equipment usage data collected using manual or paper-based forms. In this type of data collection, the equipment operators are tasked to log the time and type of utilization via pen-and-paper. Employing this kind of data collection is both very tedious to the operators and highly prone to errors. And being in the day and age where quick access to real-time data is being recognized as a need than a want, it is impossible to be able to deliver the data related needs of the company when utilizing this data collection method.

This method of data gathering causes inadequate operational control, as any inefficiency can only be detected and quantified after a week or so of data collection, or depending how long it takes from data collection to data encoding. This delay makes it difficult to detect occurrences of idling equipment and sub-optimal loader-to-hauler ratios in equipment deployment, which increases resource consumption. Inadequacy of operational control can cause inactivity of equipment that affects the end cost per tonnage of ore hauled.

Present Challenges in Surface Mines Caused by the Lack of Real-Time Monitoring System

A company that lacks visibility and access of its data not only suffers financially, but in the other aspects of the overall operation as well.

Weak Enforcement of Speed Limit

Speed limits are imposed in all surface mines to lessen risks of accidents and to control dust generation. However, enforcement of the speed limit is only done thru verbal instructions and violations can only be detected through random checks by safety inspectors. This results in a low detection rate of over-speeding vehicles.

Low Data Integrity

The current paper-based data collection method is prone to human error, which affects data integrity. The forms are initially filled-out by operators, manually checked by supervisors, and signed-off by foremen after an 8 to 12-hour shift. The filled-out forms are then encoded in a spreadsheet the next day while data consolidation and review are done on a monthly basis. Human errors can be introduced twice in this process: during filling-out and during encoding. Erroneous data are not completely detected and corrected since spreadsheet-encoded data can only be checked manually. Furthermore, the mine operations team and the equipment maintenance team monitor equipment usage separately, often resulting in contradicting information.

Untimely Equipment Repair and Maintenance

When an equipment breakdown occurs, repairing the down unit can be delayed from 1 shift up to several days if the breakdown is not promptly relayed to the maintenance crew, or if the maintenance crew is unable to pinpoint the exact location of the down unit during their deployment. Moreover, equipment servicing is done at fixed intervals of usage (*e.g.*500 hours) as part of the equipment preventive maintenance program. However, actual usage of the equipment prior to servicing often exceeds this duration since equipment usage

data is finalized monthly. This poor implementation of the maintenance program leads to early downtime of equipment [4].

To alleviate the challenges discussed, the researchers of the study and aim to develop an IoT-based equipment that serves to monitor the utilization of mining equipment in real-time and will completely replace the pen-and-paper method of collection by developing a software that will integrate multiple commercially available and affordable hardware.

CONCEPTUAL FRAMEWORK OF THE STUDY

This study aims develop an IoT-based equipment monitoring system called Service Activity and Fleet Efficiency Recorder, or SAFER for short. A similar IoT system in a research study of a home security system which used a GSM module and a Raspberry Pi [5] will be applied in this study. There will be four major milestones in the development cycle of the study as illustrated in Figure 1.1.

Each part of the development cycle shall be discussed in the succeeding chapter in full capacity. The basic premise of the framework is first to identify the data and feature requirements to be included in the study. Next is building a system design based upon the identified requirements, then the evaluation of the built system and a process loop based on the feedback of the end-users. And then lastly, evaluation of the impact of the final system produced by the researchers.

STATEMENT OF THE PROBLEM

The output of this study, which is a real-time monitoring and control system for equipment usage leveraging IoT technology, can lead to improved operational efficiency, stronger enforcement of speed limits, higher data integrity, and decreased equipment downtime in Philippine surface mines. Throughout the duration of this study, the following

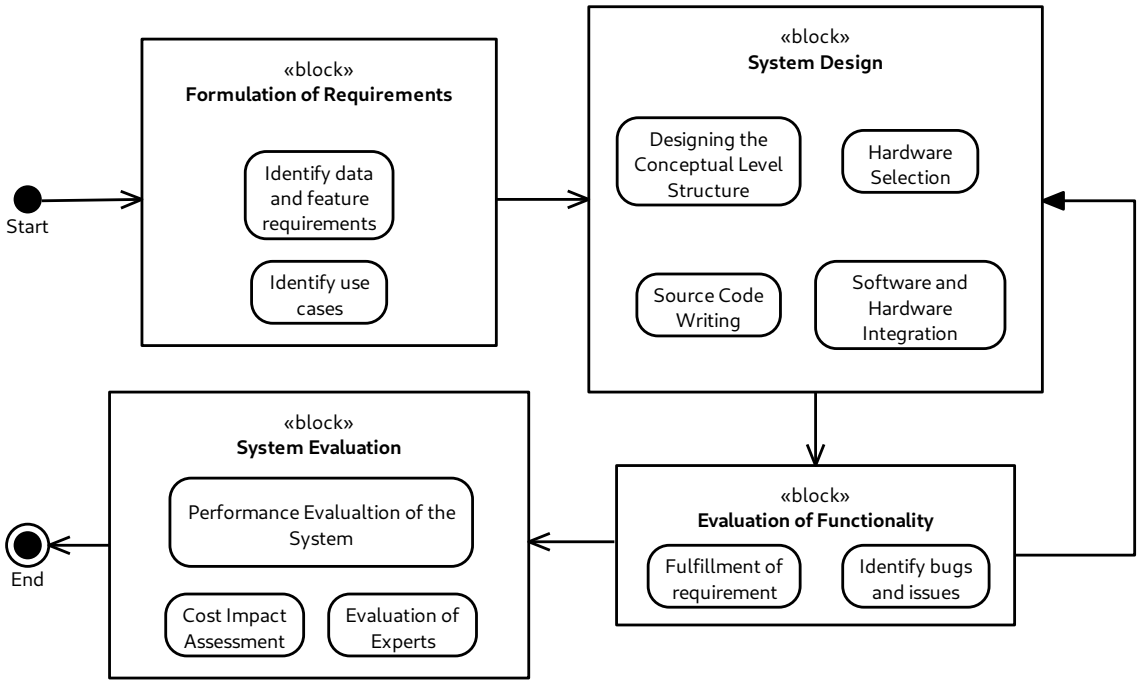

Figure 1.1: Development cycle of SAFER, an IoT-based real-time equipment activity monitoring system.

questions will be addressed:

1. What is the status of the current system of the host company in terms of
   (a) Gap between the SMR during actual PMS servicing activity and the pre-defined SMR requiremment for PMS (*e.g.*500 hours for Heavy equipment and 500 km for light equipment)
   (b) Equipment over-speeding detection
   (c) Idling time registry accuracy and detection
   (d) End-user's requirement and feedback
2. What monitoring system can we develop in terms of
   (a) Hardware stack selection
   (b) Software stack selection
   (c) Development process
3. What is the evaluation of the system by the experts and users in terms of
   (a) Efficiency
   (b) Stress tolerance
   (c) User's experience

SIGNIFICANCE OF THE STUDY

The objective of the study is to produce a working prototype of a real-time monitoring and control system for equipment usage. This study will leverage IoT technology through the use of commercially available GSM and GNSS modules removing the need for paper-based forms and manual data encoding, and removing the need for mobile internet data reception to transmit live data to the server. This study also makes it possible to create a real-time dashboard and location mapping of equipment which will improve

the response time and decision-making flexibility to sudden or unexpected changes and sub-optimal conditions in the operation.

This study is in-line with the goal of environmental sustainability of the mining industry through the use of modern technology. The use of the system being proposed in this study will help mining companies control their fleet more efficiently, reducing the fuel consumption caused by idling of equipment due to over or under assignment of trucks within an area. The proposed system also hosts a feature that alerts over-speeding of trucks, which could reduce the dust generation during the hauling activity and solid waste generation from tire wear. The host company could also benefit differently from this feature, as it also alerts the supervisors, safety officers, and managers when over-speeding has occurred, reducing the risk of accidents.

The proposed features of the end-product are; a) real-time equipment location monitoring, b) paperless equipment activity monitoring, c) touch-based equipment activity input, d) over-speeding detection and notification system, e) SMS-based data send-out, f) web-based front-end for real time data monitoring, g) paperless service meter run data collection, h) automated preventive maintenance schedule notification system, i) and assisted correction of data through automation of anomalous data detection, all of which are subject to change depending on the feedback of the end-users.

The proposed system has the potential to minimize environmental impacts and reduce the risk of vehicular accidents. With this system in place, mining companies can improve their operations, enhance safety, and promote sustainability.

SCOPE AND LIMITATION OF THE STUDY

The whole system and product development, testing, data collection, and end-user engagement will be done solely at TMC for the purpose of convenience. This means that the product's feature design, user interface and user experience choices, and the study's conceptual layout will be based on the operational framework and end-user feedback from TMC alone. Should the output of this study be applied elsewhere, minor changes will have to be made in order to accommodate the different operational framework of another mining company.

A limiting factor that may affect the degree of real time data collection process is the SMS network reception of a given area. The researchers will be addressing this issue and provide a solution so that the data collected by the device could still be sent at a later time when network reception is available. The available solutions however, could still reduce the degree of real time data collection.

DEFINITION OF TERMS

AT commands. Attention Commands. These are commands used for controlling GSM modems and are based on the Hayes command set [6]

blender. A free, open-source, professional-grade 3D computer graphics software that can be used in multiple 3D based graphics tasks such as in animation, engineering and manufacturing modelling. The software has been supporting 3D printing modelling since 2002, making it one of the most widely used software by 3D-printing enthusiasts and professionals. [7]

C. A high-level general-purpose programming language. Programs written in this language needs to be compiled but can be optimized based on targeted hardware

daemon. A long-running program in the background which usually starts right after the computer turns on and terminates when the computer shuts down [8]

gammu. A programming library to easily interface with GSM devices [9]

GNSS. Global Navigation Satellite System. A navigation system which provides autonomous geo-spatial positioning on a global coverage. This uses satellite constellations such as GPS (USA), GLONASS (Russia), Beidou (China), Galileo (EU), INRSS (India) and QZSS (Japan) [10]

GSM. Global System for Mobile Communication. A wireless communication system which enables exchanging of Short Message Service (SMS) text messages between mobile devices [11]

GTK. An open-source library used for creating graphical user interface [12]

heavy equipment. Machineries used in mining operations that are mostly utilized in the loading of ores. Compared to light equipment, heavy equipment are most of the time static in location Examples are track excavators, wheel loaders, bulldozers and more

Internet of Things (IoT). A network of electronic devices with seamless communication with each other [3]

light equipment. Machineries used in mining operations that are wheeled and more mobile in nature than heavy equipment. Examples are dumptrucks, service vehicles, water lorries, and more

microplastics. Plastic particles with sizes of less than 5 millimeters [13]

PMS. Preventive Maintenance Schedule. This is a program designed to prevent equipment breakdowns and extend the lifespan of equipment through regular maintenance activities such as inspections, cleaning, lubrication, and replacement of worn or damaged parts. The schedule is based on the manufacturer's recommendations, operating conditions, and equipment usage

PostGIS. An extension of PostgreSQL which allows storing and processing geo-spatial data such as points, line strings and polygons [14]

PostgreSQL. An SQL-compliant open-source relational database management system [15]

python. A modern interpreted general-purpose programming language. Unlike C, programs written in this language do not require compilation and are easier to read and debug

Raspberry Pi. A low-cost single-board computer with a processor similar to smart phones

redis. An in-memory database which is usually used as message broker for various applications due to its high speed [15]

SMR. Service Meter Run. Number of hours or kilometers that the equipment has operated

TMC. Taganito Mining Corporation. A mining company operating a surface mine situated in Tagantio, Claver, Surigao del Norte, Philippines

CHAPTER 2

REVIEW OF RELATED LITERATURE

Discussed in this chapter are literature that is related to the study of the authors. They may provide insights on how to execute the said study.

ENVIRONMENTAL IMPACTS OF EQUIPMENT IN SURFACE MINES

Aside from the obvious temporary destruction caused by unearthing of ores, one of the ways that a mining company contributes to the destruction of the natural ecosystem is by the use of equipment in their operations. Heavy and light equipment and machineries are essential in large scale surface mine operations in order to have an acceptable amount of throughput.

However, usage of such tools in surface mines have significant impacts to the environment. These impacts include greenhouse gas emissions, consumption of non-renewable resources from fuel and equipment parts, dust generation, and accident risks. Discussed in detail in the next subsections are ways that the use of mining equipment can contribute to the degradation of the natural environment.

Fuel Consumption and Greenhouse Gas Emissions

Heavy equipment and machinery contribute significantly to greenhouse gas emissions due to their high fuel consumption. Based on the available data from TMC (Taganito Mining Corporation), a mining company operating at Claver, Surigao del Norte, equipment units consumes 9-25 liters of diesel fuel per hour depending on the model. The worldwide sales of such equipment amount to one million units annually, making it imperative to address this issue [16]. Furthermore, the inefficient use of these machines only exacerbates

the problem.

Idling equipment units are a significant cause of increased fuel consumption in surface mines. Equipment idling can be caused by many different reasons, but ultimately, it boils down to mismanagement of equipment utilization as the root cause. Many factors are observed in order to prevent mismanagement (and eventually idling) of equipment, one of which is the correct loader-to-hauler ratio. A mismatch in the loader-to-hauler ratio can further reduce fuel efficiency. If the number of haulers is too high, this can lead to queuing of haulers waiting to be loaded by the loaders. This queuing can lead to an increase in fuel consumption per ton moved. Conversely, if there are too few haulers, the loading unit will idle waiting for haulers to arrive, leading to an increase in fuel consumption per ton moved in loaders.

Real-time data can be utilized to effectively reduce fuel consumption by identifying idling units and addressing the loader-to-hauler equipment ratio promptly. By doing so, fuel consumption can be reduced, and greenhouse gas emissions can be lowered.

Dust Generation and Accident Risk

Studies have shown that hauling speed is strongly correlated with both dust generation and the risk of accidents. Higher speeds can lead to increased dust concentrations [17], as well as an increase in both the frequency and severity of accidents [18].

Unsafe driving practices such as overtaking along intersections and curves, and over-speeding violate rule 414 of DENR Administrative Order 2000-98 [19]. It is essential to take proactive measures to detect and control such violations. By utilizing real-time data, haulers violating the speed limit can be identified and corrected promptly.

Implementing measures to control speed violations can have a significant impact on

reducing dust generation and improving safety performance. Compliance with Philippine Mine Safety Regulations can also be ensured through this approach. By taking proactive measures, we can reduce the risk of accidents, promote a safer work environment, and reduce the negative impact of dust generation on health and the environment.

Solid Waste Generation from Tires

Solid waste generation from tire wear and paper usage are two environmental issues that are prevalent in the mining industry. Higher speeds are known to cause increased tire wear [20], which in turn generates more waste from used tires. Furthermore, tire wear contributes to the generation of microplastics, which are a widespread pollutant in marine environments [21]. One way to minimize waste generation, including microplastics, is to control the speed of wheeled equipment.

Paper Usage

In the Philippines, surface mines collect equipment monitoring data using paper-based forms. This practice requires each operator to fill out a two-page form summarizing their equipment usage during their shift. With this system in place, a surface mine with 500 equipment units can consume a ton of paper in less than three months. However, the paper industry is known to cause several environmental issues, such as deforestation, greenhouse gas emissions, and water pollution [22].

One solution to these issues is to implement a paperless and real-time monitoring system for equipment usage. This system would reduce solid waste generation from tires and eliminate the need for paper usage. By using digital forms, surface mines can significantly reduce their environmental impact and promote sustainability.

IOT IN THE MINING INDUSTRY

IoT is now being used in mining companies worldwide to promote sustainability, improve safety, and enhance productivity [23]. In the Philippines, an IoT system called ER MineTracer was recently designed to improve emergency response during an incident in a mine by replacing manual communication of location via handheld radios with more reliable devices such as GPS units and mobile phones [24].

One example from the international landscape is the Koodaideri Iron Ore Mine in Pilbara, Western Australia, a project of the Rio Tinto Group. In 2018, the Rio Tinto Group announced its plans to build an 'intelligent mine' for its Koodaideri project [25]. In 2019, the group declared its partnership with the machineries manufacturer, Caterpillar, and exclusively purchased equipment from the manufacturer with agreement to study the potential of increased automation in Koodaideri project [26]. In 2021, Rio Tinto Group officially launched its Safe Production System (SPS) as a way to improve their operation through automated data collection[27], a culmination of their years of effort to build an 'intelligent mine'.

In this specific instance however, Rio Tinto's Koodaideri Project is exclusively served by Caterpillar and its proprietary systems. While the technology is already available, most commercially present systems are vendor-locked to the equipment manufacturer. In the case of most mining operations in the country, this would be less beneficial and less cost-effective as most of the companies purchase different types of equipment from different manufacturers. Thus the need for a low-cost, manufacturer agnostic system to automatically monitor and collect data is a challenge that if addressed, would be very

beneficial to the host mining company.

Low-cost Internet of Things (IoT) for Monitoring and Optimising Mining Small-scale Trucks and Surface Mining Shovels

The study by Aguirre-Jofre, et al., delves into the integration of low-cost Internet of Things (IoT) technology within medium-scale open pit mining operations. It concentrates on optimizing the performance of small-scale trucks and surface mining shovels, typically managed by commercial Fleet Management Systems (FMSs). However, the study underscores that medium-scale mines often lack the capital necessary for such systems [28].

To address this, the research presents a trial-based IoT-based Fleet Information System (FIS) designed specifically for medium-scale mines. This FIS aims to automate data collection, providing key operational insights like truck cycles per shift, loading times, truck and shovel positioning, and average fleet efficiency. By replicating certain functionalities of commercial FMSs, this low-cost IoT solution facilitates operational optimization at a fraction of the cost.

The case study conducted in a Chilean mine serves as a successful demonstration of the implemented FIS. It led to strategic operational changes that reduced loading times, optimized truck routes, and improved safety without escalating mining costs. This cost-effective solution offers medium-scale mines an opportunity to enhance their performance, potentially extending their life-of-mine, fostering local employment, and positively impacting the community.

The paper emphasizes the essential need for mining companies to mitigate operational costs to maximize profits amidst volatile market prices. It highlights the pivotal role of optimizing material handling, specifically truck-shovel systems, to reduce overall min-

ing expenses. While large-scale mines historically employed FMSs for these efficiencies, their high costs and technical prerequisites often hinder their implementation in medium-scale operations.

This study is closely related to the researchers own study as both groups employ the use of single-board computer that greatly reduces the cost of the project should it be implemented. Another aspect that the study by Aguirre-Jofre is related to the researchers own study is how it uses GNSS to collect geospatial data of the equipment. However, one key points make the two studies unique from each other. The researchers' use of SMS technology is not present in Aguirre-Jofre's study. Aguirre-Jofre et al. transmits their data via Mobile Broadband Modem or commonly known as mobile data. The researchers of this study decided to use SMS technology to hurdle the problem of lack of mobile data reception in many open-pit minesites in the Philippines.

SMS INTEGRATED IOT

It is a common misconception to believe that Internet-of-Things requires internet, but not surprising since the word *internet* is in the word itself. However, what the word IoT actually means is the interconnectedness of different objects through wireless communication with their function in-sync with one another. Through this definition one can argue that SMS can be a core part of wireless communication between objects. Presented below is a study that takes advantage SMS technology in their IoT design.

IoT-based smart protection system to address agro-farm security challenges in Bangladesh

In the realm of agro-farm security, there's a substantial gap in tailored IoT-based systems for Bangladesh. While this region thrives in agriculture, research around real-time, farmer-centric security systems is limited. Existing studies, albeit few, have made notable strides in innovating security measures globally. A Farm Intrusion Detection System using IoT, allowing immediate alerts upon farm trespassing was previously published. Building on this, another study explored advanced security systems employing IoT and image processing, enabling live camera streams for farmers. Different approaches emerged, including eco-friendly virtual fences by Suman et al. and smart security systems by Hossain et al. using Arduino-based operations.

Several studies highlighted IoT's vital role in agricultural advancement, identifying its potential to address manual monitoring and crop damage issues. Notably, a Solar Fencing Unit and Alarm for Animal Entry Prevention, aiming to secure crops from dangerous animals through electric fencing was also developed. The research scope expanded to aquaculture, such a laser-based security system in Bangladesh, ensuring safety in shrimp and prawn farming environments.

The proposed IoT-based agro-farm security system in Bangladesh by Sayem et al. operates through vibration and laser sensors at the entrance gate of the farm shed. [29] Upon detecting intrusions, these sensors trigger alerts to the farmer's mobile phone and house with the use of automated *SMS* sendout. The system's hardware setup involved rigorous testing of electronic components, ensuring their compatibility and functionality. Furthermore, a meticulous construction process was undertaken, employing 3D modeling

and precise construction to create stands equipped with laser sender and receiver sensors.

The system's functionality was tested across different calibration levels of vibration sensors, illustrating how calibration settings influence vibration intensities. The laser sensor demonstrated accurate responsiveness to intrusions. Additionally, the GSM module, serving as a vital communication link, underwent signal strength evaluation and performed well in facilitating real-time alerts and notifications. The system's robustness and effectiveness were assessed, paving the way for a comprehensive understanding of its performance in enhancing agro-farm security.

This is one of the many studies that employs SMS technology within their IoT systems. In this particular study conducted in Bangladesh, the researchers used SMS technology as an automatic communication to the farmers whenever intrusion is detected by the sensors. The researchers of SAFER will also utilize SMS technology in communicating, in this case sending out utilization and geospatial data from the equipment to the central servers, cutting out the need to use mobile data to transmit data.

Monitoring and controlling the desalination plant using IoT

This study by Murugan et al. focuses on leveraging IoT technology to monitor and manage desalination plants, crucial in meeting the increasing demand for clean water amid environmental degradation. [30] IoT enables automating tasks, especially in water treatment management. The proposed model involves sensors to monitor parameters like pH, temperature, and dissolved oxygen, storing data in the cloud. It notifies about malfunctions, regulates valves, and has shown effectiveness compared to other techniques, extending operational time compared to various methods by certain percentages.

The literature survey covers various research endeavors, focusing on IoT's role in monitoring water quality, proposing systems with diverse sensor functionalities, fault detection mechanisms, and cloud-based data handling. These studies address issues like reliability, cost, and efficiency in different ways, utilizing IoT devices, cloud integration, and diverse sensor technologies.

Murugan et al. delineates the proposed setup's components, including NodeMCU ESP8266, GSM modules, temperature, pH, turbidity sensors, water level sensors, and electrical fault detectors. The hardware setup is linked to a cloud system, facilitating remote monitoring and control.

The research emphasized the effectiveness of the proposed IoT-based desalination monitoring system. It highlights the benefits of fault detection, system prevention, and cost reduction, reinforcing the significance of the proposed system's efficacy compared to others. The conclusion also points towards future research directions, potentially exploring trust evaluation frameworks and real-world cloud contexts.

The setup proposed by Murugan et al. is also capable of sending out automated SMS messages to concerned personnel whenever it detects malfunctions. Again, the use of SMS technology within an IoT system is similar to how the researchers of SAFER is using the said technology.

Leveraging the IoT in the operation of mining companies could greatly improve the operational and environmental issues that are commonly associated with the mining industry.

CHAPTER 3

METHODS

The researchers shall use the conceptual framework discussed in the previous chapter in the development of the system. The complete process flow of each major milestone of the research will be discussed in this chapter. The methods to be used by the researchers will require knowledge on hardware and software components, operational process of the host company, and statistics.

RESEARCH DESIGN

This study aims to design a system composed of substitution of manual data gathering process, writing software components, and integration of hardware components. The study will be applying a descriptive developmental research design with multi-phase iterative method as discussed by Leedy and Ormrod. The multi-phase iterative design is commonly used in design-based research wherein the researchers apply existing theories and knowledge to create effective systems and programs [31]. This study will involve four phases as illustrated in Figure 1.1 while each iteration will be comprised by the system design and compliance evaluation.

PARTICIPANTS

The participants of this study shall compose of personnel coming from departments or divisions relevant to the chosen company's operation that utilizes heavy and light equipment. These would be operators of equipment and their respective supervisors and managers from the Technical Services Department (TSD) and the Mines Division.

The TSD personnel are responsible for attaching the device to the chosen equipment for the testing of the product. The operators are tasked to use the product and will

be utilizing the built-in software the researchers have developed and installed onto the device. The supervisors are responsible for the assignment of operators and equipment to the device, the data validation stage, and the real time monitoring of the equipment utilization. The managers shall observe the overall effects brought about by the product to the efficiency of the day-to-day operation of the company.

METHODOLOGY

The aforementioned milestones that were briefly discussed in the previous chapter's *Conceptual Framework* section will be discussed in full detail in this section. Each milestone shall represent one major part in the methodology of this research.

Formulation of Requirements

The functionalities and the features of the data collection system will be identified first. As the research progresses, the functionalities will be illustrated as use-case diagrams which are analogous to the design requirements [32]. The formulation of requirements will depend upon the existing process and systems implemented by the host company. The formulation of requirements is a way to identify the gaps in the company's process' efficiency that can be addressed by researcher's propose system.

System Design

Bulk of the work in this study is designing the system. The system shall address all requirements to be identified in the previous development milestone. An initial high-level design drafted by the authors is shown in Figure 3.1.

Hardware Selection

The researchers chose to utilize Raspberry Pi 3B+, a single-board computer (SBC) made by Raspberry Pi Foundation, as the device's processing unit due to its low-cost yet suf-

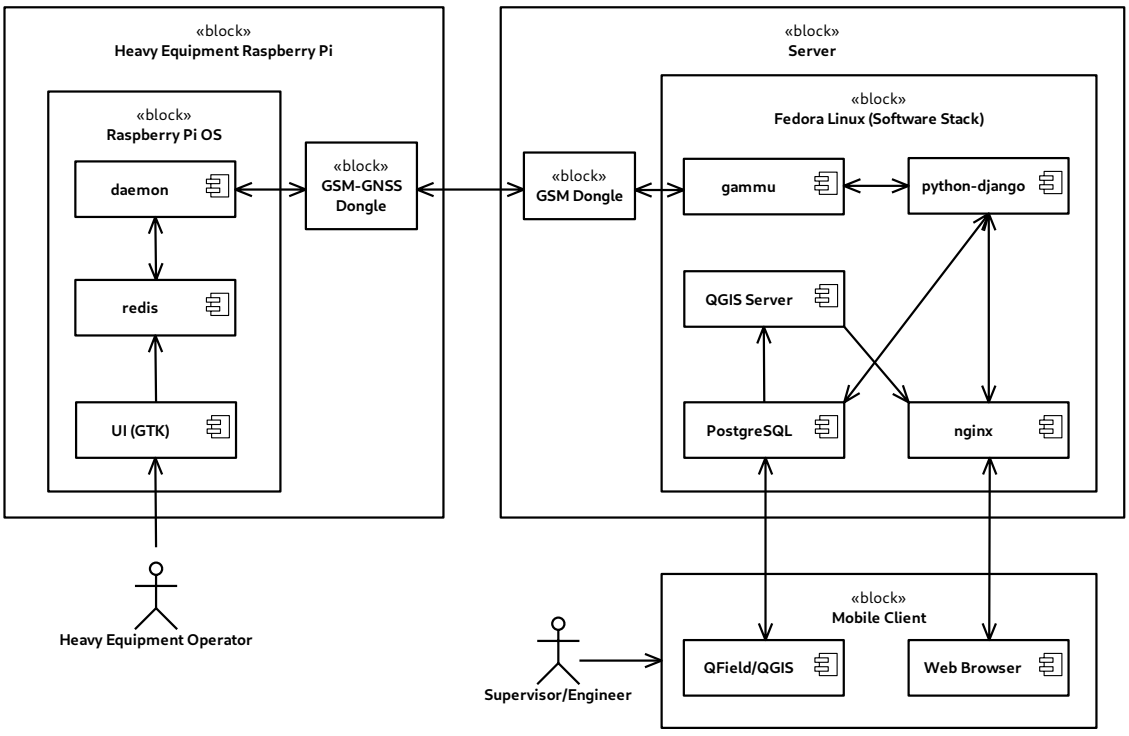

Figure 3.1: Conceptual level structure of the equipment usage data collection system.

ficiently capable hardware for the project. The low-cost of the computer will help chances of this project to have an increase in production count once a final product is demonstrated.

A 7-inch touchscreen monitor will be connected to the Raspberry Pi computers which will serve as the display for the graphical user interface, and will be receiving touch input from the operators depending on their current activity.

The researchers will be attaching commercially available integrated GSM-GNSS modules to the devices to act as receiver of location data, and sender of collected data to the central server.

A low-cost buck converter with constant voltage and current feature will also be utilized in order to control the device's voltage input to 5.1 volts, the official voltage recommendation by the Raspberry Pi Foundation [33], to protect the device from power fluctuations which may harm the device or corrupt the data.

A power management and UPS module, the Geekworm X708, is also used to provide UPS functionality, ensuring uninterrupted power during engine shut-off. It supports automatic safe shutdown, battery management, and power monitoring. Additionally, it features 5-volt output sockets for the 7" touchscreen, a restart switch socket, and GPIO extension to connect GNSS to Raspberry Pi.

A Real-Time Clock (RTC) module is also utilized to synchronize the device clock with the current time. This synchronization is crucial for the GNSS module to provide accurate positioning information.

Other electronic devices are also used to connect and communicate between components, such as the Raspberry Pi display flex cable, IDE ribbon cable, extension cables, and electrical wires. Additionally, shrinkable hose is used to insulate and cover joint wires,

and a wire jacket is used to make the wiring more presentable.

After a prototype has been developed, a fitted enclosure will be designed to make sure no dangling parts will be in the final form of the product. Blender, a 3D computer graphics tool will be used in designing the enclosure's 3D model. The model-file will be sent to a 3D printing facility and will then be used for final assembly of the hardware.

The device will be connected to the automobile auxiliary power outlet, also known as the car cigarette lighter, using a Car Cigarette Lighter/Auxiliary Charger, which outputs 12-24 volts depending on the type of vehicle. The voltage from the auxiliary power outlet will be converted down to 5.1 volts using a buck converter with constant current and voltage features.

On the server side, the researchers will be using a USB GSM module that will act as the receiver of the location and equipment utilization data sent by the device attached to the equipment. The GSM module will be interfaced to a virtual machine running Fedora Linux Server Edition hosted by a Lenovo ThinkSystem SR650 which is equipped with the second generation of Intel Xeon Processor Scalable Family (Xeon SP Gen 2).

Source Code Writing

The researchers will be using the official Raspberry Pi OS as the operating system for the device. For the Graphical User Interface (GUI) of the device, the researchers will be using GTK, a free and open-source widget toolkit for creating graphical user interfaces. Data from GUI application will be stored in redis, an in-memory high speed database [34]. A daemon to be written in C will be used in sending of location and equipment utilization. The daemon will be accessing data from the redis database and will be interfacing with the GNSS-GSM module via AT commands.

On the server side of the system, the server will be running Fedora Linux as its operating system. The researchers will still be using Python-Gammu to receive SMS text from the equipment. The data received will be processed and parsed by the server through Python scripts. The data will then be stored using PostgreSQL extended by PostGIS as the database management system for the study.

The graphical user interface for the supervisors will be made through Django, a high-level Python web framework, specifically through its Admin and Views features. QField, a mobile Geographic Information System (GIS) application and QGIS, a desk-top GIS application with direct interface to PostgreSQL database, may also be used by the engineers to access and view the real-time data of the equipment

System Testing

For convenience purpose, the researchers chose to conduct the device's testing in Taganito Mining Corporation (TMC), a nickel mining company operating in Taganito, Claver, Surigao del Norte, Philippines. TMC is also the workplace of all the aforementioned researchers of this study. Prior to the actual device pilot test, a meeting with the managers and personnel of the involved departments shall be conducted to introduce and explain the purpose of this study and the produced hardware. All questions about the study shall be addressed by the researchers in the meeting. This is illustrated in Figure 3.2.

After clearance is given by the managers of the concerned departments, the researchers shall request the service of the staff of Technical Services Department for the attachment of the device to the chosen equipment. Testing will be made, and feedback from the end-users shall be collected and considered for every new iteration of the system. Revisions will then be applied to the device and software, and testing will be continued,

and feedback will be collected once again. The cycle of testing-feedback-revision will continue until the end-users are satisfied with the experience and functionality brought by the device and the accompanying software. This process is illustrated in Figure 3.3

INSTRUMENTS

In the development of the input device, commercially available hardware components will be utilized. On the software side of the development, free and open source tools will be used. The specific hardware components and software tools to be used are already discussed in the system design in the Methodology section.

A similar method of evaluation will be used based on the ER Mine Tracer research design. The researchers of the said study used survey to qualitatively evaluate ER Mine Tracer's functionality and efficiency. In this study, the researchers will also employ surveys to gather qualitative data on the users' experience and opinion regarding the researcher's data collection and processing system [24].

Spreadsheet applications such Microsoft Excel and LibreOffice Calc will be used for data encoding and inspection while R version 4.3 [35] will be used for data manipulation and analysis.

DATA GATHERING PROCEDURE

Quantitative Data

The efficiency and effectiveness of SAFER will be evaluated in terms of data turnover time from the Mine Division to the Finance Department, the discrepancy between the actual SMR during the actual PMS and the predefined SMR, over-speeding detection of the current system, and idling equipment detection.

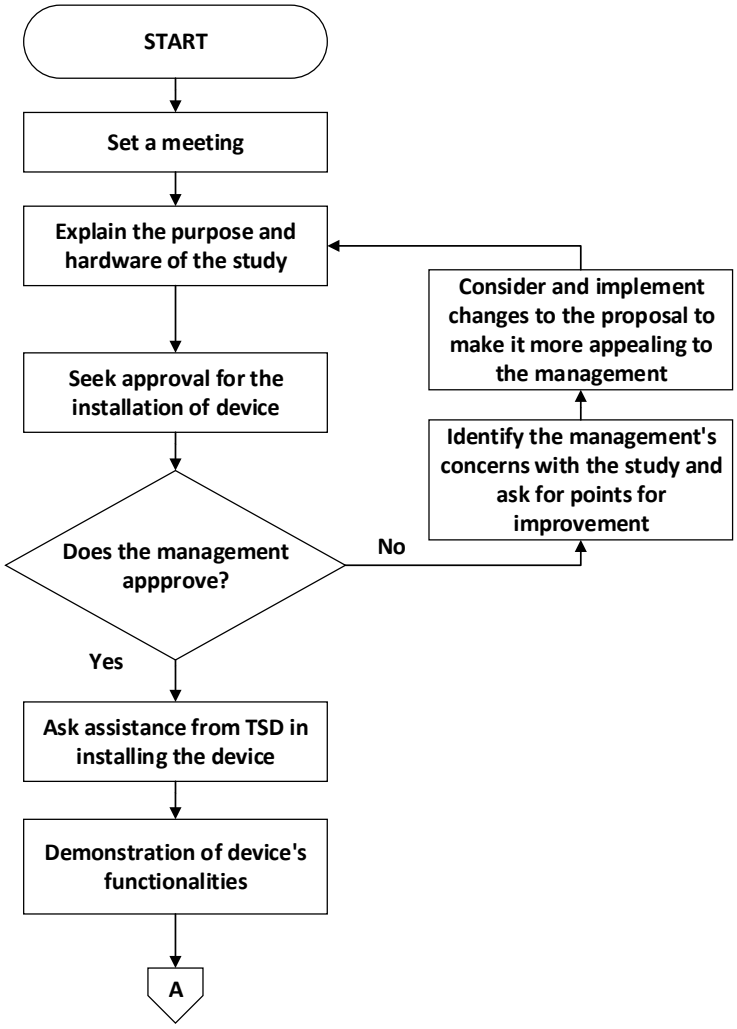

Figure 3.2: Pre-pilot testing procedure.

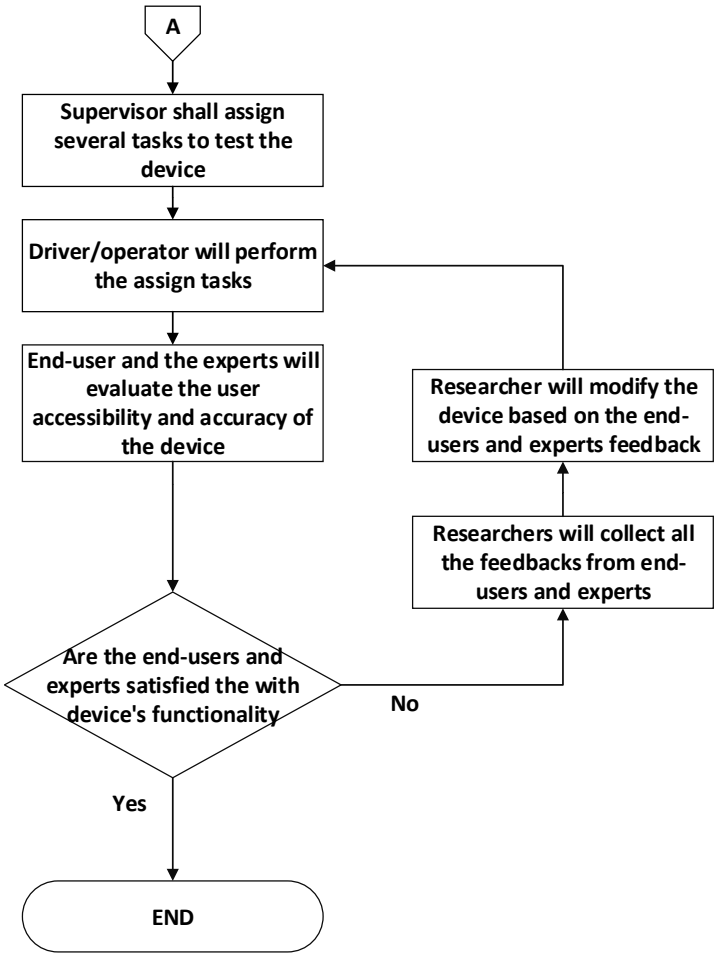

Figure 3.3: Feedback loop of the system development and functionality evaluation.

Qualitative Data

A Likert 4-point scales for "Agreement" parameter with 0.25 point increment will be basis of point system of the survey questionnaire. For each parameter, the verbal response and qualitative description provided by the system validators are used to generate the scale. Depending on the name of the performance exam, the first scale, or maximum, ranges from 3.25 to 4.00 with a parameter of 4 and a vocal response of "strongly agree". The second scale has a range of 2.5–3.25, a parameter of 3, an agree verbal response, and a qualitative description based on the name of the performance test. The third scale has a parameter of 2, a verbal response of "disagree," and a qualitative description based on the name of the performance test. Its range is 1.75-2.50.

The final scale has a range of 1.00 to 1.75, a parameter of 1, a verbal response of "strongly disagree," and a qualitative description based on the topic of the performance test [36].

DATA ANALYSIS

Descriptive statistics will be computed using quantitative data coming from SAFER during its testing. The performance of SAFER will not be compared to the existing system of TMC. The researchers will be using R version 4.3 [35] to analyze the quantitative data.

Questionnaire surveys with qualitative information will be distributed to different personnel on different departments and sections of the host company, IT Experts, Computer Engineers and other experts on this topic in order to collect data that are qualitative in nature.

The result of the surveys and the collected quantitative data will be used by the

researchers to determine if SAFER meets the user's requirements, meets the standard's of the technical experts, and does not have any technical limitations that might hinder its adoption.

ETHICAL CONSIDERATIONS

The data collected in this study shall be subject to the stipulations of the Republic Act 10173 or the Data Privacy Act of 2012. All the data gathering procedures shall be approved by the management of the host company, and shall be shared to them should they ask for it. Should there be instances where sensitive data may need to be included in publications, the company's approval shall be acquired beforehand.

CHAPTER 4

RESULTS AND DISCUSSIONS

HARDWARE DEVELOPMENT

Hardware Selection

Device Components

As mentioned in the software design, the researchers chose the Raspberry Pi 3B+ as the device's processing unit due to its low cost yet sufficiently capable hardware for the project. This low-cost device will increase the chances of this project's production count once a final product is demonstrated.

To ensure fast execution and efficient processing of data, the researchers utilize the SanDisk Extreme Pro SD Card. Specifically designed for high-performance applications, this SD card is intended for demanding tasks where speed and reliability are paramount. Whether handling large datasets, running complex algorithms, or storing multimedia files, the Extreme Pro SD Card's rapid read and write speeds ensure seamless operation, optimizing the performance of their Raspberry Pi-based projects. Its robust construction and dependable performance make it an ideal choice for research endeavors where data integrity and processing efficiency are crucial.

A 7-inch touchscreen monitor will serve as the display for the graphical user interface and will receive touch input from the operators depending on their current activity.

The researchers attached a commercially available integrated GSM/GPRS/GNSS/Bluetooth HAT, developed by Waveshare, a leading global provider of electronic components and modules. These devices will act as receivers of location data and senders of collected data to the central server.

A low-cost buck converter, the ZK-J5X with constant voltage and current features, was used in the device assembly. It can receive input ranging from 6.5V to 36V, while its output can range from 1.2V to 32V and can provide a stable current of 3.5A for a long time, reaching about 4.5A under enhanced heat dissipation. The advantage of this buck converter is that it can maintain its set output voltage despite fluctuations in the input voltage, making it the most suitable module to provide a Raspberry Pi with 5.1 volts which is the recommended voltage of the Raspberry Pi boards.

A power management and UPS module, the Geekworm X708, was used to provide an uninterruptible power supply (UPS) functionality, ensuring continuous power during engine shut-off. It supports automatic safe shutdown, battery management, and power monitoring. Additionally, it features 5-volt output sockets for the 7" touchscreen, a restart switch socket, and GPIO extension to connect GNSS to Raspberry Pi.

A Real-Time Clock (RTC) module was utilized to synchronize the device clock with the current time. This synchronization is crucial for the GNSS module to provide accurate positioning information.

Other electronic devices were also used to connect and communicate between components, such as the Raspberry Pi display flex cable, IDE ribbon cable, extension cables, and electrical wires. Additionally, shrinkable hose was used to insulate and cover joint wires, and a wire jacket is used to make the wiring more presentable.

The device will be connected to the automobile auxiliary power outlet, also known as the car cigarette lighter, using a car cigarette lighter or auxiliary charger, which outputs 12-24 volts depending on the type of vehicle. The voltage from the auxiliary power outlet will be converted down to 5.1 volts using a buck converter with constant current and voltage

features.

Server-side Components

On the server side, the researchers used a USB GSM dongle that will act as the receiver of the location and equipment utilization data sent by the device attached to the equipment. The GSM dongle utilizes SIM800C, a GSM module manufactured by SIM-Com Wireless Solutions Limited [37]. The GSM dongle is interfaced to a virtual machine running Fedora Linux Server Edition hosted by a Lenovo ThinkSystem SR650 which is equipped with the second generation of Intel Xeon Processor Scalable Family (Xeon SP Gen 2). The virtual machine instance was provided by TMC's ICT (Information and Communications Technology) Section.

Enclosure Design

For the 3D enclosure designing of the device, a digital 3D modeling software was used. After the researchers finished identifying the necessary components, they proceed to arrange those components and create a suitable 3D enclosure to easily access parts that are usually used, such as USB ports, the SD card slot, the reset switch, and place GSM and GPS SMA connectors outside the box for easy access. The 3D output file was converted to STL, which is a file format commonly accepted by 3D printers. The 3D printed enclosure has multiple layers intended for the following:

1. screen cover
2. screen holder
3. placement of the Raspberry Pi 3B+, Geekworm X708, GNSS module, buck converter, real-time clock, and reset switch
4. covers for battery, USB ports, reset switch, and SD card slot

5. stand to put the device on the dashboard of dump trucks, wheel loaders, and track excavators

The designed enclosure is shown in Figure 4.1 and Figure 4.2.

SOFTWARE DEVELOPMENT

Software Stack Selection

Device Software

Server-side Software

Software Design

Device Components

Server-side Components

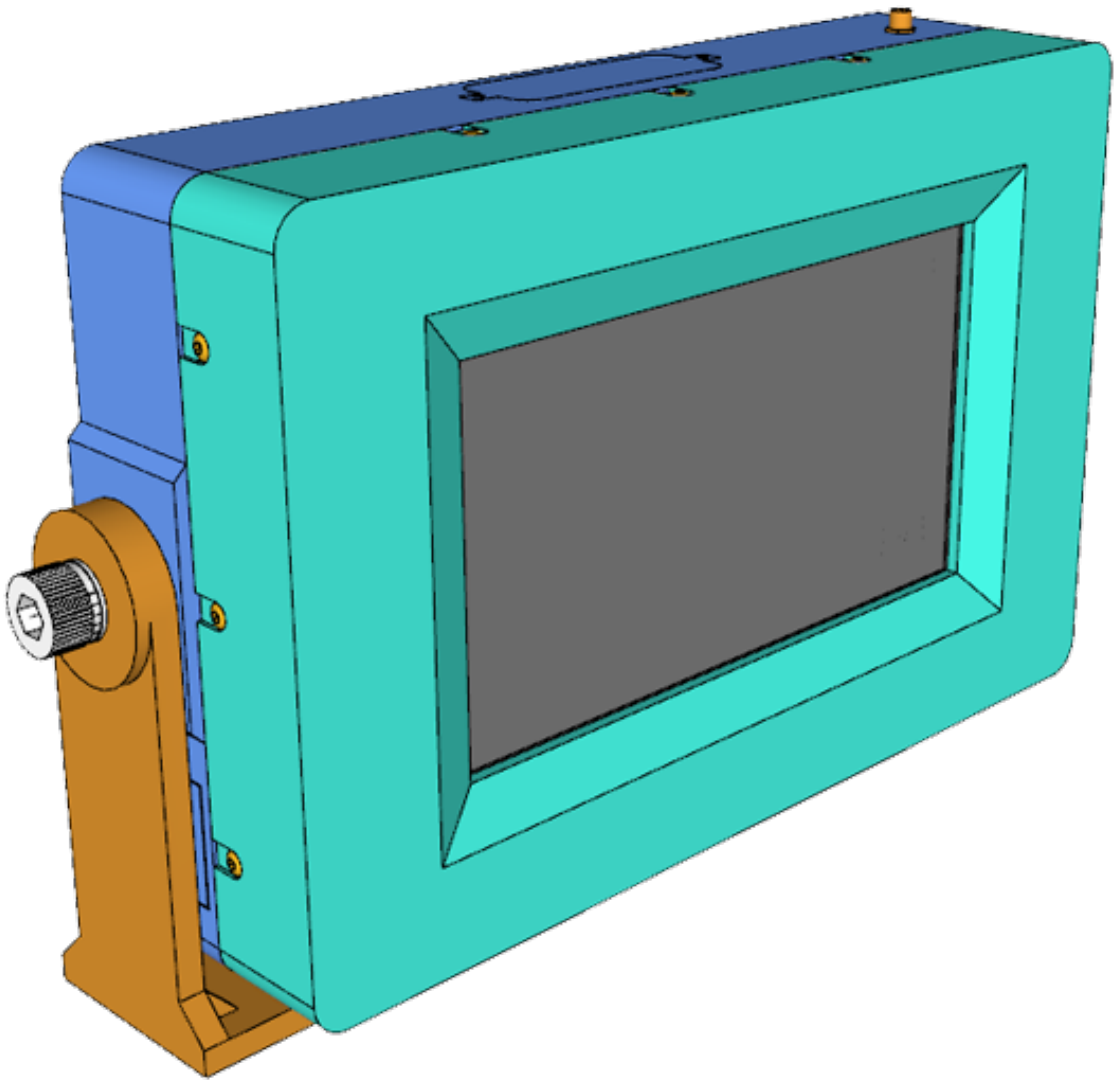

Figure 4.1: Perspective view of the enclosure design.

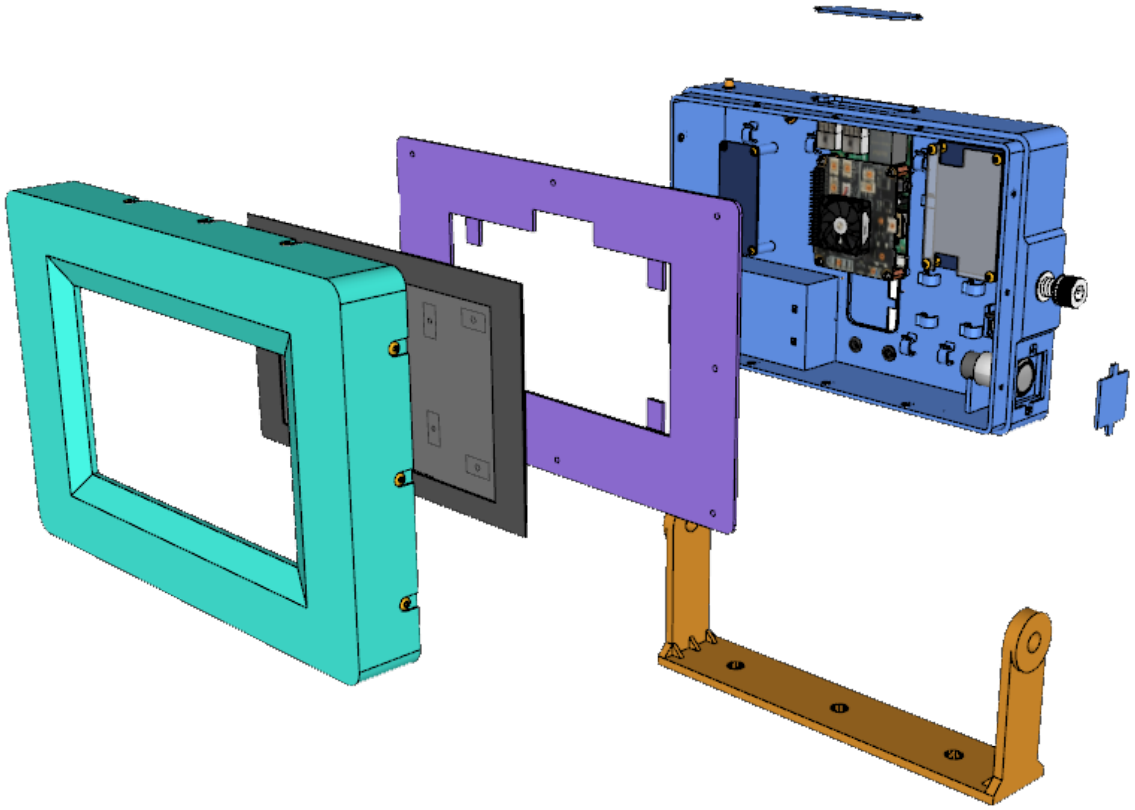

Figure 4.2: Exploded view of the enclosure design along with the hardware components.

REFERENCES

[1] A. Urhan, N. G. Inceb, R. Bondyb, and B. Alakenta, ``Soft-Sensor Design for a Crude Distillation Unit Using Statistical Learning Methods," *Computer Aided Chemical Engineering*, vol. 44, 2018. DOI: 10.1016/B978-0-444-64241-7.50373-6.

[2] L. Barnewold and B. G. Lottermoser, ``Identification of digital technologies and digitalisation trends in the mining industry," *International Journal of Mining Science and Technology*, vol. 30, 2020. DOI: 10.1016/j.ijmst.2020.07.003.

[3] S. Kumar, P. Tiwari, and M. Zymbler, ``Internet of Things is a revolutionary approach for future technology enhancement: a review," *Journal of Bigdata*, vol. 6, 2019. DOI: 10.1186/s40537-019-0268-2.

[4] S. S. Chandra, S. M. E. Sepasgozar, V. R. P. Kumar, A. K. Singh, L. Krishnaraj, and B. O. Awuzie, ``Assessing Factors Affecting Construction Equipment Productivity Using Structural Equation Modeling," *Buildings*, vol. 13, no. 2, 2023, ISSN: 2075-5309. DOI: 10.3390/buildings13020502.

[5] K. Radhika and R. Velmani, ``Bluetooth and GSM based Smart Security System using Raspberry Pi," *IOP Conference Series Materials Science and Engineering*, vol. 981, Dec. 2020. DOI: 10.1088/1757-899x/981/4/042009.

[6] EL-PRO-CUS. ``What are AT Commands : Different Types, and Their List." (2023), [Online]. Available: https://www.elprocus.com/at-commands-tutorial/ (visited on 04/15/2023).

[7] Blender Foundation. ``Blender for 3D Printing." (2023), [Online]. Available: https://studio.blender.org/training/3d-printing/ (visited on 04/15/2023).

[8] M. Kerrisk, ``The Linux Programming Interface," in William Pollock, 2010, ch. 37, pp. 768–782, ISBN 978-1-59327-220-3.

[9] M. Čihař and M. Wiącek. ``The Gammu Manual." (2020), [Online]. Available: https://docs.gammu.org/ (visited on 03/30/2023).

[10] S. Madry, ``Global Navigation Satellite Systems and Their Applications," *Springer-Briefs in Space Development*, 2015. DOI: 10.1007/978-1-4939-2608-4.

[11] J. Eberspächer, H.-J. Vögel, C. Bettstetter, and C. Hartmann, ``Gsm - architecture, protocols and services," in John Wiley & Sons, Ltd., 2009, ch. 1, pp. 1–2, ISBN 978-0-470-03070-7.

[12] GTK Development Team. ``The GTK toolkit." (2023), [Online]. Available: https://docs.gtk.org/gtk3/ (visited on 03/30/2023).

[13] S. Al-Salem, S. Uddin, and F. Al-Yamani, ``An assessment of microplastics threat to the marine environment: A short review in context of the Arabian/Persian Gulf," Jul. 2020. DOI: 10.1016/j.marenvres.2020.104961.

[14] PostGIS Project Steering Committee. ``PostGIS, spatial and geographic objects for PostgreSQL." (2022), [Online]. Available: https://postgis.net/ (visited on 03/30/2023).

[15] The PostgreSQL Global Development Group. ``PostgreSQL 15.2 Documentation." (2023), [Online]. Available: https://www.postgresql.org/docs/current/index.html (visited on 03/30/2023).

[16] M. Masih-Tehrani, S. Ebrahimi-Nejad, and M. Dahmardeh, ``Combined fuel consumption and emission optimization model for heavy construction equipment," *Automation in Construction*, vol. 110, Feb. 2020. DOI: 10.1016/j.autcon.2019.103007.

[17] J. Gillies, V. Etyemezian, H. Kuhns, D. Nikolic, and D. Gillette, ``Effect of vehicle characteristics on unpaved road dust emission," *Atmospheric Environment*, vol. 39, pp. 2341–2347, Apr. 2005. DOI: 10.1016/j.atmosenv.2004.05.064.

[18] P. Murphy and A. Morris, ``Quantifying accident risk and severity due to speed from the reaction point to the critical conflict in fatal motorcycle accidents," *Accident Analysis & Prevention*, vol. 141, Jun. 2020. DOI: 10.1016/j.aap.2020.105548.

[19] Department of Environment and Natural resources, ``Mine safety and health standards," Philippines, DENR Administrative Order, 2000.

[20] T. Wang, ``Analysis on Tyre Wear: Modelling and Simulations," Bachelor's Thesis, KTH Royal Institure of Technology, School of Engineering Sciences, 2017.

[21] Y. Andersson-Sköld, M. Johannesson, M. Gustafsson, *et al.*, ``Microplastics from tyre and road wear a literature review," May 2020. DOI: 10.13140/RG.2.2.34478.54083.

[22] S. Jiang, B. Li, and Y. Shen, ``The Influence of Pulp and Paper Industry on Environment," *E3S Web of Conferences*, vol. 308, p. 02 007, 2021. DOI: 10.1051/e3sconf/202130802007.

[23] A. Salam, ``Internet of things for sustainable mining," in *Internet of Things for Sustainable Community Development: Wireless Communications, Sensing, and Systems*. Cham: Springer International Publishing, 2020, pp. 243–271, ISBN: 978-3-030-35291-2. DOI: 10.1007/978-3-030-35291-2_8.

[24] A. O. Araojo, J. E. Arro, J. Lauro, A. V. P. Guilalas, J. C. Cano, and E. D. Gillo, ``Design and Development of ER MineTracer: A Mobile Emergency Response Application for Mining Industry," *International Journal of Current Science Research and Review*, vol. 06, 2023. DOI: 10.47191/ijcsrr/V6-i2033.

[25] R. Crozier, ``Rio Tinto to build new 'intelligent' mines," *itnews*, Jun. 2018. [Online]. Available: https://www.itnews.com.au/news/rio-tinto-to-build-new-intelligent-mines-494651.

[26] R. Crozier, ``Rio Tinto orders robot trucks, drills for digital mine," *itnews*, May 2019. [Online]. Available: https://www.itnews.com.au/news/rio-tinto-orders-robot-trucks-drills-for-digital-mine-524759.

[27] Rio Tinto. ``Finding better ways." (), [Online]. Available: https://www.riotinto.com/en/about/innovation (visited on 04/11/2023).

[28] S. V. H. Aguirre-Jofre M. Eyre, ``Low-cost internet of things (iot) for monitoring and optimising mining small-scale trucks and surface mining shovels," *Automation in Construction*, vol. 131, Nov. 2021. DOI: 10.1016/j.autcon.2021.103918.

[29] A. H. M. O. H. Nafis Sadique Sayem Sagor Chowdhury, ``Iot-based smart protection system to address agro-farm security challenges in bangladesh,'' *Smart Agricultural Technology*, vol. 6, Nov. 2023. DOI: 10.1016/j.autcon.2021.103918.

[30] P. S. T. Maris Murugan R. Kiruba Shankar, ``Monitoring and controlling the desalination plant using iot,'' *Measurement: Sensors*, vol. 27, Feb. 2023. DOI: 10.1016/j.measen.2023.100720.

[31] P. D. Leedy and J. E. Ormrod, ``Practical Research: Planning and Design,'' in Pearson Education, 2019, ch. 9, pp. 298–299, ISBN 978-0-13-477565-4.

[32] Y. Waykar, ``Role of use case diagram in software development,'' *International Journal of Management and Economics*, Jan. 2015.

[33] Raspberry Pi Foundation. ``Raspberry Pi Documentation - Getting started.'' (2023), [Online]. Available: https://www.raspberrypi.com/documentation/computers/getting-started.html (visited on 03/30/2023).

[34] R. K. Singh and H. K. Verma, ``Redis-Based Messaging Queue and Cache-Enabled Parallel Processing Social Media Analytics Framework,'' *The Computer Journal*, May 2020. DOI: 10.1093/comjnl/bxaa114.

[35] R Core Team, *R: A language and environment for statistical computing*, R Foundation for Statistical Computing, Vienna, Austria, 2023. [Online]. Available: https://www.R-project.org/.

[36] R. Likert, ``A technique for the measurement of attitudes,'' in R.S. WOODWORTH, 1932, ch. 4, pp. 21–32.

[37] SIMCom Wireless Solutions Limited. ``SIM800C." (2024), [Online]. Available: https://www.simcom.com/product/SIM800C.html (visited on 04/07/2024).

Appendices

SOURCE CODE

DEVICE CODE

File Tree

```
├── meson.build
├── python
│   ├── bat.py
│   ├── bat.service
│   └── hat.py
├── README.md
├── src
│   ├── gui.c
│   ├── gui_functions.c
│   ├── gui_functions.h
│   ├── gui.h
│   ├── main.c
│   ├── power-monitor.c
│   ├── power-monitor.h
│   ├── utils.c
│   ├── utils.h
│   ├── worker.c
│   └── worker.h
├── theme.css
└── x708
    ├── install.sh
    ├── x708-pwr.service
    ├── x708-pwr.sh
    └── x708-softsd.sh
```

meson.build

```
project('data_management_equipment_utilization_capture', 'c', version: '0.1.0')

gtk_dep = dependency('gtk+-3.0', required: true)
hiredis_dep = dependency('hiredis', required: true)
```

```
pcre_dep = dependency('libpcre2-8', required: true)
python_dep = dependency('python3', modules: ['redis'], required: true)
systemd_dep = dependency('libsystemd', required: true)

cmd = run_command('bash', '-c', 'echo $HOME', check: false)
home_dir = cmd.stdout().strip()

executable(
  'safer',
  [
    'src/gui.c',
    'src/gui.h',
    'src/gui_functions.c',
    'src/gui_functions.h',
    'src/main.c',
    'src/utils.h',
    'src/utils.c',
    'src/worker.h',
    'src/worker.c'
  ],
  dependencies : [gtk_dep, hiredis_dep, pcre_dep, python_dep, systemd_dep],
  install: true,
  install_dir: home_dir / 'bin'
)
install_data('python/bat.py', install_dir: home_dir / 'bin')
install_data('python/bat.service', install_dir: home_dir / '.config/systemd/user')
install_data('python/hat.py', install_dir: home_dir / 'bin')
install_data('theme.css', install_dir: home_dir)

run_target(
  'cppcheck',
  command : [
    'cppcheck',
    '--enable=all',
```

```
    '--suppress=missingIncludeSystem',
    '--project=' + join_paths(meson.build_root(),
    'compile_commands.json')
  ]
)
```

bat.py

```
#!/usr/bin/python3

import redis
import struct
import smbus
import sys
import time
import RPi.GPIO as GPIO

I2C_ADDR = 0x36

GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
GPIO.setup(6, GPIO.IN)
GPIO.setup(13, GPIO.OUT)

r = redis.Redis(host='localhost', port=6379, decode_responses=True)

def readVoltage(bus):
    address = I2C_ADDR
    read = bus.read_word_data(address, 2)
    swapped = struct.unpack("<H", struct.pack(">H", read))[0]
    voltage = swapped * 1.25 /1000/16
    return voltage

def readCapacity(bus):
    address = I2C_ADDR
```

```
    read = bus.read_word_data(address, 4)
    swapped = struct.unpack("<H", struct.pack(">H", read))[0]
    capacity = swapped/256
    if capacity > 100:
        capacity = 100
    return int(capacity)

bus = smbus.SMBus(1)

while True:
    r.set('voltage', f'{readVoltage(bus):.2f}')
    r.set('battery', f'{readCapacity(bus)}')
    if GPIO.input(6):
        r.set('pre_shutdown', '1')
        t = time.localtime()
        current_time = time.strftime('%Y-%m-%d-%H:%M:%S', t)
        r.set('pre_shutdown_time', current_time)
    else:
        r.set('pre_shutdown', '0')
    time.sleep(2)
```

bat.service

```
[Unit]
Description=Run Battery Monitor

[Service]
Type=simple
ExecStart=/home/datamanagement/bin/bat.py
Restart=always
RestartSec=5

[Install]
WantedBy=default.target
```

hat.py

```
#!/usr/bin/python3

import redis
import RPi.GPIO as gpio
import serial
import time

ser = serial.Serial("/dev/ttyS0", 115200)

def cmd(command):
    ser.write(f"{command}\r\n".encode())
    time.sleep(0.1)
    return ser.read(ser.inWaiting()).decode()

if cmd('AT') == '':
    gpio.setmode(gpio.BCM)
    gpio.setup(4, gpio.OUT)
    time.sleep(2)
    gpio.output(4, False)
    gpio.cleanup()

ser.close()
r = redis.Redis(host='localhost', port=6379, decode_responses=True)
r.set('hat', '1')
```

gui.c

```
#include "gui.h"
#include "gui_functions.h"
#include "utils.h"
#include "worker.h"

void activate(GtkApplication *app, gpointer data)
```

```
{
  pset *pointer_set = (pset *) data;
  GtkCssProvider *cssProvider;
  GtkWidget *boxActivity = gtk_box_new(GTK_ORIENTATION_VERTICAL, 5);
  GtkWidget *boxBrightness = gtk_box_new(GTK_ORIENTATION_VERTICAL, 5);
  GtkWidget *boxPersonnel = gtk_box_new(GTK_ORIENTATION_VERTICAL, 5);
  GtkWidget *boxPersonnelOperator = gtk_box_new(GTK_ORIENTATION_VERTICAL, 5);
  GtkWidget *boxPersonnelSupervisor = gtk_box_new(GTK_ORIENTATION_VERTICAL, 5);
  GtkWidget *boxSettings = gtk_box_new(GTK_ORIENTATION_VERTICAL, 5);
  GtkWidget *boxLabelBrightness = gtk_label_new("Brightness");
  GtkWidget *boxLabelOperator = gtk_label_new("Operator");
  GtkWidget *boxLabelSupervisor = gtk_label_new("Supervisor");
  GtkWidget *buttonIdling = gtk_radio_button_new_with_label(NULL, "Idling");
  GtkWidget *buttonProduction;
  GtkWidget *buttonQueu;
  GtkWidget *buttonRefuelling;
  GtkWidget *buttonTravel;
  GtkWidget *buttonWarmup;
  GtkWidget *comboBoxOperator = gtk_combo_box_text_new();
  GtkWidget *comboBoxSupervisor = gtk_combo_box_text_new();
  GtkWidget *notebook = gtk_notebook_new();
  GtkWidget *labelBattery = gtk_label_new("Battery");
  GtkWidget *labelVoltage = gtk_label_new("Voltage");
  GtkWidget *levelBattery = gtk_label_new("—%");
  GtkWidget *levelVoltage = gtk_label_new("—V");
  GtkWidget *sliderBrightness = gtk_scale_new_with_range(GTK_ORIENTATION_HORIZONTAL, 0,
      100, 5);
  GtkWidget *tabLabelActivity = gtk_label_new("Activity");
  GtkWidget *tabLabelPersonnel = gtk_label_new("Personnel");
  GtkWidget *tabLabelSettings = gtk_label_new("Settings");
  GtkWidget *tablePower = gtk_grid_new();
  GtkWidget *window = gtk_application_window_new(app);
  guint boxPacking = 0;
```

```
power_stat powerStatus = {
  .mutex = G_MUTEX_INIT,
  .battery = NULL,
  .voltage = NULL,
};
g_mutex_init(&powerStatus.mutex);

populate_comboboxtext(GTK_COMBO_BOX_TEXT(comboBoxOperator), "operators", pointer_set->
    context);
populate_comboboxtext(GTK_COMBO_BOX_TEXT(comboBoxSupervisor), "supervisors",
    pointer_set->context);

gtk_container_add(GTK_CONTAINER(window), notebook);
gtk_notebook_append_page(GTK_NOTEBOOK(notebook), boxPersonnel, tabLabelPersonnel);

gtk_box_pack_start(GTK_BOX(boxPersonnelOperator), boxLabelOperator, FALSE, FALSE,
    boxPacking);
gtk_box_pack_start(GTK_BOX(boxPersonnelOperator), comboBoxOperator, TRUE, TRUE,
    boxPacking);
gtk_box_pack_start(GTK_BOX(boxPersonnelSupervisor), boxLabelSupervisor, FALSE, FALSE,
    boxPacking);
gtk_box_pack_start(GTK_BOX(boxPersonnelSupervisor), comboBoxSupervisor, TRUE, TRUE,
    boxPacking);
gtk_box_pack_start(GTK_BOX(boxPersonnel), boxPersonnelOperator, TRUE, TRUE, boxPacking)
    ;
gtk_box_pack_start(GTK_BOX(boxPersonnel), boxPersonnelSupervisor, TRUE, TRUE,
    boxPacking);

gtk_notebook_append_page(GTK_NOTEBOOK(notebook), boxActivity, tabLabelActivity);

buttonProduction = gtk_radio_button_new_with_label_from_widget(GTK_RADIO_BUTTON(
    buttonIdling), "Production");
buttonQueu = gtk_radio_button_new_with_label_from_widget(GTK_RADIO_BUTTON(buttonIdling)
    , "Queuing");
```

```
buttonRefuelling = gtk_radio_button_new_with_label_from_widget(GTK_RADIO_BUTTON(
    buttonIdling), "Refueling");
buttonTravel = gtk_radio_button_new_with_label_from_widget(GTK_RADIO_BUTTON(
    buttonIdling), "Traveling/Repositioning");
buttonWarmup = gtk_radio_button_new_with_label_from_widget(GTK_RADIO_BUTTON(
    buttonIdling), "Warm-up/Cooling");

gtk_notebook_append_page(GTK_NOTEBOOK(notebook), boxSettings, tabLabelSettings);

gtk_box_pack_start(GTK_BOX(boxSettings), boxBrightness, TRUE, TRUE, boxPacking);
gtk_box_pack_start(GTK_BOX(boxBrightness), boxLabelBrightness, FALSE, FALSE, boxPacking
    );
gtk_box_pack_start(GTK_BOX(boxBrightness), sliderBrightness, TRUE, TRUE, boxPacking);

gtk_box_pack_start(GTK_BOX(boxSettings), tablePower, TRUE, TRUE, boxPacking + 5);
gtk_grid_set_column_homogeneous(GTK_GRID(tablePower), TRUE);
gtk_grid_set_row_homogeneous(GTK_GRID(tablePower), TRUE);
gtk_grid_attach(GTK_GRID(tablePower), labelBattery, 0, 0, 1, 1);
gtk_grid_attach(GTK_GRID(tablePower), labelVoltage, 0, 1, 1, 1);
gtk_grid_attach(GTK_GRID(tablePower), levelBattery, 1, 0, 1, 1);
gtk_grid_attach(GTK_GRID(tablePower), levelVoltage, 1, 1, 1, 1);

cssProvider = gtk_css_provider_new();
gtk_css_provider_load_from_path(cssProvider, pointer_set->css, NULL);
gtk_style_context_add_provider_for_screen(gdk_screen_get_default(), GTK_STYLE_PROVIDER(
    cssProvider), GTK_STYLE_PROVIDER_PRIORITY_USER);
gtk_widget_set_name(buttonIdling, "idle");
gtk_widget_set_name(buttonProduction, "production");
gtk_widget_set_name(buttonQueu, "queue");
gtk_widget_set_name(buttonRefuelling, "refuel");
gtk_widget_set_name(buttonTravel, "travel");
gtk_widget_set_name(buttonWarmup, "warm-up");
gtk_widget_set_name(comboBoxOperator, "operator");
gtk_widget_set_name(comboBoxSupervisor, "supervisor");
```

```
    gtk_box_pack_start(GTK_BOX(boxActivity), buttonProduction, TRUE, TRUE, boxPacking);
    gtk_box_pack_start(GTK_BOX(boxActivity), buttonQueu, TRUE, TRUE, boxPacking);
    gtk_box_pack_start(GTK_BOX(boxActivity), buttonTravel, TRUE, TRUE, boxPacking);
    gtk_box_pack_start(GTK_BOX(boxActivity), buttonRefuelling, TRUE, TRUE, boxPacking);
    gtk_box_pack_start(GTK_BOX(boxActivity), buttonIdling, TRUE, TRUE, boxPacking);
    gtk_box_pack_start(GTK_BOX(boxActivity), buttonWarmup, TRUE, TRUE, boxPacking);

    g_signal_connect(buttonIdling, "clicked", G_CALLBACK(toggle_status), pointer_set->
        context);
    g_signal_connect(buttonProduction, "clicked", G_CALLBACK(toggle_status), pointer_set->
        context);
    g_signal_connect(buttonQueu, "clicked", G_CALLBACK(toggle_status), pointer_set->context
        );
    g_signal_connect(buttonRefuelling, "clicked", G_CALLBACK(toggle_status), pointer_set->
        context);
    g_signal_connect(buttonTravel, "clicked", G_CALLBACK(toggle_status), pointer_set->
        context);
    g_signal_connect(buttonWarmup, "clicked", G_CALLBACK(toggle_status), pointer_set->
        context);
    g_signal_connect(comboBoxOperator, "changed", G_CALLBACK(toggle_personnel), NULL);
    g_signal_connect(comboBoxSupervisor, "changed", G_CALLBACK(toggle_personnel), NULL);

    gtk_toggle_button_set_active(GTK_TOGGLE_BUTTON(buttonWarmup), TRUE);
    gtk_container_set_border_width(GTK_CONTAINER(boxActivity), 5);
    gtk_widget_show_all(window);
    gtk_window_fullscreen(GTK_WINDOW(window));
    g_thread_new("HatThread", (GThreadFunc) hat, NULL);
    g_thread_new("PersonnelSenderThread", (GThreadFunc) personnel_sender, NULL);
    g_thread_new("PowerMonitorThread", (GThreadFunc) power_monitor, &powerStatus);
    g_thread_new("StatusSenderThread", (GThreadFunc) status_sender, NULL);
    g_thread_new("ShutdownWatcherThread", (GThreadFunc) shutdown_watcher, NULL);
    g_thread_new("ShutdownTriggerThread", (GThreadFunc) shutdown_trigger, NULL);
}
```

gui_functions.c

```
#include <systemd/sd-journal.h>
#include "gui_functions.h"
#include "utils.h"

void populate_comboboxtext(GtkComboBoxText *box,
                           const char      *list,
                           redisContext    *context)
{
  redisReply *personnel_list = redisCommand(context, "LRANGE %s 0 -1", list);
  if (personnel_list == NULL)
    sd_journal_send("MESSAGE=%s %s.", "Failed to get list of", list, "PRIORITY=%i",
        LOG_ERR, NULL);
  else {
    if (personnel_list->type == REDIS_REPLY_ARRAY) {
      for (int counter = 0; counter < personnel_list->elements; counter++)
        gtk_combo_box_text_append_text(box, personnel_list->element[counter]->str);
    }
    freeReplyObject(personnel_list);
  }
}

void toggle_personnel(GtkWidget *box,
                      gpointer   data)
{
  int shutdown = 0;
  get_int_key("shutdown", &shutdown);
  if (shutdown)
    return;
  if (gtk_combo_box_text_get_active_text(GTK_COMBO_BOX_TEXT(box)) == NULL)
    redis_cmd("SET", gtk_widget_get_name(box), "NONE");
  else
    redis_cmd("SET", gtk_widget_get_name(box), gtk_combo_box_text_get_active_text(
```

```
        GTK_COMBO_BOX_TEXT(box)));
  return;
}

void toggle_status(GtkWidget *button,
                   gpointer   data)
{
  redisContext *context = (redisContext *) data;
  int shutdown = 0;
  get_int_key("shutdown", &shutdown);
  if (shutdown)
    return;
  if (gtk_toggle_button_get_active(GTK_TOGGLE_BUTTON (button))) {
    redisReply *previous;
    previous = redisCommand(context, "GET equipment_status");
    if (previous == NULL) {
      sd_journal_send("MESSAGE=%s", "Failed to get redis response after pressing button."
          , "PRIORITY=%i", LOG_ERR, NULL);
      return;
    }
    if (previous->type == REDIS_REPLY_STRING) {
      redis_cmd("SET", "previous_equipment_status", previous->str);
      if (strcmp(previous->str, gtk_widget_get_name(button)) != 0) {
        redis_cmd("SET", "equipment_status", gtk_widget_get_name(button));
        redis_cmd("SET", "status_refresh", "1");
      }
    } else {
      redis_cmd("SET", "equipment_status", gtk_widget_get_name(button));
      redis_cmd("SET", "status_refresh", "1");
    }
    freeReplyObject(previous);
  }
}
```

gui_functions.h

```
#ifndef GUI_FUNCTIONS_H
#define GUI_FUNCTIONS_H

#include <gtk/gtk.h>
#include <hiredis/hiredis.h>

/**
 * populate_comboboxtext:
 * @box: a GtkComboBoxText
 * @list: the name of the list inside redis that will populate @box
 * @context: the redisContext to be used in connecting
 *
 * Populates the contents of @box.
 **/
void populate_comboboxtext(GtkComboBoxText *box,
                           const char      *list,
                           redisContext    *context);

/**
 * toggle_personnel:
 * @box: a GtkComboBoxText
 * @data: a redisContext to be used in connecting
 *
 * Sets the key in redis having the name of @box using the selected content of
 * @box.
 **/
void toggle_personnel(GtkWidget *box,
                      gpointer   data);

/**
 * toggle_status:
 * @button: a GtkToggleButton
```

```
 * @data: a redisContext to be used in connecting
 *
 * Sets the keys previous_equipment_status, equipment_status using the selected
 * @button.
 */
void toggle_status(GtkWidget *button,
                   gpointer   data);

/**
 * update_battery:
 * @label:
 * @data: a label_set
 *
 * Update the string of a GtkLabe.
 */
void update_label(gpointer data);

#endif // GUI_FUNCTIONS_H
```

gui.h

```
#ifndef GUI_H
#define GUI_H

#include <gtk/gtk.h>

/**
 * activate:
 * @app: a GtkApplication
 * @data: a pset to be used in activating @app
 *
 * Activates @app with all the UI components.
 */
void activate(GtkApplication *app,
              gpointer        data);
```

```
#endif // GUI_H
```

main.c

```
#define _GNU_SOURCE
#include <gtk/gtk.h>
#include <stdio.h>
#include <systemd/sd-journal.h>
#include "gui.h"
#include "utils.h"

/**
 * main:
 *
 * Main function
 *
 * Returns: exit status of application
 *          -1 if failed to initialize redisContext
 *          -2 if failed to set redis keys
 *          -3 if failed to get css
 *          -4 if failed to open serial connection to the GNSS/GSM HAT
 */
int main(int argc, char **argv) {
  GtkApplication *app;
  char *css;
  char serial_str[20];
  const char *home_dir = (char *) g_get_home_dir();
  int hat = 0;
  int status;
  int serial_file;

  if (asprintf(&css, "%s/theme.css", home_dir) < 0)
    return -3;
```

```
/* Activate HAT */
while (!hat) {
  system("hat.py");
  sleep(1);
  get_int_key("hat", &hat);
}

initialize_serial_connection(&serial_file);
if (serial_file < 0) {
  sd_journal_send("MESSAGE=Serial connection error.", "PRIORITY=%i", LOG_ERR, NULL);
  return -4;
} else
  snprintf(serial_str, sizeof(serial_str), "%d", serial_file);
app = gtk_application_new("com.nickelasia.tmc.datamanagement.safer",
    G_APPLICATION_DEFAULT_FLAGS);
redisContext *context = redisConnect("localhost", 6379);
if (context == NULL || context->err) {
  if (context) {
    sd_journal_send("MESSAGE=Connection error: %s", context->errstr, "PRIORITY=%i",
        LOG_ERR, NULL);
    redisFree(context);
  } else
    sd_journal_send("MESSAGE=%s", "Connection error: can't allocate redis context", "
        PRIORITY=%i", LOG_ERR, NULL);
  free(css);
  return -1;
}
if (!redis_cmd("SET", "hat", "0") ||
    !redis_cmd("SET", "shutdown", "0") ||
    !redis_cmd("SET", "pre_shutdown", "0") ||
    !redis_cmd("SET", "proceed_shutdown", "0") ||
    !redis_cmd("SET", "operator", "NONE") ||
    !redis_cmd("SET", "serial_file", serial_str) ||
    !redis_cmd("SET", "supervisor", "NONE")) {
```

```
        close(serial_file);
        free(css);
        redisFree(context);
        return -2;
      }
  pset pointer_set;
  pointer_set.context = context;
  pointer_set.css = css;
  while (1) {
    if (set_hat_time())
      break;
  }
  g_signal_connect(app, "activate", G_CALLBACK(activate), &pointer_set);
  status = g_application_run(G_APPLICATION(app), argc, argv);
  g_object_unref(app);
  free(css);
  redisFree(context);
  return status;
}
```

power-monitor.c

```
#include <glib.h>
#include "power-monitor.h"

struct _PowerMonitor
{
  GObject parent_instance;
  GMutex mutex;
  gchar *battery;
  gchar *voltage;
};

G_DEFINE_TYPE (PowerMonitor, power_monitor, G_TYPE_OBJECT)
```

```
enum {
  PROP_0,
  PROP_BATTERY,
  PROP_VOLTAGE,
  LAST_PROP
};

enum {
  BATTERY_LEVEL_CHANGED,
  VOLTAGE_LEVEL_CHANGED,
  LAST_SIGNAL
};

static GParamSpec *properties[LAST_PROP];

static guint signals[LAST_SIGNAL];

static void power_monitor_class_get_property(GObject    *object,
                                             guint       prop_id,
                                             GValue     *value,
                                             GParamSpec *pspec) {
  PowerMonitor *self = (PowerMonitor*) object;
  switch(prop_id) {
    case PROP_BATTERY:
      g_value_set_string(value, power_monitor_get_battery(self));
      break;
    case PROP_VOLTAGE:
      g_value_set_string(value, power_monitor_get_voltage(self));
      break;
  }
}

static void power_monitor_class_set_property(GObject      *object,
                                             guint         prop_id,
```

```
                                                const GValue *value,
                                                GParamSpec   *pspec) {
  PowerMonitor *self = (PowerMonitor*) object;
  switch(prop_id) {
    case PROP_BATTERY:
      power_monitor_set_battery(self, g_value_get_string(value));
      break;
    case PROP_VOLTAGE:
      power_monitor_set_voltage(self, g_value_get_string(value));
      break;
  }
}

static void power_monitor_class_init(PowerMonitorClass *klass) {
  GObjectClass *object_class = G_OBJECT_CLASS(klass);
  object_class->get_property = power_monitor_class_get_property;
  object_class->set_property = power_monitor_class_set_property;

  properties[PROP_BATTERY] =
    g_param_spec_string(
      "battery",
      "Battery",
      "Battery level",
      "---%",
      (G_PARAM_READWRITE | G_PARAM_STATIC_STRINGS)
    );
  properties[PROP_VOLTAGE] =
    g_param_spec_string(
      "voltage",
      "Voltage",
      "Voltage of power supply",
      "---V",
      (G_PARAM_READWRITE | G_PARAM_STATIC_STRINGS)
    );
```

```
  g_object_class_install_properties(object_class, LAST_PROP, properties);

  signals[BATTERY_LEVEL_CHANGED] =
    g_signal_new(
      "battery_level_changed",
      G_TYPE_FROM_CLASS(klass),
      G_SIGNAL_RUN_FIRST,
      0,
      NULL,
      NULL,
      NULL,
      G_TYPE_NONE,
      0
    );
  signals[VOLTAGE_LEVEL_CHANGED] =
    g_signal_new(
      "voltage_level_changed",
      G_TYPE_FROM_CLASS(klass),
      G_SIGNAL_RUN_FIRST,
      0,
      NULL,
      NULL,
      NULL,
      G_TYPE_NONE,
      0
    );
}

static void power_monitor_init(PowerMonitor *self) {}

void power_monitor_emit_battery_level_changed(PowerMonitor *self) {
  g_signal_emit(self, signals[BATTERY_LEVEL_CHANGED], 0);
}
```

```
void power_monitor_emit_voltage_level_changed(PowerMonitor *self) {
  g_signal_emit(self, signals[VOLTAGE_LEVEL_CHANGED], 0);
}

const gchar* power_monitor_get_battery(PowerMonitor *self) {
  return self->battery;
}

const gchar* power_monitor_get_voltage(PowerMonitor *self) {
  return self->voltage;
}

void power_monitor_set_battery(PowerMonitor *self,
                               const gchar  *batteryLevel) {
  if (g_strcmp0(batteryLevel, self->battery) != 0) {
    g_free(self->battery);
    self->battery = g_strdup(batteryLevel);
  }
}

void power_monitor_set_voltage(PowerMonitor *self,
                               const gchar  *voltageLevel) {
  if (g_strcmp0(voltageLevel, self->voltage) != 0) {
    g_free(self->voltage);
    self->voltage = g_strdup(voltageLevel);
  }
}
```

utils .c

```
#define PCRE2_CODE_UNIT_WIDTH 8

#include <fcntl.h>
#include <glib.h>
#include <pcre2.h>
```

```
#include <systemd/sd-journal.h>
#include <termios.h>
#include <time.h>
#include <unistd.h>
#include "utils.h"

int get_char_key(const char  *key,
                 char        **value)
{
  int out = 0;
  redisContext *context = redisConnect("localhost", 6379);
  if (context == NULL || context->err) {
    if (context) {
      sd_journal_send("MESSAGE=Connection error: %s", context->errstr, "PRIORITY=%i",
          LOG_ERR, NULL);
      redisFree(context);
    } else
      sd_journal_send("MESSAGE=%s", "Connection error: can't allocate redis context", "
          PRIORITY=%i", LOG_ERR, NULL);
    return out;
  }
  redisReply *reply = redisCommand(context, "GET %s", key);
  if (reply == NULL) {
    sd_journal_send("MESSAGE=Error getting key: %s", key, "PRIORITY=%i", LOG_ERR, NULL);
    redisFree(context);
    return out;
  }
  if (reply->type == REDIS_REPLY_STRING) {
    g_free(*value);
    int buffer_size = strlen(reply->str);
    *value = (char *) g_malloc(buffer_size * sizeof(char));
    strcpy(*value, reply->str);
    out = 1;
  }
```

```
  freeReplyObject(reply);
  redisFree(context);
  return out;
}

int get_int_key(const char *key,
                int        *value)
{
  int out = 0;
  redisContext *context = redisConnect("localhost", 6379);
  if (context == NULL || context->err) {
    if (context) {
      sd_journal_send("MESSAGE=Connection error: %s", context->errstr, "PRIORITY=%i",
          LOG_ERR, NULL);
      redisFree(context);
    } else
      sd_journal_send("MESSAGE=%s", "Connection error: can't allocate redis context", "
          PRIORITY=%i", LOG_ERR, NULL);
    return out;
  }
  redisReply *reply = redisCommand(context, "GET %s", key);
  if (reply == NULL) {
    sd_journal_send("MESSAGE=Error getting key: %s", key, "PRIORITY=%i", LOG_ERR, NULL);
    redisFree(context);
    return out;
  }
  if (reply->type == REDIS_REPLY_STRING) {
    *value = atoi(reply->str);
    out = 1;
  }
  freeReplyObject(reply);
  redisFree(context);
  return out;
}
```

```
int redis_cmd(const char *cmd,
              const char *key,
              const char *x)
{
  int out = 0;
  redisContext *context = redisConnect("localhost", 6379);
  if (context == NULL || context->err) {
    if (context) {
      sd_journal_send("MESSAGE=Connection error: %s", context->errstr, "PRIORITY=%i",
          LOG_ERR, NULL);
      redisFree(context);
    } else
      sd_journal_send("MESSAGE=%s", "Connection error: can't allocate redis context", "
          PRIORITY=%i", LOG_ERR, NULL);
    return out;
  }
  char *buffer;
  redisReply *reply;
  if (cmd && key && x) {
    buffer = (char *) malloc((strlen(cmd) + strlen(key) + strlen(x) + 3) * sizeof(char));
    sprintf(buffer, "%s %s %s", cmd, key, x);
    reply = redisCommand(context, "%s %s %s", cmd, key, x);
  }
  else if (cmd && key) {
    buffer = (char *) malloc((strlen(cmd) + strlen(key) + 2) * sizeof(char));
    sprintf(buffer, "%s %s", cmd, key);
    reply = redisCommand(context, "%s %s", cmd, key);
  }
  else {
    g_error("Invalid redis command.");
    return out;
  }
  if (reply == NULL) {
```

```
    sd_journal_send("MESSAGE=Error executing Redis command: %s", buffer, "PRIORITY=%i",
        LOG_ERR, NULL);
  } else {
    if (reply->type != REDIS_REPLY_ERROR)
      out = 1;
    freeReplyObject(reply);
  }
  free(buffer);
  redisFree(context);
  return out;
}

int send_equipment_status(redisContext *context)
{
  int output = 0;
  redisReply *status_queue = NULL;
  status_queue = redisCommand(context, "LRANGE messages 0 -1");
  if (status_queue == NULL) {
    sd_journal_send("MESSAGE=%s", "Failed to get status queue while attempting to send
        message.", "PRIORITY=%i", LOG_ERR, NULL);
    output = 1;
  } else {
    if (status_queue->type == REDIS_REPLY_ARRAY) {
      char captured_datetime[20];
      char captured_equipment_status[11];
      char captured_location[10];
      char datetime_substr[20];
      char *messages;
      char new_captured_datetime[20];
      char new_captured_equipment_status[11];
      char new_captured_location[10];
      int buffer_size;
      int counter;
      int total_characters = 0;
```

```
  for (counter = 0; counter < status_queue->elements; counter++)
    total_characters += strlen(status_queue->element[counter]->str);
  buffer_size = total_characters + (status_queue->elements * 2);
  messages = (char *) malloc(buffer_size * sizeof(char));
  memset(messages, 0, buffer_size * sizeof(char));
  strcpy(messages, status_queue->element[0]->str);
  strcat(messages, "\n");
  capture_pattern(status_queue->element[0]->str, captured_datetime,
      captured_equipment_status, captured_location);
  for (counter = 1; counter < status_queue->elements; counter++) {
    capture_pattern(status_queue->element[counter]->str, new_captured_datetime,
        new_captured_equipment_status, new_captured_location);
    str_difference(captured_datetime, new_captured_datetime, datetime_substr);
    strcat(messages, datetime_substr);
    strcat(messages, " ");
    if (strcmp(captured_equipment_status, new_captured_equipment_status) != 0) {
      strcat(messages, new_captured_equipment_status);
      strcat(messages, " ");
    }
    strcat(messages, new_captured_location);
    strcat(messages, "\n");
    strcpy(captured_datetime, new_captured_datetime);
    strcpy(captured_equipment_status, new_captured_equipment_status);
  }
  send_sms("All messages:");
  // TODO: return the status of sending messages and delete cache if sending
  // is success.
  send_sms(messages);
  free(messages);

  if (!redis_cmd("DEL", "messages", NULL))
    output = 3;
} else {
  output = 2;
```

```
    }
    freeReplyObject(status_queue);
  }
  return output;
}

int set_hat_time(void)
{
  char *rtc = NULL;
  char time_buffer[32];
  int out = 0;
  time_t current_time;
  time(&current_time);
  struct tm *time_print = localtime(&current_time);
  strftime(time_buffer, sizeof(time_buffer), "AT+CCLK=\"%y/%m/%d,%H:%M:%S+32\"",
      time_print);
  at_cmd(time_buffer, &rtc, 1);
  const char *success = strstr(rtc, "OK");
  if (success)
    out = 1;
  g_free(rtc);
  return out;
}

void at_cmd(const char     *cmd,
            char          **response,
            unsigned int    timeout)
{
  static GMutex mutex;
  g_mutex_lock(&mutex);
  char *at_command;
  char at_response[1024];
  int serial_file = -1;
  get_int_key("serial_file", &serial_file);
```

```
  if (serial_file < 0) {
    sd_journal_send("MESSAGE=Error establishing serial connection while running AT
        command: %s", cmd, "PRIORITY=%i", LOG_ERR, NULL);
    return;
  }
  at_command = (char *) g_malloc((strlen(cmd) + 3) * sizeof(char));
  if (at_command == NULL) {
    sd_journal_send("MESSAGE=AT Command allocation error: %s", cmd, "PRIORITY=%i",
        LOG_ERR, NULL);
    return;
  }
  strcpy(at_command, cmd);
  strcat(at_command, "\r\n");
  write(serial_file, at_command, strlen(at_command));
  sleep(timeout);
  ssize_t num_bytes = read(serial_file, at_response, sizeof(at_response));
  if (num_bytes > 0)
    at_response[num_bytes] = '\0';
  g_free(*response);
  *response = (char *) g_malloc(strlen(at_response) * sizeof(char));
  strcpy(*response, at_response);
  g_free(at_command);
  g_mutex_unlock(&mutex);
  return;
}

void capture_pattern(const char *source,
                     char       *datetime,
                     char       *status,
                     char       *location)
{
  const PCRE2_SPTR pattern = (PCRE2_SPTR) "^(\\d{4}-\\d{2}-\\d{2}-\\d{2}:\\d{2}:\\d{2})\\
      s+([a-z\\-]+)\\s+(LOCATION).*";
  int error_code;
```

```
int matches;
pcre2_code *re;
PCRE2_SIZE error_offset;
re = pcre2_compile(pattern, PCRE2_ZERO_TERMINATED, 0, &error_code, &error_offset, NULL)
    ;
if (re == NULL) {
  PCRE2_UCHAR buffer[256];
  pcre2_get_error_message(error_code, buffer, sizeof(buffer));
  sd_journal_send("MESSAGE=PCRE2 compilation failed at offset %d: %s", (int)
      error_offset, buffer, "PRIORITY=%i", LOG_ERR, NULL);
  return;
}
pcre2_match_data *match_data;
match_data = pcre2_match_data_create_from_pattern(re, NULL);
matches = pcre2_match(re, (PCRE2_SPTR) source, strlen(source), 0, 0, match_data, NULL);
if (matches < 0) {
  switch(matches) {
    case PCRE2_ERROR_NOMATCH:
      sd_journal_send("MESSAGE=%s", "PCRE2 no matches", "PRIORITY=%i", LOG_ERR, NULL);
      break;
    default:
      sd_journal_send("MESSAGE=PCRE2 matching error %d", matches, "PRIORITY=%i",
          LOG_ERR, NULL);
      break;
  }
  pcre2_match_data_free(match_data);
  pcre2_code_free(re);
  return;
}
const PCRE2_SIZE *ovector = pcre2_get_ovector_pointer(match_data);
PCRE2_SPTR substring_datetime = (PCRE2_SPTR) source + ovector[2];
strncpy(datetime, (char *) substring_datetime, ovector[3] - ovector[2]);
datetime[ovector[3] - ovector[2]] = '\0';
PCRE2_SPTR substring_status = (PCRE2_SPTR) source + ovector[4];
```

```
  strncpy(status, (char *) substring_status, ovector[5] - ovector[4]);
  status[ovector[5] - ovector[4]] = '\0';
  PCRE2_SPTR substring_location = (PCRE2_SPTR) source + ovector[6];
  strncpy(location, (char *) substring_location, ovector[7] - ovector[6]);
  location[ovector[7] - ovector[6]] = '\0';
  pcre2_code_free(re);
  pcre2_match_data_free(match_data);
  return;
}

void initialize_serial_connection(int *serial_file_descriptor)
{
  int baud_rate = B115200;
  *serial_file_descriptor = open("/dev/ttyS0", O_RDWR | O_NOCTTY);
  if (*serial_file_descriptor < 0)
    return;
  struct termios tty;
  if (tcgetattr(*serial_file_descriptor, &tty) != 0) {
    close(*serial_file_descriptor);
    *serial_file_descriptor = -1;
    return;
  }
  cfsetospeed(&tty, baud_rate);
  cfsetispeed(&tty, baud_rate);
  tty.c_cflag &= ~PARENB;
  tty.c_cflag &= ~CSTOPB;
  tty.c_cflag &= ~CSIZE;
  tty.c_cflag |= CS8;
  tty.c_cflag &= ~CRTSCTS;
  tty.c_cflag |= CREAD | CLOCAL;
  tty.c_lflag &= ~(ICANON | ECHO | ECHOE | ISIG);
  tty.c_cc[VMIN] = 0;
  tty.c_cc[VTIME] = 10;
  if (tcsetattr(*serial_file_descriptor, TCSANOW, &tty) != 0) {
```

```
    close(*serial_file_descriptor);
    *serial_file_descriptor = -1;
  }
  return;
}

void send_sms(const char *format,
              ...)
{
  va_list args;
  va_start(args, format);
  int len = vsnprintf(NULL, 0, format, args) + 1;
  va_end(args);
  char *text = (char *) g_malloc(len * sizeof(char));
  va_start(args, format);
  vsnprintf(text, len, format, args);
  va_end(args);
  // TODO: send actual SMS
  printf("%s\n", text);
  g_free(text);
  return;
}

void str_copy(char       **destination,
              const char  *source)
{
  g_free(*destination);
  *destination = (char *) g_malloc(strlen(source) * sizeof(char));
  strcpy(*destination, source);
  return;
}

void str_difference(const char *old,
                    const char *new,
```

```
                    char        *holder)
{
  int counter = 0;
  int idx;
  holder[0] = '\0';
  while (1) {
    if (old[counter] != new[counter])
      break;
    counter += 1;
  }
  for (idx = 0; counter < strlen(new); idx++) {
    holder[idx] = new[counter];
    counter += 1;
  }
  holder[idx] = '\0';
  return;
}

void str_sub(char        *destination,
             const char *source,
             const int   start,
             const int   end)
{
  int i, j = 0;
  for (i = start; i <= end; ++i)
    destination[j++] = source[i];
  destination[j] = '\0';
  return;
}
```

worker.c

```
#define _GNU_SOURCE

#include <hiredis/hiredis.h>
```

```
#include <stdio.h>
#include <systemd/sd-journal.h>
#include <time.h>
#include "power-monitor.h"
#include "worker.h"
#include "utils.h"

void hat()
{
  char *response = NULL;
  int serial_file = -1;
  int shutdown = 0;
  while (serial_file < 0)
    get_int_key("serial_file", &serial_file);
  at_cmd("AT+GSMBUSY=1", &response, 1);
  while (shutdown == 0) {
    sleep(1);
    get_int_key("shutdown", &shutdown);
    // TODO: gnss query
    // TODO: queued SMS sending
    // TODO: read incmoming SMS (single part, multi-part)
  }
  return;
}

void personnel_sender()
{
  char *operator = NULL;
  char *previous_operator = NULL;
  char *previous_supervisor = NULL;
  char *supervisor = NULL;
  const int seconds_refresh_cutoff = 300;
  int refresh = 0;
  int shutdown = 0;
```

```
time_t current_time;
time_t refresh_time;
time(&refresh_time);

while (1) {
  sleep(1);
  time(&current_time);
  get_int_key("shutdown", &shutdown);
  if (shutdown)
    return;
  if (!get_char_key("operator", &operator) || !get_char_key("supervisor", &supervisor))
    continue;
  if (g_strcmp0(operator, "NONE") == 0 || g_strcmp0(supervisor, "NONE") == 0)
    continue;
  if (g_strcmp0(operator, previous_operator) != 0) {
    refresh = 1;
    str_copy(&previous_operator, operator);
    time(&refresh_time);
  }
  if (g_strcmp0(supervisor, previous_supervisor) != 0) {
    refresh = 1;
    str_copy(&previous_supervisor, supervisor);
    time(&refresh_time);
  }
  if (refresh) {
    if (g_strcmp0(operator, previous_operator) == 0 && g_strcmp0(supervisor,
        previous_supervisor) == 0) {
      if ((refresh_time + seconds_refresh_cutoff) <= current_time) {
        send_sms("Operator: %s\nSupervisor:%s", operator, supervisor);
        refresh = 0;
        refresh_time = current_time;
      }
    }
  }
```

```
  }
}

void power_monitor(gpointer data)
{
  char *previous_battery = NULL;
  char *previous_voltage = NULL;
  char buffer[10];
  power_stat *powerStatus = (power_stat *) data;
  powerStatus->battery = NULL;
  powerStatus->voltage = NULL;
  while (1) {
    sleep(2);
    g_mutex_lock(&powerStatus->mutex);
    if (!get_char_key("battery", powerStatus->battery) || !get_char_key("voltage",
        powerStatus->voltage)) {
      g_mutex_unlock(&powerStatus->mutex);
      continue;
    }
    if (g_strcmp0(powerStatus->battery, previous_battery) != 0) {
      str_copy(&previous_battery, battery);

    }
    if (g_strcmp0(voltage, previous_voltage) != 0) {
      str_copy(&previous_voltage, voltage);
      sprintf(buffer, "%sV", voltage);
      gtk_label_set_label(GTK_LABEL(powerStatus->voltage), buffer);
    }
  }
}

void shutdown_trigger()
{
  int proceed_shutdown = 0;
```

```
  int serial_file = -1;
  while (1) {
    redisContext *context = redisConnect("localhost", 6379);
    if (context == NULL || context->err) {
      if (context) {
        sd_journal_send("MESSAGE=Connection error: %s", context->errstr, "PRIORITY=%i",
            LOG_ERR, NULL);
        redisFree(context);
      } else
        sd_journal_send("MESSAGE=%s", "Connection error: can't allocate redis context", "
            PRIORITY=%i", LOG_ERR, NULL);
    } else
      break;
  }
  while (1) {
    sleep(1);
    get_int_key("proceed_shutdown", &proceed_shutdown);
    if (proceed_shutdown) {
      if (!redis_cmd("SET", "proceed_shutdown", "0"))
        continue;
      get_int_key("serial_file", &serial_file);
      if (serial_file < 0)
        continue;
      close(serial_file);
      system("sudo /usr/local/bin/x708-softsd.sh");
    }
  }
}

void shutdown_watcher()
{
  redisContext *context;
  while (1) {
    context = redisConnect("localhost", 6379);
```

```
  if (context == NULL || context->err) {
    if (context) {
      sd_journal_send("MESSAGE=Connection error: %s", context->errstr, "PRIORITY=%i",
          LOG_ERR, NULL);
      redisFree(context);
    } else
      sd_journal_send("MESSAGE=%s", "Connection error: can't allocate redis context", "
          PRIORITY=%i", LOG_ERR, NULL);
  } else
    break;
}
// Number of seconds before the shutdown key is triggered.
const int seconds_refresh_cutoff = 5;
char time_buffer[20];
int pre_shutdown = 0;
struct tm *time_print;
time_t current_time;
time_t refresh_time;
time(&refresh_time);
while (1) {
  sleep(1);
  time(&current_time);
  get_int_key("pre_shutdown", &pre_shutdown);
  if (pre_shutdown) {
    if ((refresh_time + seconds_refresh_cutoff) <= current_time) {
      time_print = localtime(&refresh_time);
      strftime(time_buffer, sizeof(time_buffer), "%Y-%m-%d-%H:%M:%S", time_print);
      if (!redis_cmd("SET", "shutdown", "1") || !redis_cmd("SET", "shutdown_time",
          time_buffer))
        continue;
      else {
        redisFree(context);
        return;
      }
```

```
      }
    } else
      refresh_time = current_time;
  }

}

void status_sender()
{
  redisContext *context;
  while (1) {
    context = redisConnect("localhost", 6379);
    if (context == NULL || context->err) {
      if (context) {
        sd_journal_send("MESSAGE=Connection error: %s", context->errstr, "PRIORITY=%i",
            LOG_ERR, NULL);
        redisFree(context);
      } else
        sd_journal_send("MESSAGE=%s", "Connection error: can't allocate redis context", "
            PRIORITY=%i", LOG_ERR, NULL);
    } else
      break;
  }

  // Number of messages to be queued before sending
  const int message_limit = 10;
  // Number of seconds before the new equipment status is recorded to allow making
      mistakes
  const int seconds_refresh_cutoff = 5;
  // Number of seconds between location queries
  const int seconds_location_period = 30;

  char time_buffer[20];
  char *message;
```

```
char *equipment_status = NULL;
char *pre_shutdown_time = NULL;
char *previous_equipment_status = NULL;
const char *location = "LOCATION"; // Temporary placeholder for GNSS query
const char *last_status = "off";
int push_message_status = 0;
int refresh_status = 0;
int shutdown = 0;
struct tm *time_print;
time_t current_time;
time_t refresh_time;
time(&refresh_time);

while (1) {
  sleep(1);
  time(&current_time);
  redisReply *status_queue = redisCommand(context, "LRANGE messages 0 -1");
  if (!get_char_key("equipment_status", &equipment_status)
      || !get_char_key("previous_equipment_status", &previous_equipment_status)
      || status_queue == NULL)
    {
      sd_journal_send("MESSAGE=%s", "Failed to get redis response.", "PRIORITY=%i",
          LOG_ERR, NULL);
      if (status_queue != NULL)
        freeReplyObject(status_queue);
      continue;
    }
  get_int_key("status_refresh", &refresh_status);
  get_int_key("shutdown", &shutdown);
  if (shutdown) {
    if (!get_char_key("pre_shutdown_time", &pre_shutdown_time)) {
      freeReplyObject(status_queue);
      continue;
    }
```

```
  if (asprintf(&message, "%s %s %s", pre_shutdown_time, last_status, location) < 0) {
    freeReplyObject(status_queue);
    continue;
  }
  push_message_status = redis_cmd("RPUSH", "messages", message);
  free(message);
  if (!push_message_status) {
    freeReplyObject(status_queue);
    continue;
  }
}

// TODO: GNSS query

// Send SMS when the message queue limit is reached or there is shutdown signal.
if (status_queue->type == REDIS_REPLY_ARRAY && (status_queue->elements >=
    message_limit || shutdown)) {
  if (send_equipment_status(context) != 0)
    continue;
  if (shutdown) {
    if (!redis_cmd("SET", "proceed_shutdown", "1"))
      continue;
  }
}

if (!shutdown) {
  if (refresh_status) {
    if (g_strcmp0(previous_equipment_status, equipment_status) == 0) {
      if ((refresh_time + seconds_refresh_cutoff) <= current_time) {
        if (!redis_cmd("SET", "status_refresh", "0")) {
          freeReplyObject(status_queue);
          continue;
        }
        time_print = localtime(&refresh_time);
```

```
      strftime(time_buffer, sizeof(time_buffer), "%Y-%m-%d %H:%M:%S", time_print);
      if (asprintf(&message, "%s %s %s", time_buffer, equipment_status, location) <
          0) {
        freeReplyObject(status_queue);
        continue;
      }
      push_message_status = redis_cmd("RPUSH", "messages", message);
      free(message);
      if (!push_message_status) {
        freeReplyObject(status_queue);
        continue;
      }
      refresh_time = current_time;
    }
  } else {
    if (!redis_cmd("SET", "previous_equipment_status", equipment_status)) {
      freeReplyObject(status_queue);
      continue;
    }
    refresh_time = current_time;
  }
} else {
  if ((refresh_time + seconds_location_period) <= current_time) {
    // TODO: GNSS query
    time_print = localtime(&current_time);
    strftime(time_buffer, sizeof(time_buffer), "%Y-%m-%d %H:%M:%S", time_print);
    if (asprintf(&message, "%s %s %s", time_buffer, equipment_status, location) <
        0) {
      freeReplyObject(status_queue);
      continue;
    }
    push_message_status = redis_cmd("RPUSH", "messages", message);
    free(message);
    if (!push_message_status) {
```

```
            freeReplyObject(status_queue);
            continue;
          }
          refresh_time = current_time;
        }
      }
    }
    freeReplyObject(status_queue);

    if (shutdown)
      break;
  }
  g_free(equipment_status);
  g_free(pre_shutdown_time);
  g_free(previous_equipment_status);
  redisFree(context);
  return;
}
```

worker.h

```
/**
 * This contains functions which are constantly running throughout the life
 * of the main process. All of these functions will terminate when the shutdown
 * status is fetched from redis.
 */

#ifndef WORKER_H
#define WORKER_H

#include <glib.h>

/**
 * hat:
 *
```

```
 * A worker thread for dropping incoming calls, querying location, processing
 * received SMS, and sending queued SMS.
 */
void hat();

/**
 * personnel_sender:
 *
 * A separate thread for watching the operator and supervisor keys. If there
 * is an update in these keys, send an SMS.
 */
void personnel_sender();

/**
 * power_monitor:
 *
 * A thread for regulary querying the power statistics.
 */
void power_monitor(gpointer data);

/**
 * shutdown_trigger:
 *
 * Triggers the actual system shutdown after all equipment status are sent. The
 * following redis keys are triggered in order:
 * pre_shutdown: triggered by ignition off
 * shutdown: triggered if ignition is still off within a certain duration
 * proceed_shutdown: triggered when all messages are sent
 */
void shutdown_trigger();

/**
 * shutdown_watcher:
 *
```

```
 * A separate thread for watching the pre_shutdown key which will be triggered
 * when the ignition is off. Should the pre_shutdown key remains on for the set
 * period of time, the shutdown key will be set to TRUE. The pre_shutdown key is
 * needed to account for temporary disconnection as experienced in the devices
 * from Manila GPS.
 *
 * TODO: When shutting down, the time should be recorded so when booting up
 * again, the time difference will be analyzed to check if the real-time clock
 * is still functioning properly.
 */
void shutdown_watcher();

/**
 * status_sender:
 *
 * A background worker which will send the cached equipment status in redis to
 * the server. This will also queues equipment statuses to redis at the defined
 * time interval in the function.
 */
void status_sender();

/* sms_receiver:
 *
 * Required use cases:
 * 1. updating operators
 * 2. updating supervisors
 * 3. pms notification
 *
 * Nice to have use cases:
 * 1. habitual over speeding notification
 * 2. deployment/reassignment notification
 */
void sms_receiver();
```

```
#endif // WORKER_H
```

theme.css

```
* {
  border-width: 10px;
  font-size: 32px;
}
```

install .sh

```
#!/usr/bin/bash -x

if [[ "${UID}" -ne 0 ]]
then
  echo "This script must be run as root or with sudo."
  exit 1
fi

cp x708-pwr.service /lib/systemd/system/
cp x708-pwr.sh /usr/local/bin/
cp x708-softsd.sh /usr/local/bin/
systemctl daemon-reload
systemctl enable --now x708-pwr.service
```

x708-pwr.service

```
[Unit]
Description=Run Hardware Power Management & Safe Shutdown daemon

[Service]
Type=simple
ExecStart=/usr/local/bin/x708-pwr.sh
Restart=always

[Install]
```

```
WantedBy=multi-user.target
```

x708-pwr.sh

```
#!/bin/bash

SHUTDOWN=5
REBOOTPULSEMINIMUM=200
REBOOTPULSEMAXIMUM=600
echo "$SHUTDOWN" > /sys/class/gpio/export
echo "in" > /sys/class/gpio/gpio$SHUTDOWN/direction
BOOT=12
echo "$BOOT" > /sys/class/gpio/export
echo "out" > /sys/class/gpio/gpio$BOOT/direction
echo "1" > /sys/class/gpio/gpio$BOOT/value

echo "Your device are shutting down..."

while [ 1 ]; do
  shutdownSignal=$(cat /sys/class/gpio/gpio$SHUTDOWN/value)
  if [ $shutdownSignal = 0 ]; then
    /bin/sleep 0.2
  else
    pulseStart=$(date +%s%N | cut -b1-13)
    while [ $shutdownSignal = 1 ]; do
      /bin/sleep 0.02
      if [ $(($(date +%s%N | cut -b1-13)-$pulseStart)) -gt $REBOOTPULSEMAXIMUM ]; then
        echo "Your device are shutting down", SHUTDOWN, ", halting Rpi ..."
        sudo poweroff
        exit
      fi
      shutdownSignal=$(cat /sys/class/gpio/gpio$SHUTDOWN/value)
    done
    if [ $(($(date +%s%N | cut -b1-13)-$pulseStart)) -gt $REBOOTPULSEMINIMUM ]; then
      echo "Your device are rebooting", SHUTDOWN, ", recycling Rpi ..."
```

```
        sudo reboot
        exit
      fi
    fi
done
```

x708-softsd.sh

```
#!/bin/bash

BUTTON=13

echo "$BUTTON" > /sys/class/gpio/export;
echo "out" > /sys/class/gpio/gpio$BUTTON/direction
echo "1" > /sys/class/gpio/gpio$BUTTON/value

SLEEP=${1:-4}

re='^[0-9\.]+$'
if ! [[ $SLEEP =~ $re ]] ; then
   echo "error: sleep time not a number" >&2; exit 1
fi

echo "Your device will shutting down in 4 seconds..."
/bin/sleep $SLEEP

# restore GPIO 13
echo "0" > /sys/class/gpio/gpio$BUTTON/value
```

SERVER-SIDE CODE

CURRICULUM VITAE

BASIL ERIC C. RABI
P-4, Magosilom, Cantilan, Surigao del Sur
ericbasil.rabi@gmail.com

PERSONAL INFORMATION

DATE OF BIRTH : January 17, 1991
PLACE OF BIRTH : Trinidad, Calbayog City, Samar
AGE : 32
GENDER : Male
NATIONALITY : Filipino
RELIGION : Roman Catholic
CIVIL STATUS : Married
FATHER'S NAME : Ricardo A. Rabi
MOTHER'S NAME : Ester C. Rabi

EDUCATIONAL BACKGROUND

COLLEGE : Saint Paul University Surigao
: Corner Rizal and San Nicolas Streets, Surigao City
HIGH SCHOOL : Philippine Science High School
: Quezon City, Metro Manila
ELEMENTARY : Trinidad Elementary School
: Trinidad, Calbayog City, Samar

JAMES M. PAJE
Purok 1, Ladgaron, Claver, Surigao del Norte
xkitzie23@gmail.com

PERSONAL INFORMATION

DATE OF BIRTH : September 8, 1992
PLACE OF BIRTH : Pangi, Ladgaron, Claver, Surigao del Norte
AGE : 30
GENDER : Male
NATIONALITY : Filipino
RELIGION : Catholic
CIVIL STATUS : Married
FATHER'S NAME : Tomas N. Paje, Sr.
MOTHER'S NAME : Luz M. Paje

EDUCATIONAL BACKGROUND

COLLEGE : Saint Paul University Surigao
: Corner Rizal and San Nicolas Streets, Surigao City
HIGH SCHOOL : Claver National High School
: Tayaga, Claver, Surigao del Norte
ELEMENTARY : Surigao West Central Elementary School
: Brgy. San Juan, Surigao City

JOHN KENNETH C. VELONTA
Blk. 5 Lot 9 Lopez Habitat, Brgy. Baan Km 3, Butuan City
jkvelonta@gmail.com

PERSONAL INFORMATION

DATE OF BIRTH : May 23, 1997
PLACE OF BIRTH : Butuan City
AGE : 25
GENDER : Male
NATIONALITY : Filipino
RELIGION : Scientology
CIVIL STATUS : Single
FATHER'S NAME : Greg S. Velonta
MOTHER'S NAME : May C. Velonta

EDUCATIONAL BACKGROUND

COLLEGE : Saint Paul University Surigao
: Corner Rizal and San Nicolas Streets, Surigao City
HIGH SCHOOL : Philippine Science High School - Central Visayas Campus
: Brgy. Talaytay, Argao, Cebu
ELEMENTARY : Butuan Central Elementary School - Science and Technology Education Center
: A.D. Curato Street, Butuan City

RICHARD BANOG
Purok 6, Taganito, Claver, Surigao del Norte
richard.banog@gmail.com

PERSONAL INFORMATION

DATE OF BIRTH : September 24,1990
PLACE OF BIRTH : Poblacion, Lingig, Surigao del Sur
AGE : 32
GENDER : Male
NATIONALITY : Filipino
RELIGION : Iglesia ni Cristo
CIVIL STATUS : Single
FATHER'S NAME : N/A
MOTHER'S NAME : Norita Maganyan Banog

EDUCATIONAL BACKGROUND

COLLEGE : Saint Paul University Surigao
: Corner Rizal and San Nicolas Streets, Surigao City
HIGH SCHOOL : Lingig National High School
: Poblacion, Lingig, Surigao del Sur
ELEMENTARY : Hinipaan Elementary School
: Hinipaan, Poblacion, Lingig, Surigao del Sur