

Introduction

1.1 Introduction

Tirius is a small but efficient and intelligent approach to experiment with the Internet of Things. It makes use of the concept of Internet of Things. The IoT devices enable M2M (Machine to Machine) communication methodology. We make use of this feature to integrate several monitoring tools to develop a multi-monitoring system. This software does not identify itself with industrial use, though it is intended for such utility.

1.1.1 Purpose

The primary purpose of this product is to provide effective assistance in monitoring environment, health and safety (EHS) related equipment. It is a follow-on product of an already existing product family. Industries need powerful applications that are reliable and need minimal human assistance. This product is developed with the intention of substituting manual operations of industrial devices with automated ones.

Hence, we should be able to provide industries with a software that can be reliable in terms of task management, and can also be user-friendly to help aid easy monitoring of the system.

1.1.2 Product Scope

This software provides Environment, Health and Safety assistance to industrial equipment. It is in our intention to cater to industries using multiple tools to provide various types of such assistance. Our product incorporates minimal features to show all the aspects that were originally intended. We provide automated assistance as well as GUI for easy user interaction with the software. It can be scaled suitably to meet the requirements of the specific industry. Hence, the tools, nomenclatures and specifications are subject to variation.

1.1.3 Definitions, acronyms and abbreviations

TERM	DEFINITION
EHS	Environment, Health and Safety
CCV	Compliance Condition Variable
MU	Monitoring Unit
IoT	Internet of Things
GUI	Graphical User Interface
IR	Infrared Radiation
M2M	Machine-to-Machine
OSHA	Occupational Safety and Health Administration
GSM	Global System for Mobile Communications
SIM	Subscriber Identification Module
ETSI	European Telecommunications Standards Institute
TCP/IP	Transfer Control Protocol/Internet Protocol

Table 1.1.3

1.2 Literature Survey

The present industrial EHS regulation systems offer fairly efficient problem discovery and monitoring of machine equipment vis-à-vis established regulations. H. Thimm in his paper “Using IoT enabled Multi-Monitoring Data for Next-Generation EHS Compliance Management Systems” introduces a conceptual framework, which offers automation capabilities and smart assistance for compliance management with the use of multi-monitoring data.

Compliance systems in the global market span several fields such as Construction, Healthcare, Pharmaceuticals, Biotechnology, etc.

Today's EHS management systems provide services ranging from eye-protection gear and exit signs to isolation mechanisms for materials with harmful radiations. H. Thimm in another paper "A Continuous Risk Estimation Approach for Corporate Environmental Compliance Management" offers risk assessment, risk aggregation and risk mitigation using an information system based approach.

1.3 Existing System

Velocity EHS (founded 1996), is a private company catering industries with various solutions such as Air Emissions, Compliance Management, Risk Analysis, Waste Compliance, Water Quality, Safety Meetings and other such services. New possibilities are being achieved with the use of IoT in these industries. The United States Department of Labor implements OSHA (Occupational Safety and Health Administration), equipping workers by training, providing directives, enforcement and emphasis programs.

Accenture's article on "Implementing a Digital Strategy for Improved Environmental, Health and Safety Performance" lists multiple improvised automated technologies like:

- Mobile Apps – Can run simultaneously on the fields to collect multiple data streams on the go, benefitting multiple business functions.
- Sensory Equipment – To continuously record various occupational hazard causing factors and to provide authoritative access to designated workers.
- Analytical Models – To gather relevant information, provide operational insight and access possible scalability and improvement.
- Predictive Models – To anticipate the occurrence of an EHS incident with specificity.

1.4 Proposed System

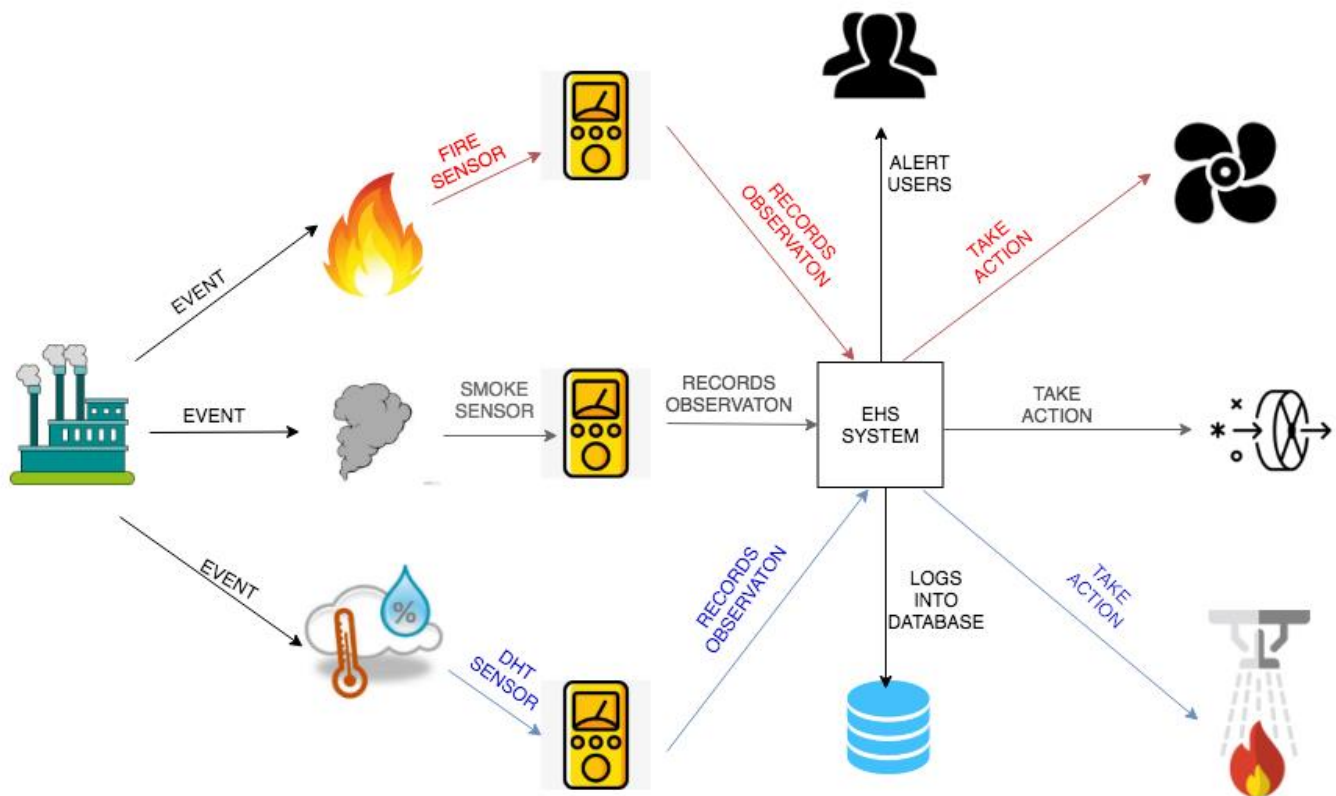


Fig. (1.4) - Proposed System

The EHS system acts as the Monitoring Unit (MU) of the system. It controls the set features of the IoT devices, in the sense that it manages the functions automatically and runs in a loop. The inputs and outputs of all the individual components in this system are monitored in parallel.

First, the administrator starts the operation at the industry by starting off the hardware and the software components manually. The program then runs on the hardware components in the form of the MU, enabling all the sensory equipment present at the site to record data. The sensors start to record the data and print the outputs onto the serial monitor through wired connection. A PyFlask back-end program runs continuously and takes in these inputs and stores them in the database.

Upon detecting any abnormal reading, the reading is immediately checked by the MU of the EHS system, and corrective actions are initiated. In our system, we have incorporated three different setups to demonstrate this. The temperature and humidity sensor records temperature and humidity and readings to the serial monitor. This reading is read by the running python script at the backend and is collected and stored in the database. This happens continuously. Next, the value recorded is checked if it exceeds the normal range. If yes, the MU starts off the corrective action by turning on the DC Fan. The fan speeds up with the increase in temperature and slows down with the decrease in temperature. If the recorded temperature is extremely high, this is a high priority incident and the system sends a message to the user with the help of a GSM module.

Similarly, the fire sensor continuously detects the light around it for any flames/fires by checking for IR light. Unlike the temperature and humidity sensor, the fire detector acknowledges a fire and takes the same kind of operation route, no matter the scale of the fire. Here, the fire detector first detects a flame, upon which the MU turns on the counter-action tool, which is the water sprinkler, with the help of a voltage relay. The water sprinkler can be set to run for a specific amount of time. In the next loop, if the fire detector does not detect flames, the water sprinkler is immediately turned off. A message is then sent to the user with the help of a GSM module, as this is regarded as a high priority incident.

The gas sensor continuously monitors the air in the ambience by checking for alcohol, benzene, CH₄, hexane, LPG and CO. Again, unlike the temperature and humidity sensor, the gas detector acknowledges the presence of smoke/gas in the atmosphere and regards this as a high priority incident. Here, the gas sensor first detects the presence of any of the listed gases. Then, the MU calls for counter-action procedure by turning on the air purifier, which is done with the help of a voltage relay. The air purifier can be set to run for a specific amount of time. In the next loop, if the smoke sensor does not detect any gases, the air purifier is immediately turned off. A message is then sent to the user with the help of a GSM module, as this is regarded as a high priority incident.

An ESP8266 Wi-Fi module is present in the hardware circuitry. This component is needed to send all the recorded values and observations over the Wi-Fi to a specific IP address. The mobile application installed in this case is a third-party application and can be found on both Android and iOS phones. The MU consists of snippets where the Wi-Fi module sends the updates through the channel. The mobile applications run over TCP/IP protocols. And since the MU sends the

messages through Wi-Fi, the user only needs to know the specific IP address and the port address, after which he can gain access and then view the updates in real-time.

Second, the users can access the webpage to gain insights on the temperature and humidity readings as well. This is done by a python script running in the backend, which extracts the readings from the database and displays them on the webpage. A user must first create a login ID for himself, by clicking on the register link. Navigation across the webpage is extremely easy, and all fields are relatable. No specific training is required for this purpose. Users that do not have an ID are denied login

1.5 Statement of the Problem

Industries offer progressive improvement in activities, but also come with several occupational hazards. An EHS Management system continues to aid this development, additionally providing environmental protection and occupational safety, with equal importance to health. Tirius was designed and implemented to handle several features such as monitoring of devices and providing efficient interactive facilities.

1.6 Summary

The design and implementation of Tirius is aimed at linking several IoT devices to provide industrial assistance. In this approach, the system is characterized by “incidents”, that are recorded and then by the “actions”. When multiple incidents occur, the application is able to handle each of the incidents independently. A software module is written for each of these features, and more can be added depending upon added functionality. In Tirius, the incidents are recorded in a database and can be accessed online through a webpage or through a mobile application.

Chapter 2

Software Requirement Specification

2.1 Software Requirements Specification

This section of the document is gathered to provide all the software details of the tools used in this project. Basic functionalities and the environment where the system would be deployed were carefully understood and accounted for before venturing into the implementation. It provides an overview of both the software's requirements and the parameters required for its operation.

2.2 Operating Environment

2.2.1 Hardware Requirements

HARDWARE REQUIREMENTS	
Memory Capacity	1 Gb
Disk Capacity	8 Gb
IP Address - Localhost	127.0.0.1
Port Address - Localhost	5000
IP Address – TCP/IP Application	192.168.4.1
Port Address – TCP/IP Application	80
Microcontroller	Arduino Genuino/Uno

Table 2.2.1

2.2.2 Software Requirements

SOFTWARE REQUIREMENTS	
Operating System	MacOS
Operating System Release	High Sierra v.10.13.4
Python Version	Python v.2.7.14
Database Vendor	MySQL

Database Version	MySQL v.5.7.1
Arduino IDE	Arduino v.1.8.5
UI Framework	PyFlask v.0.12.2

Table 2.2.2

2.3 Functional Requirements

The following table consists of a list of all functional requirements and their corresponding descriptions. For each requirement, priorities are assigned. This is only indicative and the number with the lower value holds greater priority. It is only assigned to help the user understand the order of execution of events. The high priority incidents occur first, followed by the rest.

2.3.1 System Requirements

SYSTEM REQUIREMENTS			
ID	Requirement	Description	Priority
REQ-SYS-01	Identification of incident	Incident should be identified correctly and recorded in a timely manner	3
REQ-SYS-02	Incident Logging	Upon recording the values from the hardware, the data must be logged into the database	3
REQ-SYS-03	Alerting Personnel/Users	Upon the occurrence of an incident, the MU should immediately alert the maintenance personnel and the users	2
REQ-SYS-04	Resolving Tasks	The MU must take counter-action against the incident that has occurred.	1
REQ-SYS-05	Sensor Detection	The used sensors should be able to efficiently detect any violations (in this case, DHT11, MQ-3 and D191)	1
REQ-SYS-06	Error Handling	The sensor modules should have error handling blocks to handle hardware failures	2

Table 2.3.1

2.3.2 User Requirements

USER REQUIREMENTS			
ID	Requirement	Description	Priority
REQ-USR-01	User Registration	The user can register on the website by providing essential details	2
REQ-USR-02	User Login	The user can login in the subsequent sessions, provided he/she has a registered account and can then view the status of the devices	2
REQ-USR-03	Accessing TCP/IP Application	The user can access the TCP/IP application provided he/she knows the IP and the port address	2

Table 2.3.2

2.4 Non Functional Requirements

NON FUNCTIONAL REQUIREMENTS			
ID	Requirement	Description	Priority
REQ-NF-01	Reliability	Should provide realistic results and should be capable of restoring operations upon the occurrence of incidents	1
REQ-NF-02	Performance	Should provide an ideal response time of maximum 10 seconds and 30 seconds in the worst case scenario	1
REQ-NF-03	Security	No unauthorized individuals should be allowed to change the system configurations and new integrations should not affect the existing system	1

Table 2.4

2.5 User Characteristics

This application facilitates support for any normal user who uses the webpage and the mobile application to gain access to the status of the devices. They have no additional privileges. They are however allowed to register themselves on the webpage, and therefore, can login on subsequent sessions (as the user is now added to the database).

2.6 Application of Tirius

Tirius is being made with the intention of providing both industrial automation and user access to the status of all the appliances in the industry. Even though it is simply an addition to an already present family of similar products, Tirius could sustain itself in the market due to the large spectrum of functionalities it is able to provide to the user, especially in terms of the awareness of the user about the happenings in the industry. Tirius does this by incorporating the following functionalities simultaneously:

- It comes with minimal installation procedures, acts as a plug and play device (unless it is subject to any kind of modification).
- Any more external IoT devices can be integrated easily as all the functionalities are coded in the form of modules.
- Status can be monitored both offline and online, i.e., messages are sent offline when an incident occurs.

2.7 Summary

This project was built with limited resources. The application is easily scalable to a much greater extent and is only subject to the type of customization the user/customer requires. Adding new functionalities can bring about several more possibilities and the application could cater industries at any scale.

Chapter 3

High Level Design

3.1 High Level Design

Tirius was built with the help of multiple sensors and their corresponding counter components that must compulsorily work in synchronization to give the results with sufficient efficiency. The details of the design are explained in the later sections.

3.2 Design Considerations

3.2.1 Assumptions and Dependencies

The following are the assumptions that are taken when any customer is operating the system:

- The user must be able to successfully login to the webpage to access its contents.
- The user must be able to successfully gain access to the mobile application by going to the correct IP address.
- The administrator must know how to add/delete/modify the contents of the database or add/delete/modify the modules to vary the functionalities.

The following are the list of dependencies that are considered in order to operate the system:

- Steady power supply should be made available to provide efficient live logging of the data
- Sufficient storage space must be available to store logs and errors to monitor application performance and for further analysis of the records
- A working SIM (Subscriber Identity Module) with strong network signals is necessary to send messages to the user during the occurrence of the incident.
- A third-party mobile application that runs on the TCP/IP client-server protocol where the user enters the IP address and the port number to get live updates of the readings.

3.2.2 Goals and Constraints

The application requires no kind of expensive software/hardware to run. The application is built on all open-source tools and frameworks. The hardware being used is also any UNIX or UNIX based servers. The cost constraint for the database systems are usually not necessary, and even if

present, are only minimal. Large-scale implementations of the same would require extensive hardware components and preferably much more sensitive equipment for better performance and efficiency. The system response time is also a very important constraint, and is presently real-time monitoring. Usage of much better equipment could ensure much faster response time. However, this would depend on the industry being used, and could be used only wherever necessary (such as sensors that monitor high level hazards in the industry).

3.3 System Architecture

System Architecture with two overlapping Architectural patterns, namely Layered and Client and

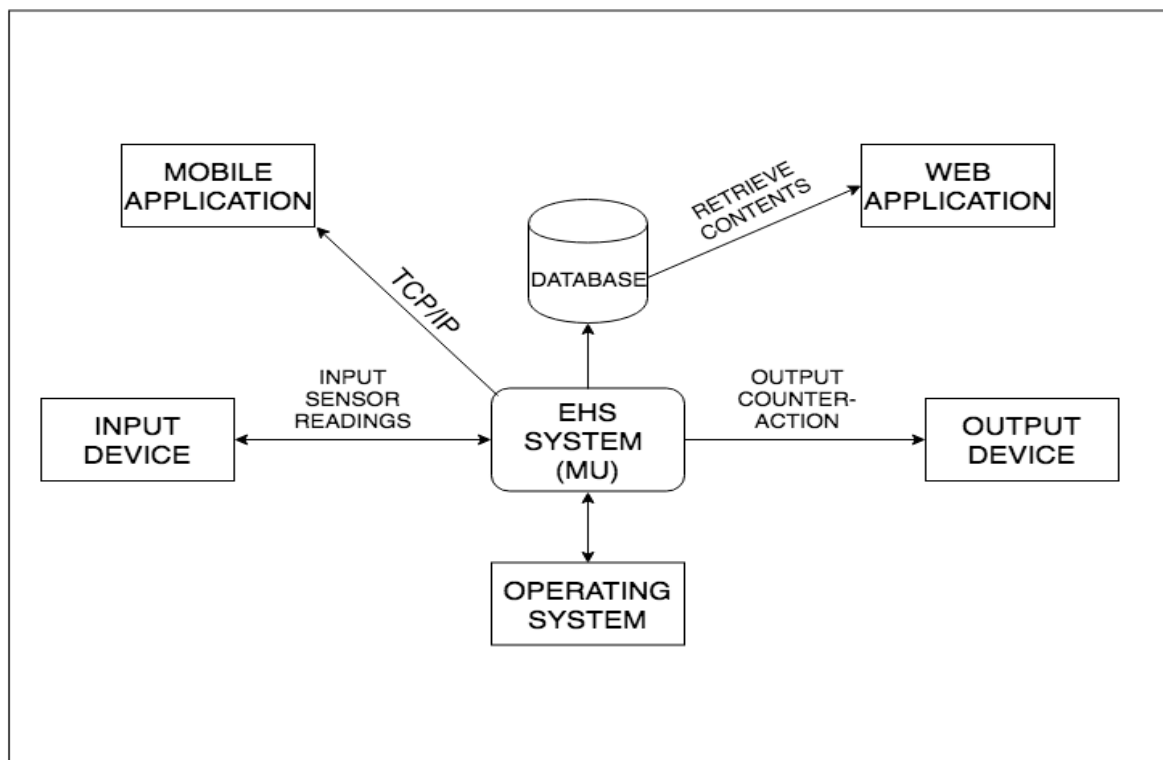


Fig. (3.3) –System Architecture

Server Model:

The System Architecture here is presented by making use of two different system architectures, namely the Layered Architecture and the Client and Server Architecture. It is seen that the architecture here consists of three different layers, where one is the Operating System, the next is the Monitoring Unit and the final layer is the external application.

3.4 Data Flow Diagrams

3.4.1 Data Flow Diagram – Level 0

Level 0 diagram showcases the high level view of the application system. The IoT devices communicate with the MU of the EHS system, and the MU logs the readings into the database.

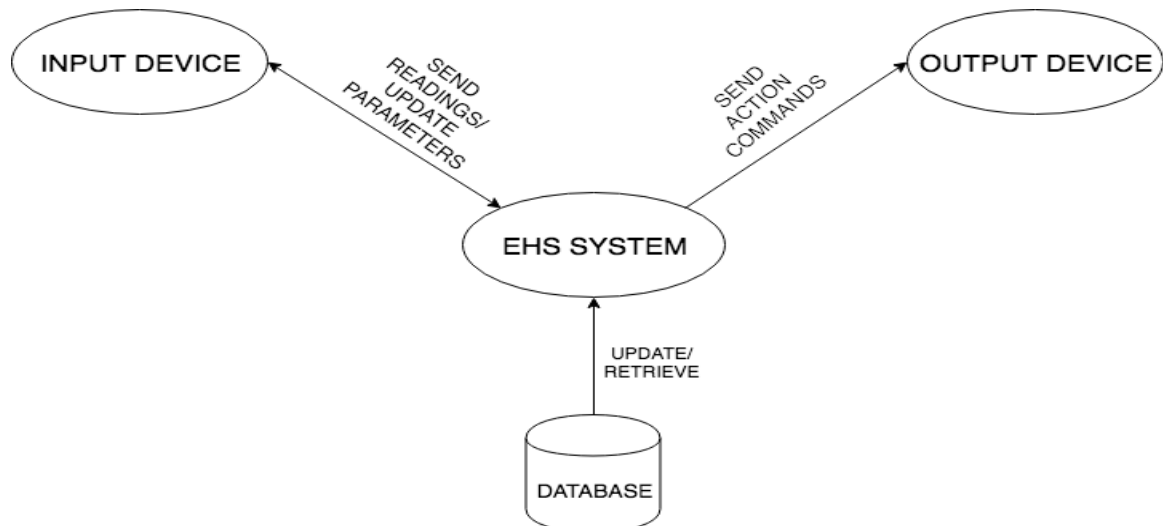


Fig. (3.4.1) – Data Flow Diagram Level 0

3.4.2 Data Flow Diagram – Level 1

Level 1 tries to open up the functions of the application held in the Arduino and displays all the functionalities relevant to it.

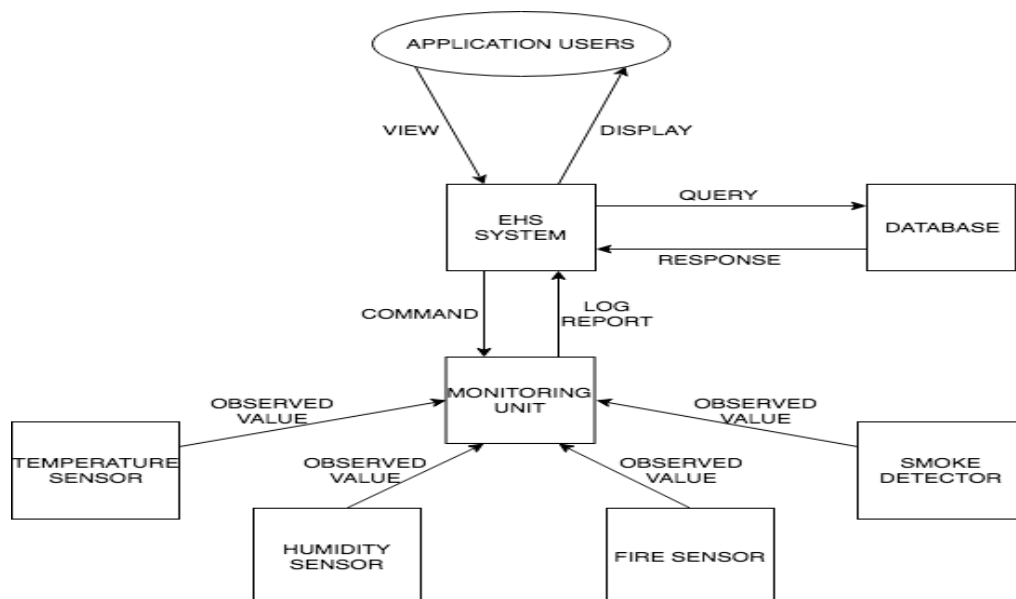


Fig. (3.4.2) – Data Flow Diagram Level 1

3.5 Activity Diagram

The figure shows the user interaction with Tirius.

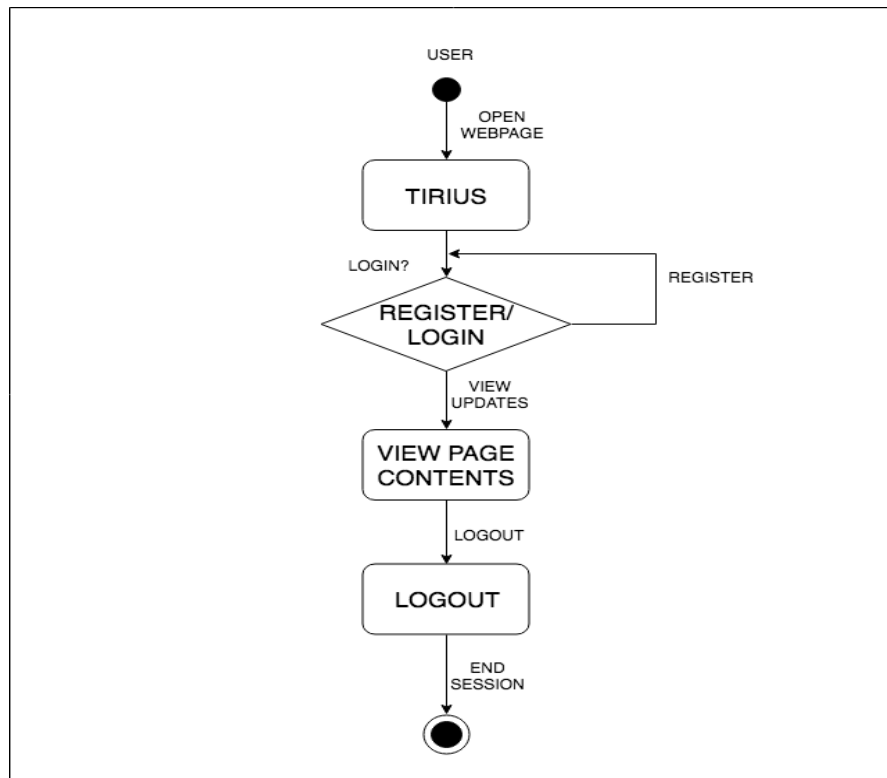


Fig. (3.5) – Activity Diagram

3.6 Functions of Tirius

Tirius has the following functionalities:

- Collect readings from all the IoT sensors connected to the MU of the EHS system.
- Maintain a database to store the readings as they are read from the database.
- Uploading the content of the database onto the webpage.
- Giving access to rightful users to access the webpage contents
- Register new users on the webpage.
- Initiate counter-action when an incident occurs and alert user for high-priority incident.

3.7 Use Case Diagram

The user can access the application through both the mobile application and the web application to simply view the status of the IoT devices that are connected on-site to Tirius.

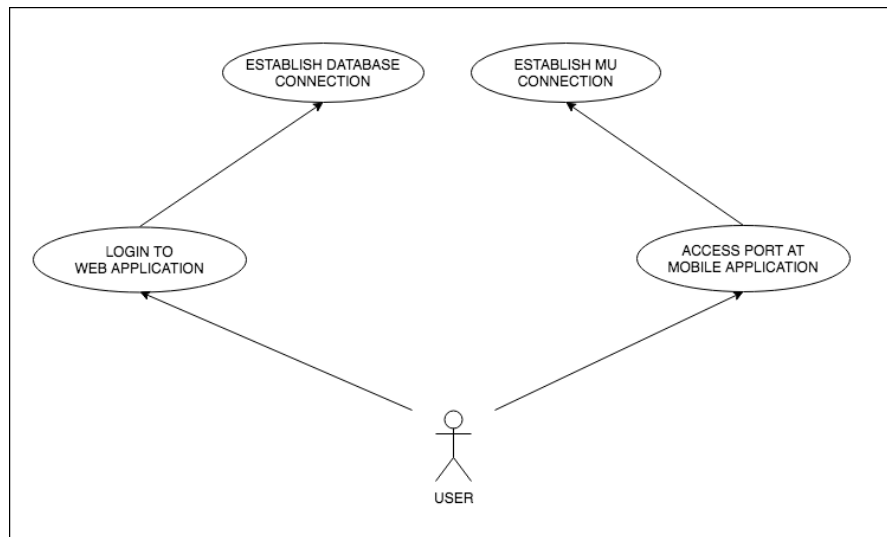


Fig. (3.7) – Use Case Diagram

3.8 Sequence Diagram

The figure shows the data synchronization between the mobile/web application and the EHS monitor, sensors and the database.

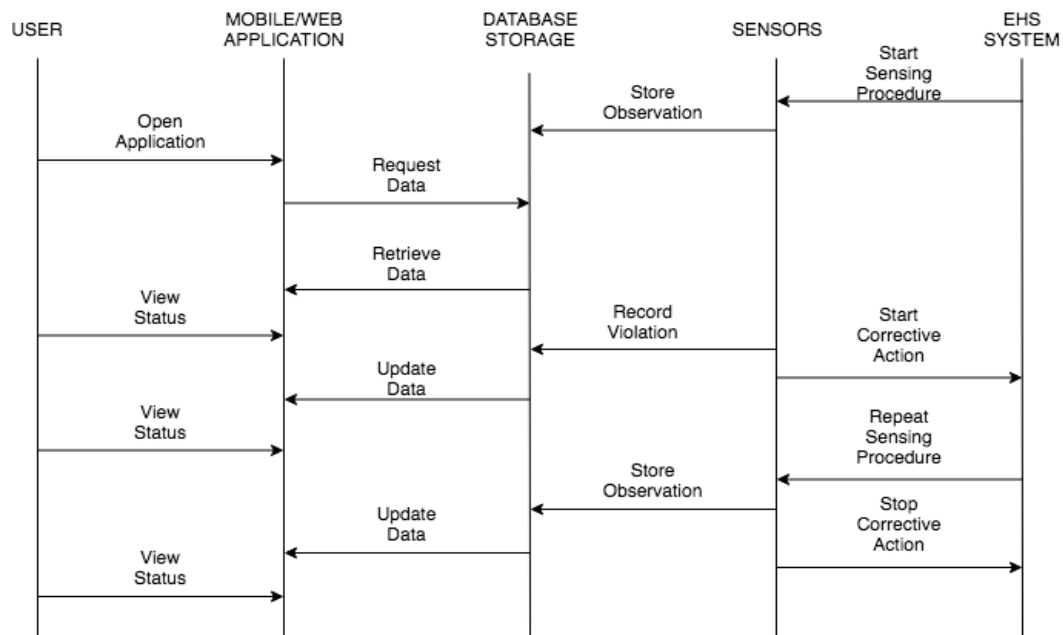


Fig. (3.8) – Sequence Diagram

3.9 Summary

The high level design gives us a brief overview of the device components and the other modules that work together to run the system. This design was the base of making the whole application, from implementation to testing.

Chapter 4

Detailed Design

4.1 Purpose

The idea behind providing the details of the design is to understand the system in the way it was built, such as the sub-systems and the smaller modules. This is done to gain more information about the project and give the developer reading this document the transparency necessary to debug the system. Though it could be irrelevant to the customer/user, one could look through this section to get a better insight on the devices and their working. It also gives an overview of each of the hardware and some software components by providing the inputs and the related outputs, so that the functionalities can be understood thoroughly.

4.2 User Interface Design

4.2.1 Hardware Interfaces

Hardware interfacing is done in a near transparent manner to the user. By this, we mean that the user could visibly look at the interfacing of the hardware components to understand their connectivity. The circuitry is assembled with the help of several jumper wires and by using extra pins, which are shorted to provide equal voltage to all the components, thus enabling them to run simultaneously. The user however does not need to worry about connecting any of the circuitry on his own, unless any administrator is called for to provide any modifications to the already existing functionalities. The interface is designed keeping in mind that the status and data need to be accessed by the mobile phone (smart phone).

4.2.2 Software Graphical User Interfaces

The User Interface has been designed keeping in mind the ease of use of the end user. The intuitive design helps the end user to be familiar with the software without any effort from the very first use. The webpage was designed so as to run smoothly on any browser or any operating environment, thereby making it portable and upgradable for the users at any point in time. It is also flexible, to both the younger and older generations, with minimal information on the screen to avoid any confusion. It is devised to operate with the help of minimal mouse clicks to establish connections, add and delete devices/users in the application.

4.3 Module 1: Temperature and Humidity Sensor (DHT11)

The DHT11 Sensor is a module which can sense both humidity and temperature and record readings simultaneously. The object has to be in close proximity to the sensor, though it should not be in direct contact with the sensor. The input for this module would be nothing that is user assisted. One only needs to place the sensor where he/she wishes the temperature and humidity to be detected. The output would be in numeric, with a precision of two decimal points. Now the output could be in °C, °F or even °K for the temperature reading, a feature we found highly convenient in the ‘TroykaDHT.h’ library.

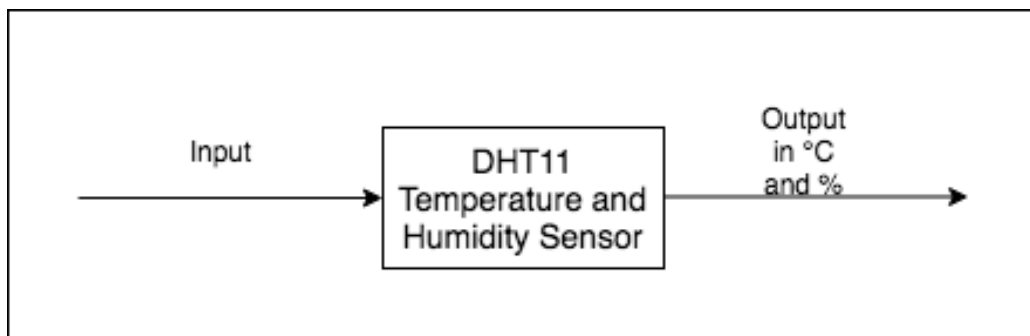


Fig. (4.3) – DHT11 Temperature and Humidity Sensor

4.4 Module 2: Smoke/Gas Sensor (MQ-3)

The MQ-3 smoke detector is a module that is suitable for sensing alcohol, benzene, CH₄, hexane, LPG and CO. This sensor has an impressive sensitivity and fast response time. The input for this module would be any gas or a combination of these gases in the range of the scope of this sensor. The output here would be just a message that gives the status of the incident, in this case telling us if gas was detected or not.

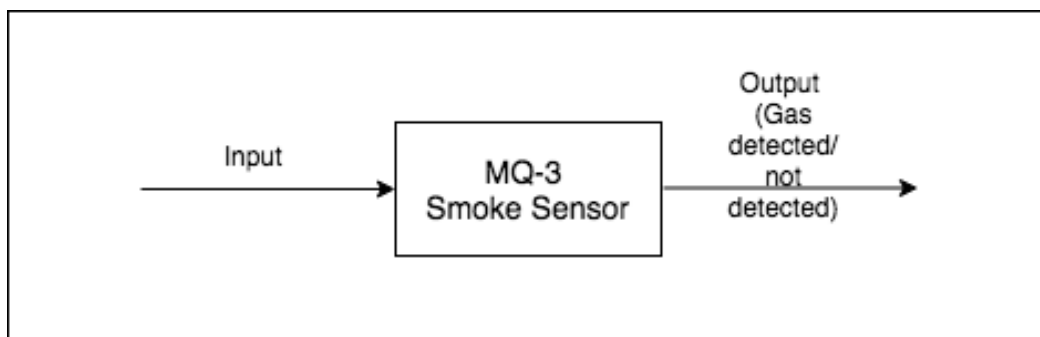


Fig. (4.4) – MQ-3 Smoke Sensor

4.5 Module 3: Fire Detector

The D191 fire detector is a module that consists of an LED bulb that can detect IR radiations. The sensor is relatively sensitive and has fast response time. The input for this module can be sufficed with any small flame/fire that is lit near the bulb. The output here would again, just be an indication that tells us the status of the incident, in this case telling us if a fire was detected or not.

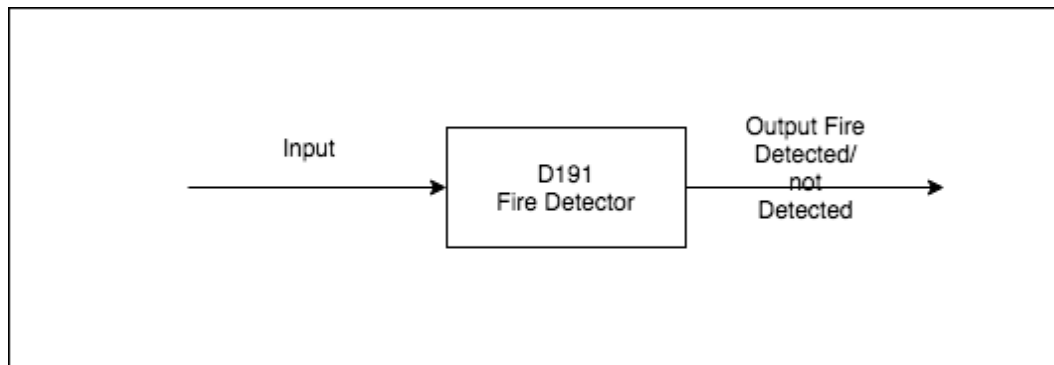


Fig. (4.5) – D191 Fire Detector

4.6 Module 4: DC Fan

The DC Fan is a counter-action device that works in synchronization with the DHT11 sensor. The sensor takes corrective action/measure for a particular assigned threshold range and the fan will speed up based on the temperature reading. If the temperature detected is higher, the fan will speed up and if the temperature detected is lower, the fan will slow down.

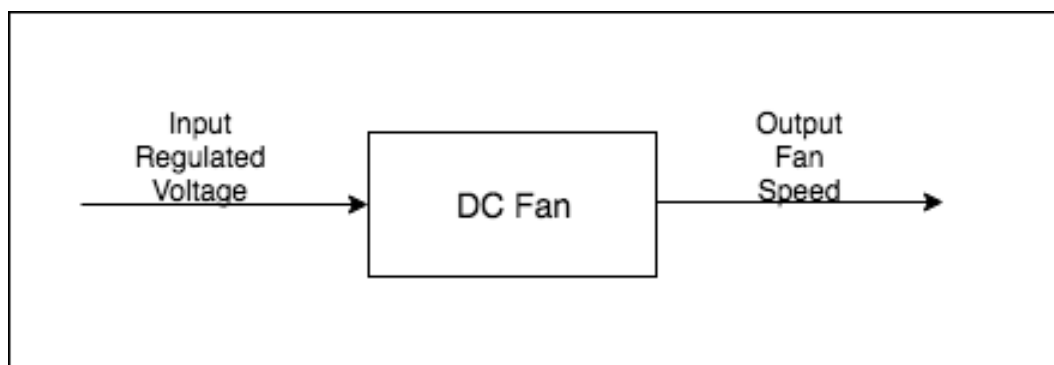


Fig. (4.6) – DC Fan

4.7 Module 5: Air Purifier

The air purifier module is a counter-action device that works in synchronization with the MQ-3 smoke sensor. The sensor registers the incident as ‘gas detected’. The output to this would be enabling the air purifier when gas was detected. The air purifier is connected to the voltage relay that monitors the input voltage to device.

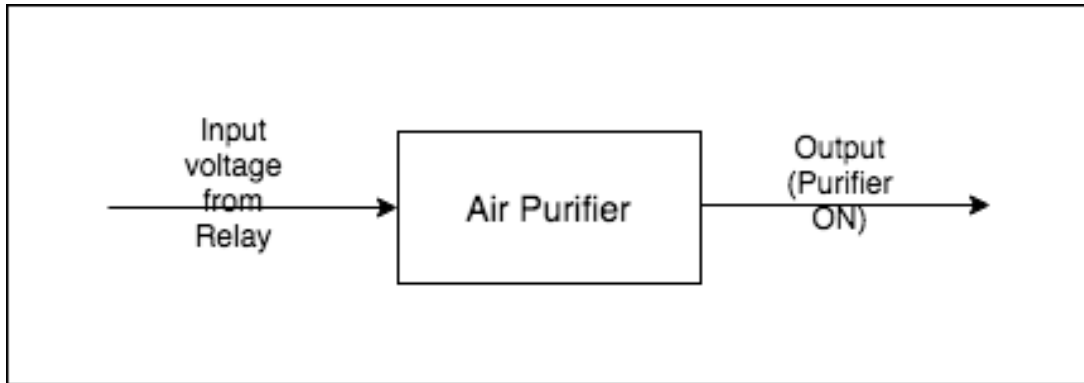


Fig. (4.7) – Air Purifier

4.8 Module 6: Water Sprinkler

The water sprinkler module is a counter-action device that works in synchronization with the D191 fire detector. The sensor registers the incident as ‘fire detected’. The output to this would be enabling the water sprinkler in such instances. The water sprinkler is connected to the voltage relay that monitors the input voltage to the device.

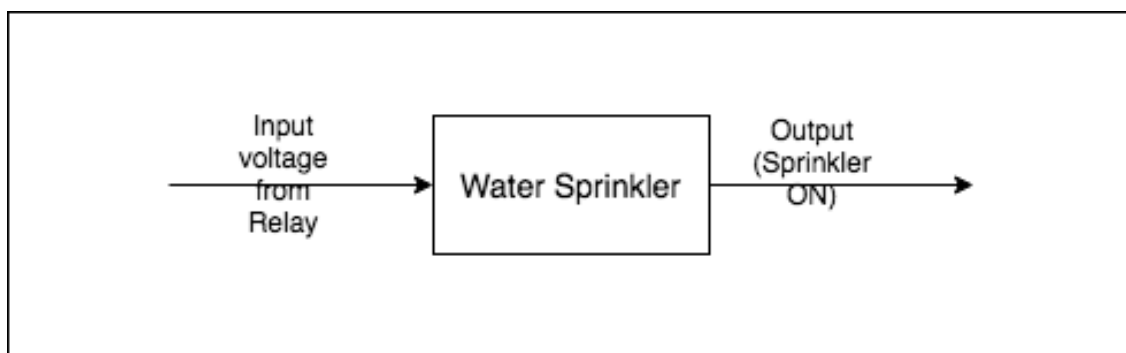


Fig. (4.8) – Water Sprinkler

4.9 Module 7: GSM 800C

The GSM 800C (Global System for Mobile) developed by ETSI (European Telecommunications Standards Institute) is integrated into the system as a part of the error handling mechanism. The GSM module does this by sending messages as an offline part of the application to the user whenever any threats are registered as instances. The GSM module works along with other sensor modules and is therefore, called upon only when necessary.

4.10 Summary

The components used are simple and are assembled with the intention of experimenting the IoT domain. The drivers are coded in Arduino to connect and interact with the hardware components.

Chapter 5

Implementation

5.1 Implementation

The complete project has been implemented using the tools and services available on the web. The backend runs through shell commands to interact with the hardware devices and the server.

5.2 Programming Language Selection

The programming languages were selected after checking for the ones that logically aided the best solution to the project.

5.3 Platform Selection

The developing platform was MacOS and the hardware part of the application is hosted on Arduino Genuino/Uno. However, all the code is portable to every UNIX based environment, except for the fact that they have different installation features for the software required to run these operations.

5.4 Code Conventions

5.4.1 Naming Conventions

The application front end was developed using HTML, CSS and PHP. The naming conventions were strictly followed as per the devices features shown on the screen. For example, the module that was related to the humidity and temperature aspect of the measured readings was named dht11(). Additional functions carried out were also named the same, followed by a dot and then followed by the name of the extended feature.

5.4.2 File Suffix and File Organization

The files were all named as per their functionalities. Each file had a unique function of its own and can be called from any page whenever required. For example, the file which reads the serial monitor of the Arduino Uno's USB port and stores the contents into the database on the backend was named

‘db.py’. This was listed under the /FlaskApp directory, which included both the front end and the backend files.

All the files were then grouped into further sub-directories after checking that they has similar uses and functions.

All functionalities including the application to be run were clubbed into the /FlaskApp directory.

The HTML templates were all clubbed together into the /FlaskApp/templates directory.

5.4.3 Class and Interface Declarations

The hardware module nomenclatures were assigned with respect to the functions they performed. Each of them provided methods of initializing, connecting and modulating the same.

5.4.4 Comments

Most of the code is self-explanatory due to the coding conventions and the file names provided. Some places where the need for explanation was felt necessary has been helped with minimal comments in single lines.

5.5 Graphical User Interface Design

5.5.1 Overview of PHP, Shell, HTML

PHP is a server-side scripting language designed for web development but also used as a general purpose programming language.

The PHP code may be embedded into the HTML code, or it can be used in combination with various other web template systems, web content management systems and web frameworks. PHP code is usually processed by a PHP interpreter implemented as a module in the web server or as a Common Gateway Interface (CGI) executable. The web server combines the results of the interpreted and executed PHP code, which may be any type of data, including images, with the generated web page. PHP code may also be executed with a command-line interface (CLI) and can be used to implement standalone graphical applications.

A UNIX Shell is a command-line interpreter or shell that provides a traditional Unix-like command line user interface. Users direct the operation of the computer by entering commands as text for a command line interpreter to execute, or by creating text scripts of one or more such commands. Users typically interact with a UNIX shell using a terminal emulator, however, direct operation via serial hardware connections, or networking session, are common for server systems. All UNIX shells provide filename wildcarding, piping, here documents, command substitution, variables and control structures for condition-testing and iteration.

Hyper Text Mark-Up Language, commonly abbreviated as HTML, is the standard mark-up language used to create web pages. Along with CSS and JavaScript, HTML is a cornerstone technology used to create web pages, as well as to create user interfaces for mobile and web applications. Web browsers can read HTML files and render them into visible or audible web pages. HTML describes the structure of a website semantically and, before the advent of Cascading Style Sheets (CSS), included cues for the presentation or appearance of the document (web page), making it a mark-up language, rather than a programming language.

5.6 Hardware Interface Design

5.6.1 Overview of PythonFlask, Arduino

Flask is a micro web framework written in Python and based on the Werkzeug toolkit and Jinja2 template engine. It is called a micro framework because it does not require any particular tools or libraries. It has no database abstraction layer, form-validation or any other components where pre-existing third-party libraries provide common functions. However, Flask supports extensions that can add application features as if they were implemented in the Flask itself. It also contains integrated support for unit testing. It also contains its own development servers and debuggers.

The Arduino Integrated Development Environment (IDE) is a cross-platform application that is written in the programming language Java. It originated from the IDE for the languages Processing and Wiring. It includes a code editor that provides simple one-click mechanisms to compile and upload the programs to an Arduino board. The Arduino IDE employs the program avrdude to convert the executable code into a text file in hexadecimal encoding that is loaded into the Arduino board by a loader's program in the board's firmware.

5.7 Database Design

The following diagram shows the database design with the table name, fields and foreign keys.

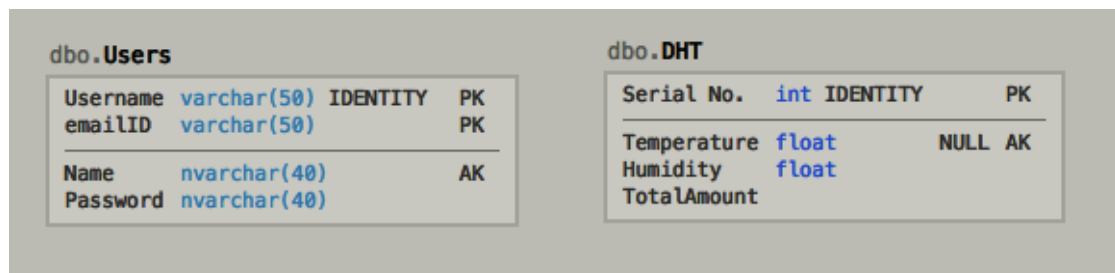


Fig. (5.7) – Database Schema

5.8 Summary

All the programming languages used have been selected carefully keeping in mind its power and contribution to the system development. The file organization and the naming convention aided to the better development and easy understanding of the application in terms of performance and security. The programming languages used also provide fluidity between each other making it possible to have pleasant interfaces without any hassles. They were able to share data and communicate. Proper comments helped to understand the functions to its core and thus enhance the overall efficiency of the system.

Chapter 6

Testing

6.1 Software Testing

Software testing is an investigation conducted to provide stakeholders with information about the quality of the product or service under test. Software testing can also provide an objective, independent view of the software to allow the business to appreciate and understand the risks of software implementation.

Software testing can be stated as the process of validating and verifying that a software program/application/product:

- Meets the requirements that guided its design and development.
- Works as expected.
- Can be implemented with the same characteristics.

6.2 Testing Process

6.2.1 Levels of Testing

Tests are frequently grouped by where they are added in the software development process, or by the level of specificity of the test. The main levels during the development process are unit, integration and system testing that is distinguished by the test target without implying a specific process model. Other test levels are classified by the testing objective.

UNIT TESTING:

Unit testing, also known as component testing refers to tests that verify the functionality of a specific section of code, usually at the function level. In an object-oriented environment, this is usually at the class level, and the minimal unit tests include the constructors and destructors.

The goal of unit testing is to isolate each part of the program and show that the individual parts that are correct. A unit test provides a strict, written contract that the piece of code must satisfy. As a result it affords several benefits.

- Unit tests find problems early in the development cycle.
- Readily available unit tests make it easy for the programmer to check whether a piece of code is still working properly.
- Unit testing provides a sort of living documentation of the system. Developers looking to learn what functionality is provided by a unit and how to use it can look at the unit test to gain a basic understanding of the unit's API.

INTEGRATION TESTING

Integration testing is any type of software testing that seeks to verify the interfaces between components against a software design. Software components may be integrated in an iterative way or even altogether. We used the bottom-up approach for this purpose. Bottom-up Testing is an approach to integrated testing where the lowest level components are tested first, then used to facilitate the higher level components. The process is repeated until the component at the top of the hierarchy is tested.

All the bottom or low-level modules, procedures and functions are integrated and then tested. After the integration testing of the lower level integrated modules, the next level of modules will be formed and can be used for integration testing.

SYSTEM TESTING

The entire system/application was verified and validated that it meets the considered set of objectives. System testing tests a completely integrated system to verify that it meets its requirements. System testing of software or hardware is a testing conducted on a complete, integrated system to evaluate the systems compliance with its specified requirements. System testing falls within the scope of black box testing and as such should require no knowledge of the inner design of the code or logic.

6.3 Testing Environment

A testing environment is a setup of software and hardware on which the testing team is going to perform testing of the newly built product. The test environment of this application consists of an Arduino Genuino/Uno board which runs the hardware aspect of the application and all the IoT devices that are registered.

6.4 Testing of Main Modules

6.4.1 Unit Test of DHT11 Temperature and Humidity Sensor

The application reads the ambient temperature and humidity using the following code:

```
dht.read();

switch(dht.getState()) {

  case DHT_OK:

    dtostrf(dht.getTemperatureC(), 4, 2, temp1);

    Serial.print(temp1);

    Serial.print(dht.getHumidity());

    Serial.print("\r\n");

    if((dht.getTemperatureC())>27)&& (dht.getTemperatureC())<32 )

        analogWrite(analogOutPin, 35);

    else if((dht.getTemperatureC())>32)&& (dht.getTemperatureC())<34 )

        analogWrite(analogOutPin, 40);

    else if((dht.getTemperatureC())>34)&& (dht.getTemperatureC())<36 )

        analogWrite(analogOutPin, 60);
```

```
else if((dht.getTemperatureC())>36)&& (dht.getTemperatureC())<38) )

    analogWrite(analogOutPin, 80);

else if((dht.getTemperatureC())>38)&& (dht.getTemperatureC())<40) )

    analogWrite(analogOutPin, 100);

else if(dht.getTemperatureC())>40)

    analogWrite(analogOutPin, 100);

else if(dht.getTemperatureC())>42)

    analogWrite(analogOutPin, 100);
```

6.4.2 Unit Test of MQ-3 Smoke/Gas Sensor

The application detects if smoke or gases are present in the ambience using the following code:

```
if(digitalRead(gas)==0)

    digitalWrite(relay1,LOW);

else

{

    Serial.println("Gas relay is off");

    digitalWrite(relay1,HIGH);

}
```

6.4.3 Unit Test of D191 IR Fire Detector

The application detects the presence of fire in the ambience using the following code:

```
if(digitalRead(fire)==0)
```

```
        digitalWrite(relay1,LOW);

else

{

Serial.println("Fire relay is off");

        digitalWrite(relay3,HIGH);

}
```

6.4.4 Unit Test of GSM 800C Module for Error Handling

The application made use of the GSM to send timely messages as offline alerts to the user using the following code:

The following code is a snippet of the DHT11 sensor module, and is for instances when the temperature exceeds the set threshold

```
else if(dht.getTemperatureC(>42)

        intGsm("9035994005","ALERT!! TEMPERATURE EXCEEDED!");
```

The following code is a snippet of the MQ-3 sensor module, and is for instances when smoke is detected in the ambience

```
if(digitalRead(gas)==0)

        intGsm("9035994005","ALERT!! SMOKE/GAS DETECTED!");
```

The following code is a snippet of the D191 sensor module, and is for instances when fire is detected in the ambience

```
if(digitalRead(fire)==0)

        intGsm("9035994005","ALERT!! FIRE DETECTED!");
```

6.4.5 Unit Test of ESP8266 Wi-Fi Module for Real-Time Update on Mobile Application

The mobile application runs in parallel and shows all the updates just as they are being updated in the serial monitor. The Wi-Fi module works in synchronization with the other sensor modules.

The following code is a snippet of the Wi-Fi module's role in the DHT11 Temperature and Humidity Sensor module:

```
dht.read();

switch(dht.getState()) {

  case DHT_OK:

    dtostrf(dht.getTemperatureC(), 4, 2, temp1);

    wifiSerial.write(temp1);

  wifiSerial.write(dht.getHumidity());
```

The following code is a snippet of the Wi-Fi module's role in the MQ-3 Smoke/Gas Sensor module:

```
if(digitalRead(gas)==0)

    wifiSerial.write("ALERT!! SMOKE/GAS DETECTED!");
```

The following code is a snippet of the Wi-Fi module's role in the D191 IR Fire Detector module:

```
if(digitalRead(fire)==0)

    wifiSerial.write("ALERT!! FIRE DETECTED!");
```

Chapter 7

Conclusion

7.1 Conclusion

Tirius is a small but efficient and intelligent approach to experiment with the Internet of Things. It makes use of the concept of Internet of Things. The IoT devices enable M2M (Machine to Machine) communication methodology. We make use of this feature to integrate several monitoring tools to develop a multi-monitoring system. This software does not identify itself with industrial use, though it is intended for such utility.

The application developed can be deployed on any scale and if required, can be tuned to be used in any kind of industry. It is easily understandable by all age groups and is not an expensive investment. A common man who would require a home automation device could also use Tirius with a little customization.

The design and implementation of Tirius, an EHS compliance management system that flexibly connects the functionalities of several IoT devices was thus explained in this document. In our approach, the IoT devices wait for an ‘incident’ to occur, upon which a necessary corrective action is taken. The most minimal corrective action that was considered here was the display of an error message. The Arduino handles the operation of all the devices independently, and there is fine co-ordination among the devices. A back end python script is started with the start of Tirius, enabling data to be logged live into the database, which is later retrieved and presented on the webpage.

The operation of Tirius in its hardware form without any user interfaces would not require any kind of prior programming knowledge and can be done with just the click of a switch (no compilation and upload of code into Arduino).

7.2 Limitations of the Project

Tirius presently has the following flaws:

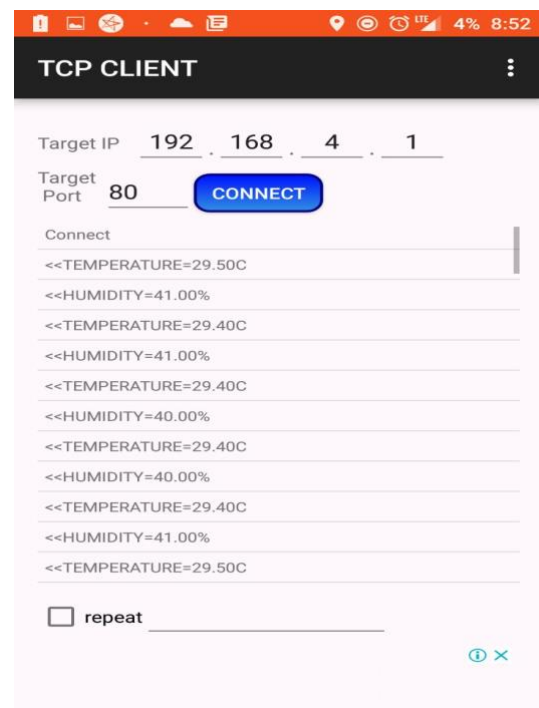
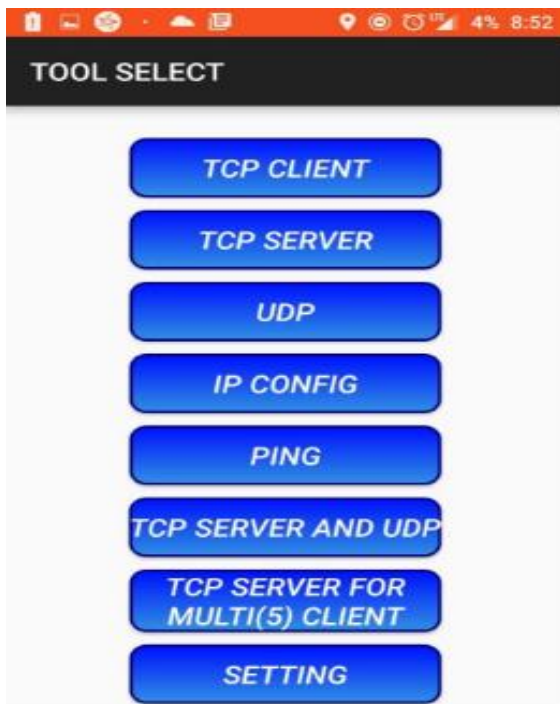
- A maximum of four devices can presently connect to the Wi-Fi module.
- The relays have a latency of approximately 3 seconds.

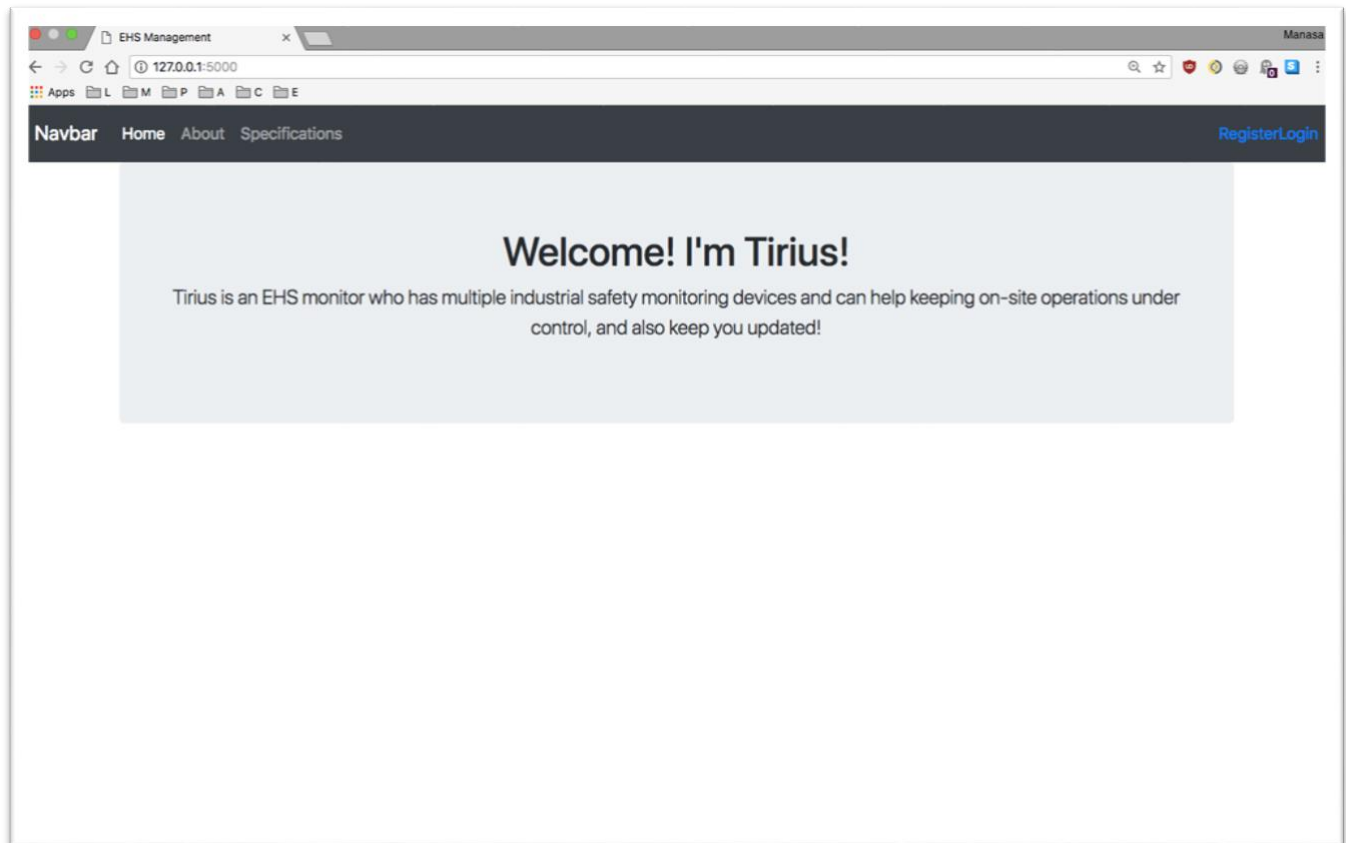
7.3 Future Enhancements

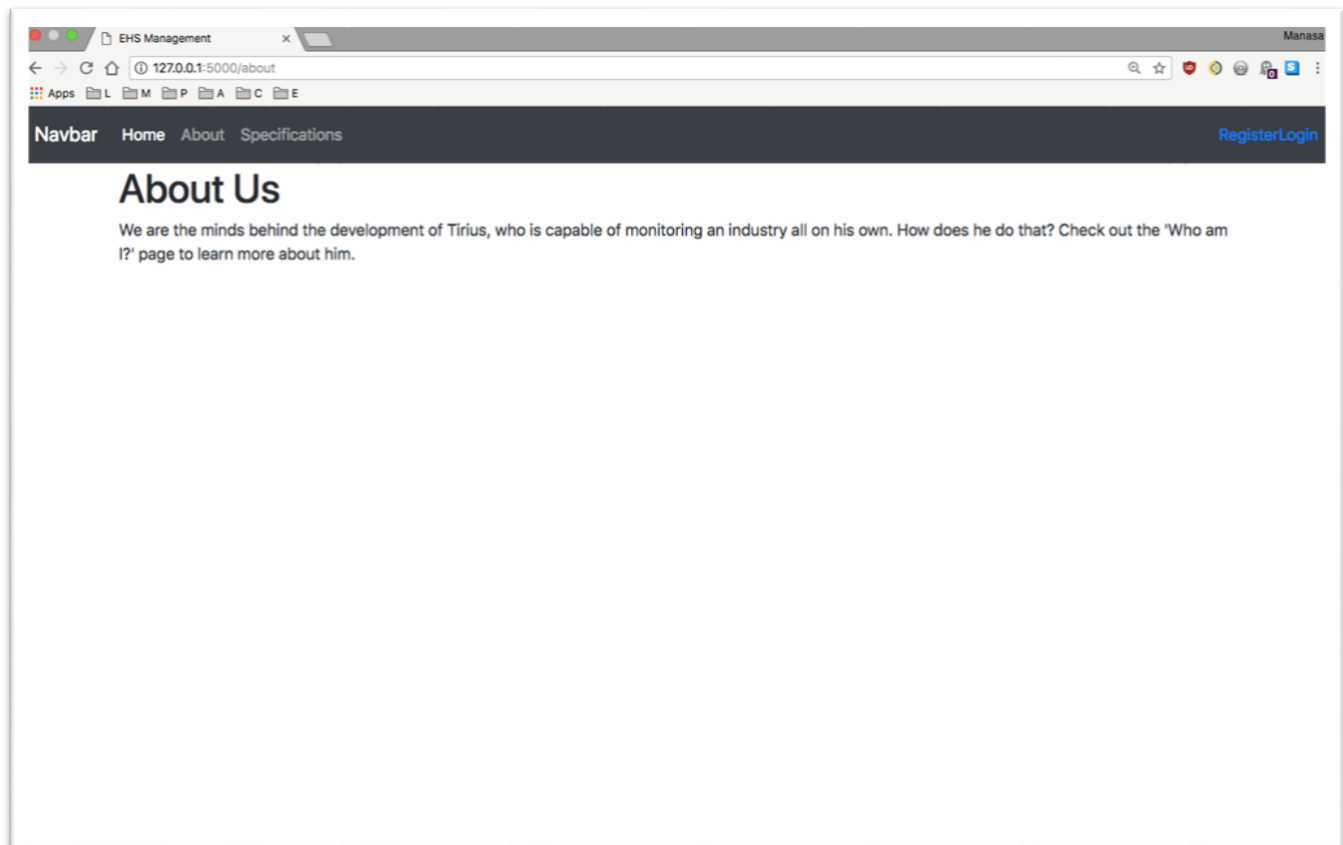
Tirius has several possibilities in terms of growth. The application currently built is powerful and promising. However, Tirius presently handles three different types of hazards of an industry. Better enhancements can be done in the following ways:

- Integrate more devices like sensors such as level indicators, air pressure sensors, flammable liquid sensors, light blocking sensors and so on.
- The latency periods and response times can be improved with the help of better quality sensor modules.
- Analysis of logged reports to help the human resources in making important decisions.

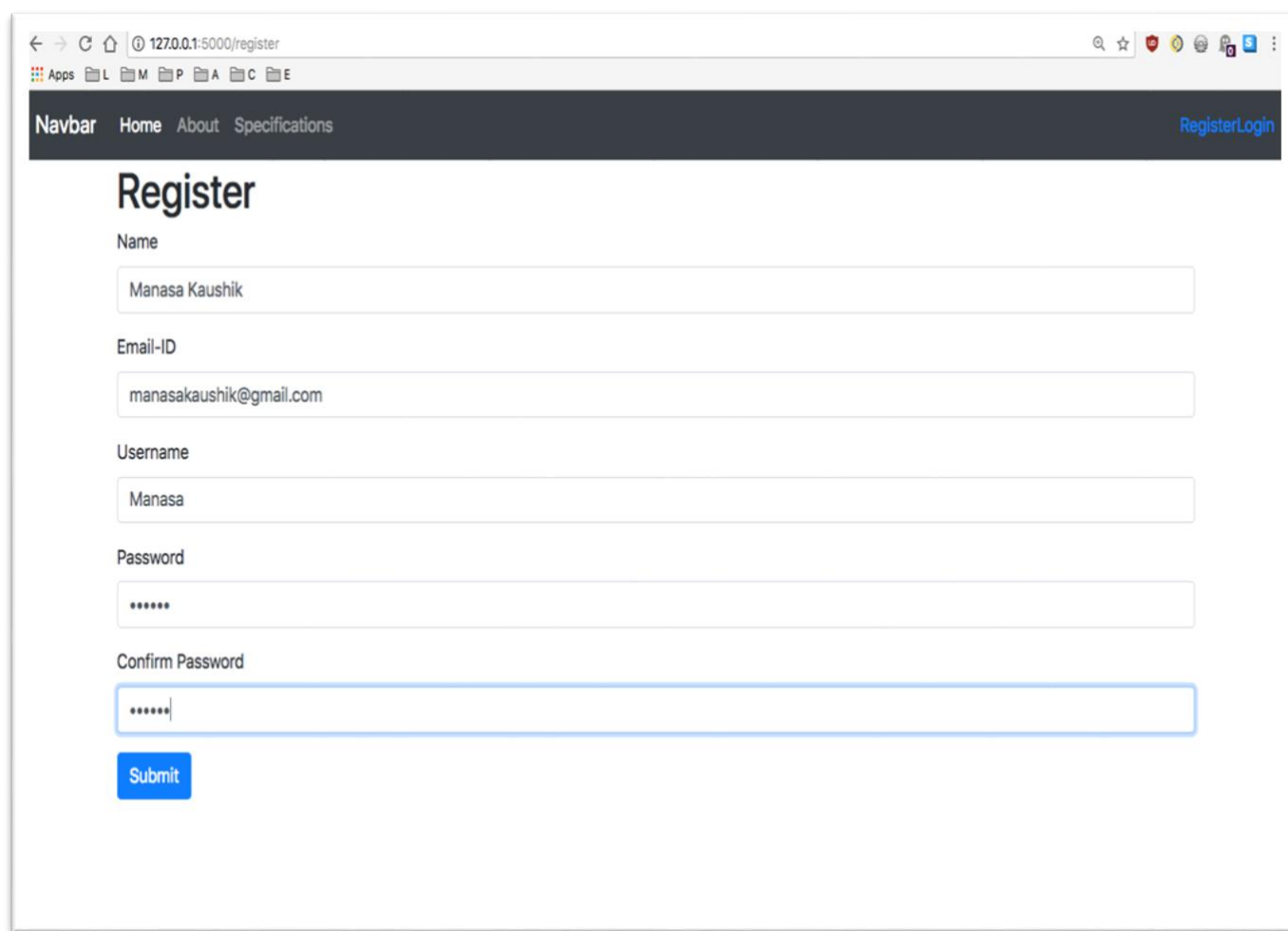
Snapshots







The screenshot shows the 'Register' page of the EHS Management application. The browser window has the title 'EHS Management' and the address bar shows '127.0.0.1:5000/register'. The navigation bar is identical to the previous page. The main content area has the heading 'Register' and a form with the following fields: 'Name', 'Email-ID', 'Username', 'Password', and 'Confirm Password'. Each field is represented by a text input box. A blue 'Submit' button is located at the bottom left of the form.



The screenshot shows a web browser window with the address bar displaying "127.0.0.1:5000/register". The browser's file explorer bar shows a path: "Apps > L > M > P > A > C > E". The page has a dark blue header with a "Navbar" containing links for "Home", "About", and "Specifications". On the right side of the header is a link "RegisterLogin". The main content area is titled "Register" in a large, bold font. Below the title are five input fields, each with a label above it: "Name" (containing "Manasa Kaushik"), "Email-ID" (containing "manasakaushik@gmail.com"), "Username" (containing "Manasa"), "Password" (containing six dots), and "Confirm Password" (containing six dots). A blue "Submit" button is located below the "Confirm Password" field.

Navbar Home About Specifications RegisterLogin

Register

Name

Email-ID

Username

Password

Confirm Password

Submit

A screenshot of a web browser displaying the login page of the Tirius application. The browser's address bar shows the URL `127.0.0.1:5000/login`. The page features a dark navigation bar at the top with links for 'Home', 'About', and 'Specifications', and a 'RegisterLogin' link on the right. A green success message banner reads: 'Your registration was successful! You can now login!'. Below this, the 'Login' section contains two input fields labeled 'Username' and 'Password', and a blue 'Submit' button.

A second screenshot of the same Tirius login page, but with the input fields empty. The browser address bar now shows `127.0.0.1:5000/login`. The navigation bar and success message remain the same. The 'Login' section shows the 'Username' and 'Password' input fields and the 'Submit' button, ready for user input.

Navbar Home About Specifications RegisterLogin

Login

Username

Manasa

Password

Submit

DHT Sensor Readings

Serial No.	Temperature (°C)	Humidity (%)
1	37.80	34.00
2	37.50	34.00
3	37.60	34.00
4	37.60	34.00
5	37.60	34.00
6	37.60	34.00
7	37.50	34.00
8	37.50	34.00
9	37.50	34.00
10	37.50	34.00
11	37.50	34.00
12	37.50	34.00
13	37.50	34.00
14	37.50	34.00
15	37.50	34.00
16	37.60	33.00
17	37.60	33.00
18	37.60	35.00
19	37.60	35.00
20	37.50	34.00
21	37.60	34.00
22	37.50	34.00
23	37.50	34.00
24	37.50	34.00
25	37.40	34.00

References

- [1] H. Thimm, "A Continuous Risk Estimation Approach for Corporate Environmental Compliance Management," Proc. IEEE 15th Int. Conference on Environmental and Electrical Engineering, Rom, Italy, pp. 83-88, 2015.
- [2] A. Rozinat and W. Van der Aalst, "Conformance Checking of Processes Based on Monitoring Real Behavior, " Inf. Syst., vol. 1, pp. 64-95, 2008.
- [3] M. Kharbili, A. de Medeiros, S. Stein and W. Van der Aalst, "Business Process Compliance Checking: Current State and Future Challenges," Proc. MobIS'08, Saarbrücken, Germany, pp. 107-113, 2008.
- [4] C. Ullrich, M. Aust, R. Blach, M. Dietrich, C. Igel, N. Kreggenfeld, D. Kahl, C. Prinz and S. Schwantzer, "Assistance- and KnowledgeServices for Smart Production," Proceedings 15th Int. Conf. Knowledge Technologies and Data-driven Business, 21-23 October 2015.
- [5] C. Giblin, S. Müller und B. Pfitzmann, „From Regulatory Policies to Event Monitoring Rules: Towards Model-Driven Compliance Automation,“ IBM Rüschlikon, Switzerland, 2006.
- [6] <https://en.wikipedia.org/wiki/PHP>
- [7] [https://en.wikipedia.org/wiki/Python_\(programming_language\)](https://en.wikipedia.org/wiki/Python_(programming_language))