X00168331 Lauren Dwyer

Software Engineering | Supervisor: Michael gill

Transport Damage Detection System

fourth year project

# Declaration

*"This report is prepared as a partial fulfilment towards graduation.*

*requirements for Bachelor of Engineering (Honours) in Electronic Engineering,*

*at the Technological University Dublin. I declare that the contents of this.*

*report, and the project to which it refers, are entirely the work of the author.*

*and have not been submitted for a degree in any other institute.”*

Signed: Lauren Dwyer

Date: 23/04/2023

# Abstract

**Transport Damage Detection:** This system consists of microcontroller based embedded systems attached to pallets that carry goods. If a pallet is dropped or roughly handled in a warehouse, a three-axis accelerometer detects this and sets off an alarm and a visual indicator on the pallet. This alarm state is also indicated to a remote operator. The alarm and visual indicator can be reset after the goods are examined.

The following is what I created in the pursuit of executing my chosen topic:

1. Web Browser – prompts the employee to identify themselves through login alert messages when the webpage is reloaded. Once the employee is logged in, if a pallet in the warehouse is found to have been dropped this is indicated on the webpage in an error message that tells the employee what pallet has apparently sustained damage. The employee is then prompted to add a password to confirm they have taken due diligence and repaired the pallet. The employees must identify themselves using both their name and their employee ID numbers.
2. Employees have the option to change the employee details they entered on page reload through the use of the ‘Change Employee’ button.
3. They can also print a copy of the webpages window as is.
4. They can also download a copy of the log as a text file.
5. If the pallet moves suddenly (accelerates), damage is reported to the log on the webpage [part 1] and an LED turns on and a buzzer sounds on the corresponding pallet. Once the employee has located the pallet and repaired it, they must reset the hardware aspect of this system by pressing the button on the pallet.

Contents

[Declaration 1](#_Toc133620270)

[Abstract 1](#_Toc133620271)

[Introduction 6](#_Toc133620272)

[Environment and Ethics 7](#_Toc133620273)

[Background 8](#_Toc133620274)

[Web Server 9](#_Toc133620275)

[What is a web server? 9](#_Toc133620276)

[What is a webpage and what should it consist of? 9](#_Toc133620277)

[jQuery 9](#_Toc133620278)

[Express.js 10](#_Toc133620279)

[Node.js 11](#_Toc133620280)

[Socket.io 11](#_Toc133620281)

[Software Background 12](#_Toc133620282)

[Example of Node.js, Express.js and Socket.io in action 12](#_Toc133620283)

[Calculator 13](#_Toc133620284)

[Top-Level Flowchart 14](#_Toc133620285)

[Design 16](#_Toc133620286)

[Hardware 16](#_Toc133620287)

[Arduino Grove Beginner Kit 16](#_Toc133620288)

[Client-side, Server-side, and MC boardside hardware 17](#_Toc133620289)

[Original Software implementation 18](#_Toc133620290)

[Login Page 18](#_Toc133620291)

[Log Entries Page 20](#_Toc133620292)

[Software Design 25](#_Toc133620293)

[Implementation 26](#_Toc133620294)

[Hardware implementation 26](#_Toc133620295)

[Excessive Acceleration 26](#_Toc133620296)

[Software Implementation 29](#_Toc133620297)

[Getting Employee Details 29](#_Toc133620298)

[Excessive Acceleration 31](#_Toc133620299)

[Employee Notes 32](#_Toc133620300)

[‘Change Pallet’ button. 33](#_Toc133620301)

[‘Change Employee’ button. 34](#_Toc133620302)

[‘Display Shift Log’ button 36](#_Toc133620303)

[‘Download Shift Log (txt file) button. 37](#_Toc133620304)

[‘Download Log as displayed (txt file)’ button. 37](#_Toc133620305)

[‘Print this webpage’ button. 38](#_Toc133620306)

[Top-Level Flowchart 39](#_Toc133620307)

[Command prompt 39](#_Toc133620308)

[Code explanation 40](#_Toc133620309)

[Index 40](#_Toc133620310)

[JSONserverMSP.js 41](#_Toc133620311)

[SensorJSON.ino 42](#_Toc133620312)

[Conclusion 43](#_Toc133620313)

[References 44](#_Toc133620314)

[Appendices 45](#_Toc133620315)

[Index.html 45](#_Toc133620316)

[JSONserverMSP.js 55](#_Toc133620317)

[SensorJSON.ino 58](#_Toc133620318)

[Figure 1: Chatroom Example practice working. 12](#_Toc133620319)

[Figure 2: Calculator code practice example 14](#_Toc133620320)

[Figure 3: Semester 7: Top-Level Flowchart 15](#_Toc133620321)

[Figure 4: Grove Beginner Kit image 16](#_Toc133620322)

[Figure 5: Login Page initially 19](#_Toc133620323)

[Figure 6: Glowing Input fields of the Login Page 19](#_Toc133620324)

[Figure 7: Go to the Log button on the Login Page 20](#_Toc133620325)

[Figure 8:Log Entry Page initially 21](#_Toc133620326)

[Figure 9: Employee Type undefined 22](#_Toc133620327)

[Figure 10:Employee ID number undefined 22](#_Toc133620328)

[Figure 11:Pallet undefined 22](#_Toc133620329)

[Figure 12: Log Entry undefined 23](#_Toc133620330)

[Figure 13: Example Log Entry 23](#_Toc133620331)

[Figure 14:Log Entry printed on the webpage. 24](#_Toc133620332)

[Figure 15:Printing the log/webpage. 24](#_Toc133620333)

[Figure 16: Excessive Acceleration -OLED message 27](#_Toc133620334)

[Figure 17: Button Pressed -OLED message. 28](#_Toc133620335)

[Figure 18: Welcome -OLED message 28](#_Toc133620336)

[Figure 19: Accel value -OLED message 29](#_Toc133620337)

[Figure 20: Webpage at run-time 29](#_Toc133620338)

[Figure 21: Prompt for the employee’s name 30](#_Toc133620339)

[Figure 22: Prompt for the employee's ID number 30](#_Toc133620340)

[Figure 23: Employee details entered, and accelerometer starts. 30](#_Toc133620341)

[Figure 24: excessive acceleration reading. 31](#_Toc133620342)

[Figure 25:excessive acceleration alert message prompt 31](#_Toc133620343)

[Figure 26: multiple instances = multiple password entries 32](#_Toc133620344)

[Figure 27: Using Employee Notes 32](#_Toc133620345)

[Figure 28: Employee note added in the log. 33](#_Toc133620346)

[Figure 29: ‘Change Pallet’ button 33](#_Toc133620347)

[Figure 30: 'Change Pallet' alert message prompt 33](#_Toc133620348)

[Figure 31: Changed palletID in action. 34](#_Toc133620349)

[Figure 32: 'Change Employee' button 34](#_Toc133620350)

[Figure 33: 'Change Employee' name prompt 35](#_Toc133620351)

[Figure 34: 'Change Employee' ID prompt 35](#_Toc133620352)

[Figure 35: 'Change Employee' resulting window. 35](#_Toc133620353)

[Figure 36: Log will multiple employees and their entries 36](#_Toc133620354)

[Figure 37: 'Display Shift Log' window 36](#_Toc133620355)

[Figure 38: 'Download Shift Log (txt file)' - txt file contents 37](#_Toc133620356)

[Figure 39: 'Download Log as displayed (txt file)' -txt file contents. 38](#_Toc133620357)

[Figure 40: 'Print this webpage' button results. 38](#_Toc133620358)

[Figure 41: excessive acceleration on command prompt 39](#_Toc133620359)

[Figure 42: Adding an employee note on the command prompt. 39](#_Toc133620360)

[Figure 43: 'Changing Pallet' on the command prompt 39](#_Toc133620361)

[Figure 44: 'Change Employee' on the command prompt 40](#_Toc133620362)

[Figure 45: Pressing the hardware button -command prompt. 40](#_Toc133620363)

# Introduction

Transporting goods from one place to another can be a challenging task, and it is essential to ensure that the goods arrive at their destination safely. One of the most significant risks that goods face during transport is damage caused by rough handling, such as dropping or impacts. This can result in significant financial losses for the owner of the goods and, in some cases, can cause delays or even cancellations of the entire shipment.

To mitigate these risks, a system was designed that utilizes microcontroller-based embedded systems attached to pallets that carry goods. This system could be useful for logistics and supply chain management. It incorporates a three-axis accelerometer that detects sudden movements or impacts and sets off an alarm (using an on-board buzzer) and visual indicator (on-board LED) on the pallet.

The motivation for choosing the approach of using microcontroller-based embedded systems with accelerometers was to create a cost-effective and reliable system that can be easily attached to pallets carrying goods and easily replicated. Additionally, the system's overall design and implementation were influenced by relevant technology trends, such as the increasing use of Internet of Things (IoT) devices in various industries and the growing demand for real-time monitoring and reporting.

The proposed system offers several advantages over traditional methods of monitoring goods during transport. First, it provides real-time monitoring, allowing for timely intervention in the event of damage. Second, it is cost-effective and easy to implement, requiring minimal modifications to existing pallets. Third, it is easily replicable, enabling large-scale deployment in logistics and supply chain management.

The system's web browser interface offers several features to employees. It prompts employees to identify themselves through login alert messages when the webpage is loaded. Once the employee is logged in, if a pallet in the warehouse is found to have been dropped, an error message is displayed on the webpage, indicating which pallet has sustained damage. The employee is then prompted to add a password to confirm they have taken due diligence and repaired the pallet. The employees must identify themselves using both their name and their employee ID numbers. Employees can also change the employee details they entered on page reload through the use of the ‘Change Employee’ button. Additionally, employees can print a copy of the webpages window as is or download a copy of the log as a text file.

The rest of this report will discuss the system's design and implementation in detail, including the hardware and software components, as well as the design considerations that were considered. Furthermore, it will provide an overview of the testing and validation processes, including the challenges that were encountered and the solutions that were implemented. The benefits of the system in reducing transport damage and minimizing financial losses will also be discussed. The report will also cover the limitations of the system and potential areas for improvement. Finally, the report will conclude with a summary of the key findings and recommendations for future work.

Overall, the proposed system has the potential to revolutionize logistics and supply chain management by providing a cost-effective, real-time, and reliable solution for monitoring goods during transport. With the increasing use of IoT devices in various industries, it is expected that the demand for such solutions will only continue to grow, making the proposed system a timely and valuable contribution to the field.

# Environment and Ethics

Environmental and ethical considerations are essential in any project that aims to improve logistics and supply chain management. In the case of transport damage detection, there are several environmental and ethical factors to consider.

Firstly, the use of embedded systems and IoT devices raises concerns about the disposal of electronic waste (e-waste). It is essential to consider the environmental impact of the materials used in the construction of these systems and ensure that they are properly disposed of at the end of their lifecycle. In this project, we made sure to use materials that are recyclable and environmentally friendly.

Secondly, the use of transportation services contributes to carbon emissions and air pollution. While transport damage detection systems can reduce the need for re-shipping, it is important to consider the environmental impact of the transportation industry as a whole. In this project, we aimed to minimize the need for transportation by improving the quality of shipped goods.

In terms of ethics, it is crucial to consider the privacy and security of the data collected by the system. The collection and storage of personal data must be done in compliance with relevant data protection regulations. It is also necessary to ensure that the data collected is used solely for the purpose of improving logistics and supply chain management and not for other purposes.

Additionally, the system must be designed to minimize the risk of false alarms, which could lead to unnecessary expenses and delays. It is also essential to ensure that the system is not used to monitor employee behaviour, which could infringe on their privacy and rights.

Overall, this project was designed with the aim of improving logistics and supply chain management while considering environmental and ethical considerations. By considering these factors, we can ensure that the system is not only effective but also sustainable and socially responsible.

# Background

This project involves the integration of various technologies to create a transport damage detection system. The system utilizes microcontroller-based embedded systems, a web server, and a web-based interface to provide real-time monitoring and reporting of potential transport damage.

## Web Server

### What is a web server?

A web server is a computer program that serves content to clients on the World Wide Web. It uses the HTTP protocol to communicate with clients and deliver web pages or other content. The server listens for requests from clients and sends back the appropriate response, which can be a web page, a file, data from a database or external hardware. In the transport damage detection system, a web server is used to serve a web page that allows employees to monitor the status of the pallets carrying goods and record when they have fixed any damage to these pallets to a log.

### What is a webpage and what should it consist of?

A webpage is a document or resource that is suitable for the World Wide Web and can be accessed through a web browser.

A webpage typically consists of the following elements:

* HTML (Hypertext Markup Language): The structure of the webpage, which defines the different elements of the page such as headings, paragraphs, lists, images, and links.
* CSS (Cascading Style Sheets): The presentation of the webpage, defines the layout, colours, and font styles used on the page.
* JavaScript: Used to make the webpage interactive and dynamic.
* Media: Images, videos, and audio files that are embedded in the webpage.
* URLs (Uniform Resource Locators): The addresses of webpages and other resources on the web.

Some webpages may also use other technologies like server-side programming languages such as PHP, JavaScript (Node.js), etc.

## jQuery

jQuery is a popular JavaScript library that makes it easier to work with HTML documents, handle events, create animations, and develop AJAX-based applications.

Some of the common tasks that jQuery is used for:

* Selecting and manipulating HTML elements
* Handling events: Makes it easy to add event listeners and handle events such as clicks, mouseovers, and key presses.
* Animations and effects: Provide built-in methods for creating animations and effects, such as fading, sliding, and shaking elements.
* AJAX (Asynchronous JavaScript and XML): Makes it easy to create dynamic, interactive webpages without having to refresh the page.
* DOM Traversal: Provides an easy-to-use API for traversing the DOM, which allows developers to easily access and manipulate elements in the document.

Overall, jQuery is a powerful tool that can help you make your webpages more dynamic and interactive, with less code and effort. It simplifies the process of creating web pages that respond to employee actions.

(JQuery, n.d.)

## Express.js

Express.js is used for building web applications and APIs. It is built on top of Node.js and provides a simple and minimalistic way to create server-side applications using JavaScript.

Some of the common tasks that Express.js is used for including:

* Routing: Allows you to handle different HTTP requests and respond with the appropriate action or content.
* Middleware: These functions are executed before the final request handler, can be used for tasks such as authentication, logging, and error handling.
* Serving static files: Can serve static files such as images, CSS, and JavaScript to the browser, making it easy to create a full-featured web application.
* Request and response handling: Provides a request object that contains information about the incoming request, and a response object that you can use to send a response to the client.

Overall, it is a flexible framework that makes it easy to create web applications and APIs using JavaScript and Node.js.

(Sharma, 2022)

## Node.js

Node.js is an open-source, cross-platform JavaScript runtime environment that executes JavaScript code outside a web browser. It is built on the JavaScript V8 engine, which is the same engine that runs JavaScript in Google Chrome.

Node.js is well-suited for real-time applications that require a high degree of concurrency. This means that it can handle many connections and requests simultaneously, making it a viable choice for real-time applications such as chat applications, online games, etc.

Node.js also provides a built-in package manager called ‘npm’, which makes it easy to install and manage third-party modules and packages.

Node.js can be used for a wide variety of tasks, such as creating web servers, building command-line tools and scripts, and even automating tasks. With the help of frameworks like Express.js, developers can quickly build web applications and APIs.

(Kinsta.com, 2023)

## Socket.io

Socket.IO is a JavaScript library that enables real-time, bidirectional communication between web clients and servers. It provides an abstraction layer over the low-level WebSocket protocol, which allows for real-time communication between a client and a server without the need for HTTP polling. Socket.IO is widely used for building modern web applications that require real-time data updates, such as chat applications, online gaming platforms, and stock tickers.

One of the key features of Socket.IO is its event-driven architecture. It enables developers to define custom events and associated data that can be sent between clients and servers. When a client or server emits an event, any connected clients or servers can receive the event and the associated data. This enables real-time updates and synchronization of data between clients and servers.

Socket.IO also provides a number of other features, such as automatic reconnection, support for multiple communication channels, and the ability to send binary data. It can be used with a variety of server-side platforms, including Node.js, Python, and Ruby.

In the context of this report, Socket.IO is relevant as it was used to enable real-time communication between the microcontroller-based embedded systems attached to pallets carrying goods and the remote operator. This communication allows for timely action to be taken in the event of transport damage, minimizing financial losses, and ensuring the safe delivery of goods.

(Socket.io, 2022)

## Software Background

### Example of Node.js, Express.js and Socket.io in action

In the learning stage of this project, I implemented a chatroom example code from GitHub that gave the below results:

Graphical user interface, text, application, chat or text message

Description automatically generated

Figure 1: Chatroom Example practice working.

To do this I created a folder called “NODE” with the chat-example code stored inside of it. The following files should then be created within the folder:

* index.html
* index.js
* package.json

A copy of the “node\_modules” folder from the GitHub example should also be stored in this folder.

We then go to terminal within the Visual Studios app and apply the change directory command to the terminal:

**cd C:\Employees\laure\OneDrive\Desktop\NODE\chat example**

Then input the following commands to install the needed packages:

* **Npm install express@4**
* **Node index.js**
* **Npm i socket.io**

The chatroom webpage can now be opened at localhst:3000. The result is an editable chat input area that can be opened on multiple browsers and those browsers can communicate with one another.

This helped me to understand the basics of using express.js, socket.io and node.js.

(darrachesquesne, 2021)

### Calculator

I also created a functional calculator as an introduction to coding using HTML, CSS, and JavaScript in conjunction. In the html file I defined what elements I wanted included in the calculator (buttons and an editable text box for a screen). The CSS file was used to add colours and other layout attributes to the calculator. Lastly, the JavaScript file added functionality to each of the buttons.

This is the resulting user interface:

A screenshot of a cell phone

Description automatically generated with medium confidence

Figure 2: Calculator code practice example

### Top-Level Flowchart

This is the flowchart for my webpage in the first semester.

The green coloured sections of this flowchart represent the two webpages that I have created, the login and log entries webpages. The rest of the elements are related to the hardware elements of this project.

A picture containing text, sign

Description automatically generated

Figure 3: Semester 7: Top-Level Flowchart

# Design

## Hardware

The hardware design of the system involved selecting and integrating various components to create a reliable and cost-effective solution. The main components used in the system were:

* a microcontroller,
* an accelerometer sensor,
* a buzzer,
* an LED indicator,
* OLED display.

The system was designed to be attached to pallets carrying goods during transportation so it should be lightweight and easily replicated for mass production. The Grove Beginner Kit for Arduino has all the components necessary for this system.

### Arduino Grove Beginner Kit

The Arduino Grove Beginner Kit is a beginner-friendly platform that can be easily used for prototyping and testing. The Grove Beginner Kit uses a modular system, where sensors and modules can be easily connected to the base shield without the need for soldering. It was programmed to read the data from the accelerometer sensor and control the buzzer, OLED display and LED indicator.

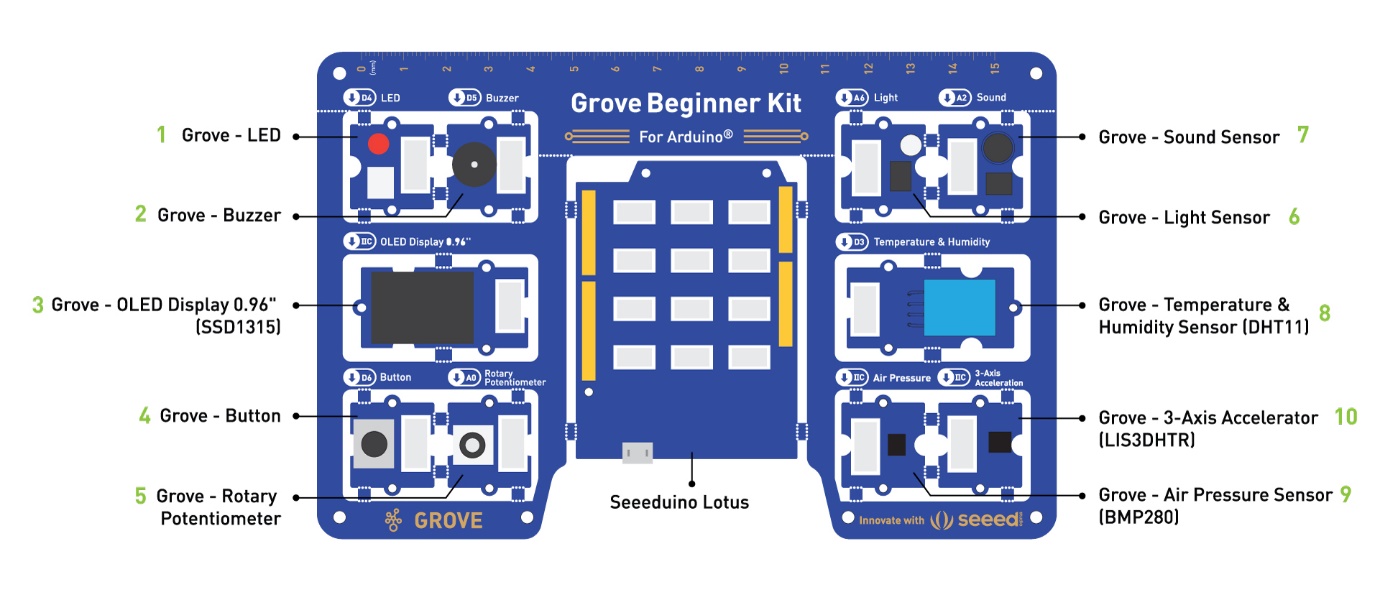


Figure 4: Grove Beginner Kit image

One of the benefits of using the Grove Beginner Kit is that it simplifies the process of connecting sensors and modules to the base shield, which saves time and reduces the risk of damaging the board. The Grove Beginner Kit has a wide range of sensors and modules that are already pre-built and tested, making it easier to integrate them into the project.

Another benefit of using the Grove Beginner Kit is that it has an user-friendly interface that is designed for beginners. The kit comes with a simple, easy-to-use programming environment, which is based on the Arduino IDE, allowing employees to quickly get started with coding without having to learn complex programming languages.

The Grove Beginner Kit is also affordable and widely available, making it accessible and affordable for a wide range of business and industries. Additionally, the Grove Beginner Kit is open source, which means that employees can access the schematics and design files, modify the design to suit their needs, and share their modifications with the community.

(Wiki, 2022)

### Client-side, Server-side, and MC boardside hardware

The hardware components of this program are divided into three main categories: client-side, server-side, and MC boardside.

The **client-side hardware** includes the web browser that employees use to access the program. The program prompts employees to identify themselves through login alert messages when the webpage is reloaded. Once logged in, the program indicates on the webpage in an error message that a pallet in the warehouse has been dropped. The employee is then prompted to add a password to confirm they have taken checked/ repaired the pallet. Employees must identify themselves using their name and employee ID numbers. They can also change their employee details and print a copy of the webpage's window or download a copy of the log as a text file.

The **server-side hardware** includes the hardware used to host the program and store data. This includes a server computer or cloud-based server and storage devices to store the program and log data.

The **MC boardside hardware** includes the sensors and hardware components used to detect and report damage to the pallet. If the pallet moves suddenly, damage is reported to the log on the webpage, and an LED turns on and a buzzer sounds on the corresponding pallet. Once the employee has located the pallet and repaired it, they must reset the hardware aspect of this system by pressing the button on the pallet.

Overall, the hardware components of this program work together to provide a system that helps employees identify and repair damaged pallets in a warehouse. The client-side hardware provides the interface for employees to access the program, while the server-side hardware stores the program and log data. The MC boardside hardware detects and reports damage to the pallets.

## Original Software implementation

The following section is the webpages I created in the first semester, this code didn’t implement the necessary hardware or node.js, socket.io, etc. I created these webpages in a way to plan out the design of my webpage for my second semester and to gain a better understanding of HTML, CSS, and JavaScript.

### Login Page

When the web page is first run a login page (“login.html”) is initialised and prompts the employee to provide their “Employee ID” and “Password”.

(GeeksForGeeks, HTML forms, 2022)

This webpage looks as follows:

Graphical user interface

Description automatically generated

Figure 5: Login Page initially

If you select one of the input fields (for the employee ID number and for the password) the field will pulse with a glowing green effect. ​(Stack-Overflow, 2013)​

Graphical user interface

Description automatically generated

Figure 6: Glowing Input fields of the Login Page

If you hover over the “Go to the Log” button the writing on the button and its border will turn green.

:

Graphical user interface, application

Description automatically generated

Figure 7: Go to the Log button on the Login Page

When the button on this login page is pressed it will open another html file called “log.html”.

### Log Entries Page

Pressing the “Go to the Log” button on the login page will redirect the employee and initialise the following webpage:

Graphical user interface, application

Description automatically generated

Figure 8:Log Entry Page initially

The dropdown menus where the employees can choose what type of employee they are and what pallet they want to reference are implemented using the HTML select element.

When first initialise this web page asks the employee for 4 main pieces of data:

1. The type of employee they are. Are they admin, full-time or part-time staff?
2. Their employee ID number.
3. The pallet that they want to add details about.
4. The details (log) regarding this selected pallet.

All the employee input fields must be completed before selecting the submit button. If one of the fields is not filled an alert message will appear at the top of the window telling the employee which field must still be completed, seen in the following images:

A screenshot of a computer

Description automatically generated

Figure 9: Employee Type undefined

Graphical user interface, application

Description automatically generated

Figure 10:Employee ID number undefined

Graphical user interface, application

Description automatically generated

Figure 11:Pallet undefined

Graphical user interface, application

Description automatically generated

Figure 12: Log Entry undefined

​​ (GeeksForGeeks, HTML window alert() method, 2022)

If all the employee input fields are filled in correctly then the “Submit” button’s function will execute. This button takes the details entered and displays them in the “Log Entries” section of the web page. So, if the employee inputs the following data:

Graphical user interface, text, application

Description automatically generated

Figure 13: Example Log Entry

This will be printed in the Log Entries and the employee input fields will be cleared for the next entry to the log:

Graphical user interface, text, application

Description automatically generated

Figure 14:Log Entry printed on the webpage.

(GeeksForGeeks, HTML DOM textarea object, 2022)

The employee can also use the “Print this page” button to print a copy of the webpages interface, creating a physical copy of the log entries.

A screenshot of a computer

Description automatically generated with medium confidence

Figure 15:Printing the log/webpage.

(GeeksForGeeks, Window print() method, 2022)

## Software Design

The software design of this system includes a web browser interface that prompts the employee to identify themselves through employee name and employee ID alert messages when the webpage is reloaded. The login process requires both the employee's name and employee ID number for identification. Once the employee is logged in, the employee is presented a pallet ID and the current value being detected by the accelerometer on that pallet.

A formal login page was disregarded in this new design to allow for the quicker changing of the employee using the program to allow for multiple users to be coming and going from the same remote user in quick succession where necessary.

If a pallet is found to have been dropped or sustained damage during transport (if its acceleration value is 1.5 or above), an error message is displayed on the webpage, indicating which pallet is affected. Damage reports are automatically added to the log on the webpage, allowing employees to monitor and address any issues in real-time. The employee responsible for the pallet is prompted to enter a password to the webpage to confirm that they have taken appropriate action to repair the pallet.

The software design also includes three buttons that gives the system additional functionality. These buttons are:

1. A 'Change Employee' button that allows employees to modify the employee details they entered during the login process.
2. A ‘Change Pallet’ button that allows employees to view the acceleration of a pallet o their choice.
3. A ‘Print this webpage’ button that allows employees to print a copy of the current webpage they are viewing.
4. A ‘Download log as displayed (txt file)’ button that downloads a copy of the log as a text file for further/future analysis.
5. A ‘Display Shift Log’ button that opens a window that categorises the log by the employee who made each entry and then displays the log.
6. A ‘Download Shift Log’ button that operates the same as the ‘Download log as displayed (txt file)’ button but for the categorised version of the log.

Overall, the software design of this system enables efficient and effective monitoring of the pallets during transport, with real-time reporting of any damage or issues. The web browser interface is employee-friendly, and the system's use of password authentication ensures the integrity of the data collected. The system's log and reporting features allow for easy analysis and management of the transport process, promoting a more efficient and sustainable supply chain management system.

# Implementation

## Hardware implementation

### Excessive Acceleration

When the “ERROR: EXCESSIVE ACCELERATION” message is printed to the webpage, a similar message is printed to the OLED display:

**

Figure 16: Excessive Acceleration -OLED message

Simultaneously, to the above message being printed to the OLED, the hardware LED is also turned on and the buzzer sounds. All this happens because the accelerometer gets a reading of 1.5 or above.

Below the “Excessive Accel” message, an instruction to the employee is also printed which says, “Press button for three seconds”. Following this instruction will ensure the hardware on the pallet is rested to detect any future instances of excessive acceleration.

Once the button has been pressed by the employee the following message will display and the LED will turn off:

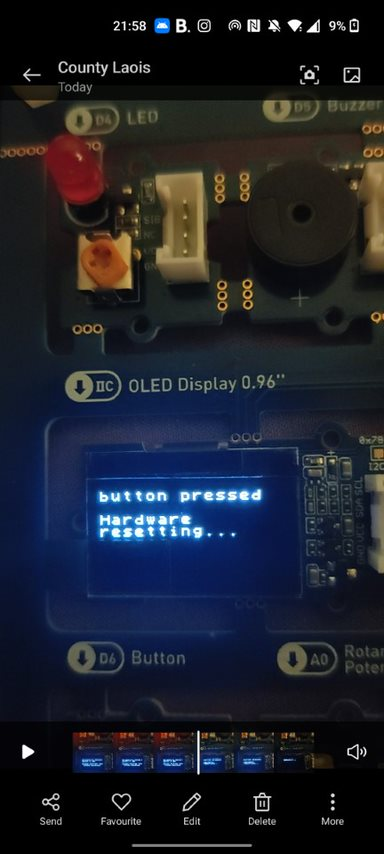
**

Figure 17: Button Pressed -OLED message.

This signals to the employee that they have successfully reset the hardware. The hardware is reinitialised. This can be seen in how it prints the welcome message:



Figure 18: Welcome -OLED message

The OLED will then start displaying accelerometer values again until an acceleration of 1.5 or more is detected by the accelerometer again:



Figure 19: Accel value -OLED message

Here is a video showing the way the OLED display changes once the user presses the on-board button:



## Software Implementation

### Getting Employee Details

When the page is run for the first time the employee (employee) will be prompted to provide their name and employee ID through alert messages that appear consecutively:

Graphical user interface, website

Description automatically generated

Figure 20: Webpage at run-time

Graphical user interface, website

Description automatically generated

Figure 21: Prompt for the employee’s name

Graphical user interface, website

Description automatically generated

Figure 22: Prompt for the employee's ID number

Once the employee’s name and ID number is entered, they will be printed to the window and the connected clients will be incremented. A random pallet and: its acceleration will be shown to the employee:

Graphical user interface, website

Description automatically generated

Figure 23: Employee details entered, and accelerometer starts.

### Excessive Acceleration

If the accelerometer gets a reading of 1.5 or above another alert message will prompt that tells the employee that there is an error on the current pallet.

It gives instructions to the employee to reset the hardware manually by pressing the Grove’s button and input a password “password” in the given input field to add entries to the log once the employee has checked/repaired the pallet in question:

Graphical user interface, website

Description automatically generated

Figure 24: excessive acceleration reading.

Graphical user interface, website

Description automatically generated

Figure 25:excessive acceleration alert message prompt

If there is more than one instances of excessive acceleration, then this program will asks for the password for each instance but the hardware button need only to be pressed once:

Graphical user interface, website

Description automatically generated

Figure 26: multiple instances = multiple password entries

### Employee Notes

If the employee wants to add additional notes to the log, they can do so using the “Employee Notes” text area and “submit note to the log” button. This is useful to record if part of the pallet needs to be replaced or if any goods were damaged, etc. Below is an example of a note that can be added:

Graphical user interface, website

Description automatically generated

Figure 27: Using Employee Notes

Graphical user interface, text, application

Description automatically generated

Figure 28: Employee note added in the log.

### ‘Change Pallet’ button.

If the employee wishes to look at another pallet, they simply have to press the “Change Pallet” button and an alert message box will appear that will prompt them to enter the pallet ID of the pallet they want to view/ add to the log of:

Graphical user interface, website

Description automatically generated

Figure 29: ‘Change Pallet’ button

Graphical user interface, website

Description automatically generated

Figure 30: 'Change Pallet' alert message prompt

Once the pallet ID has been changed the program runs the same as before but for this new pallet:

Graphical user interface, website

Description automatically generated

Figure 31: Changed palletID in action.

### ‘Change Employee’ button.

The “Change Employee” button will run the authenticate function just like it does at the initial running of the program. This means the employee will once again be prompted for their name and ID number and a random Pallet ID will be displayed in the window:

Graphical user interface, application, website

Description automatically generated

Figure 32: 'Change Employee' button

Graphical user interface, application, website

Description automatically generated

Figure 33: 'Change Employee' name prompt

Graphical user interface, website

Description automatically generated

Figure 34: 'Change Employee' ID prompt

Graphical user interface, text, application

Description automatically generated

Figure 35: 'Change Employee' resulting window.

Graphical user interface, text, application

Description automatically generated

Figure 36: Log will multiple employees and their entries

### ‘Display Shift Log’ button

The “Display Shift Log” button opens a window that displays the log after categorising it by employee. This is beneficial to shift managers/supervisors so they can keep track of their employee’s productivity:

Graphical user interface, application

Description automatically generated

Figure 37: 'Display Shift Log' window

### ‘Download Shift Log (txt file) button.

The manager/supervisor can also download a text file version of this categorised log using the “Download Shift Log (txt file)” button on the webpage:

Text

Description automatically generated

Figure 38: 'Download Shift Log (txt file)' - txt file contents

### ‘Download Log as displayed (txt file)’ button.

There is also the option to download a copy of the log that is not categorised. This is done to be able to see any of the notes added by the employees. This is done through the use of the “Download Log as displayed (txt file)” button. This is the resulting txt file:

Text

Description automatically generated

Figure 39: 'Download Log as displayed (txt file)' -txt file contents.

### ‘Print this webpage’ button.

There is also a “Print this webpage” button that can be used by an employee to report back to their supervisors that an accelerometer on one of the pallets is giving a strange reading or if they are getting an unknown error on the webpage:

Graphical user interface, text

Description automatically generated

Figure 40: 'Print this webpage' button results.

### Top-Level Flowchart

## Command prompt

When the accelerometer detects an acceleration value of 1.5 or more, a message prints on the webpage, on the OLED and on the command prompt:

Text

Description automatically generated

Figure 41: excessive acceleration on command prompt

When the “Submit” button for the “Employee Note” section of the webpage is pressed the following prints on the command prompt:

Text

Description automatically generated

Figure 42: Adding an employee note on the command prompt.

When the ‘Change Pallet’ button has been pressed and a new palletID has been entered in the prompt that then appears, the following is printed to the command prompt:

Text

Description automatically generated

Figure 43: 'Changing Pallet' on the command prompt

The ‘Change Employee’ button works similarly and prints the following:

Text

Description automatically generated

Figure 44: 'Change Employee' on the command prompt

When the hardware button is pressed to reset the LED and the OLED, the following is printed to the command prompt:

Text

Description automatically generated

Figure 45: Pressing the hardware button -command prompt.

# Code explanation

## Index

This is a script written in JavaScript. It uses socket.io to establish a connection with the server, receives sensor data from the server and displays it on the web page.

The script allows users to log in with a username and employee ID, and then displays the user's name and ID on the web page. The script also generates a random pallet ID and displays it on the web page. The pallet ID can be changed by clicking the "Change Pallet ID" button.

The script listens for sensor readings from the server and displays them on the web page. If the acceleration value is greater than or equal to 1.5, the function handleExcessiveAcceleration() is called which makes automatic entries to the log.

The script also allows users to log their own notes, which are displayed on the web page along with the log entries from the handleExcessiveAcceleration() function. The notes are logged with the user's name, ID, and the pallet ID. Users can download the log or print it. The log can be downloaded in two formats: a text file with all entries or a text file with entries categorized by employee.

Finally, the script also provides a button to display the log entries in a new window categorised by which employee entered them into the log and another button that allows this same altered log to be downloaded as a text file.

## JSONserverMSP.js

This is a JavaScript file that defines a Node.js application that listens for client connections and sends messages to a serial port, as well as receives data from the serial port and sends it back to the clients.

The application uses the Express framework for creating a web server and Socket.io for real-time communication between the server and clients. It also uses the SerialPort and ReadlineParser libraries to communicate with a serial port.

The application listens for client connections and sends a response containing an HTML file called "index.html" when a client connects to the root path ("/"). When a client disconnects, the application updates a counter for the number of active clients and broadcasts this information to all connected clients.

The application defines several event listeners for Socket.io events, each of which sends a message to the serial port with a specific command. These events include:

* EmpChange: Sends a message to the serial port with the command "EmpChange". This is used to notify the system that an employee's details have changed, and a new pallet is ready to be scanned.
* PalletChange: Sends a message to the serial port with the command "PalletChange". This is used to notify the system that the pallet ID has changed due to an employee's action.
* LogNote: Sends a message to the serial port with the command "LogNote". This is used to notify the system that an employee has added a note to the log.
* LogPrint: Sends a message to the serial port with the command "LogPrint". This is used to notify the system to print the log.
* LogDownloaded: Sends a message to the serial port with the command "LogDownloaded". This is used to notify the system that the log has been downloaded as a text file.
* The application also defines an event listener for data received from the serial port. This listener reads the data and checks if it starts with the string "Sensors=". If it does, it extracts the JSON object that follows and emits it to all connected clients using the "sensorReadings" event. If the data is equal to the string "Button Pressed", it emits a message containing this information to all connected clients using the "buttonReading" event.

Finally, the application defines event listeners for the "close" and "error" events on the serial port. These listeners log messages when the port is closed or encounters an error.

The application is configured to listen on port 4000 using the HTTP server provided by the Express framework. When the server is started, it logs a message to the console indicating that it is listening on port 4000.

## SensorJSON.ino

This is an Arduino sketch (.ino file) that responds with a JSON string when it receives a "readSensor\n" message on the Serial port. It is based on the Serial Event example and it reads accelerometer values from the LIS3DHTR sensor.

When the accelerometer values exceed a certain threshold, it triggers an LED and buzzer and sends a JSON string with the accelerometer value to the Serial port. It also displays a message on an OLED display.

If a button connected to the Arduino is pressed while the LED is triggered, it turns off the LED and displays a message on the OLED display. If the LED is triggered, it also displays a message on the OLED display.

The sketch includes some conditional compilation instructions to handle different hardware configurations. It uses the U8x8lib library to interface with the OLED display and the LIS3DHTR library to interface with the accelerometer. The software or hardware I2C implementation is selected based on the #ifdef SOFTWAREWIRE directive. The LED\_BUILTIN constant is defined differently based on the hardware platform.

The ‘setup()’ function initializes the OLED display, the serial port, the input string buffer, the random seed generator for temperature, and the accelerometer. It also sets up the LED pin, the buzzer pin, and the button pin.

The ‘loop()’ function reads the accelerometer values, calculates the acceleration value, and sends a JSON string with the accelerometer value to the Serial port if the threshold is exceeded. It also turns on the LED and buzzer and displays a message on the OLED display. If the button is pressed, it turns off the LED and displays a message on the OLED display. If the LED is triggered, it displays a message on the OLED display. The ‘loop()’ function also handles the Serial communication by reading incoming characters and assembling them into a string until a newline character is received.

# Conclusion

In conclusion, the development of a web-based system using jQuery, Node.js, and socket.io has led to the creation of a comprehensive platform for warehouse management. The integration of software and hardware components provides a more efficient way of managing a warehouse and improving the overall quality of service. This system provides an effective solution for tracking product damages and allows for timely repairs, ultimately resulting in an improved customer experience.

The completion of a project such as this required me to gain a good understanding of web development technologies such as jQuery, Node.js, and socket.io. These technologies are essential for building a responsive and interactive web-based system that can handle user inputs, display real-time data, and integrate hardware components.

A project like this also required careful planning and organization, starting from defining the requirements, identifying the necessary components and features, and designing the system architecture. This planning process helps to ensure that the end product is well-structured, efficient, and meets the intended objectives.

This project helped to improve my problem-solving skills. The development of a warehouse management system requires the identification and resolution of complex issues, such as real-time data synchronization, user authentication, and hardware integration. Overcoming these challenges requires creative thinking and problem-solving skills.

Lastly, completing a project like this helped to improve my project management skills, including planning, scheduling, and tracking progress. Effective project management is crucial for ensuring that the project is completed on time, within budget, and to the required quality standards.

In conclusion, the completion of a project such as a warehouse management system can provide valuable learning opportunities, including the development of technical skills, problem-solving skills, and project management skills. These skills are transferable and can be applied to future projects, leading to more efficient and effective software development.

# References

darrachesquesne. (2021, June 28). *Basic Chat example with socket.io*. Retrieved from gitbub.com: https://github.com/socketio/chat-example

GeeksForGeeks. (2022). *HTML DOM textarea object*. Retrieved from geeksforgeeks.org: https://www.w3schools.com/jsref/dom\_obj\_textarea.asp

GeeksForGeeks. (2022, June 22). *HTML forms*. Retrieved from GeeksForGeeks.org: https://www.geeksforgeeks.org/html-forms/

GeeksForGeeks. (2022, December 8). *HTML window alert() method*. Retrieved from geeksforgeeks.org: https://www.geeksforgeeks.org/html-window-alert-method/

GeeksForGeeks. (2022). *Window print() method*. Retrieved from geeksforgeeks.org: https://www.w3schools.com/jsref/met\_win\_print.asp

JQuery. (n.d.). *What is jQuery?* Retrieved from jquery.com: https://jquery.com/

Kinsta.com. (2023, January 17). *What is Node.js and why you should use it*. Retrieved from kinsta.com: https://kinsta.com/knowledgebase/what-is-node-js/#:~:text=Download-,Node.,suitable%20for%20real%2Dtime%20applications.

Sharma, A. (2022). *What is Express JS in Node JS*. Retrieved from simplilearn.com: https://www.simplilearn.com/tutorials/nodejs-tutorial/what-is-express-js

Socket.io. (2022, December 12). *Introduction to scoket.io*. Retrieved from socket.io: https://socket.io/docs/v3/

Wiki, S. (2022). *Grove Beginner Kit for Arduino*. Retrieved from Seeed Wiki: https://www.google.com/url?sa=i&url=https%3A%2F%2Fwiki.seeedstudio.com%2FGrove-Beginner-Kit-For-Arduino%2F&psig=AOvVaw26uNW-OZMC088xsU5NecPI&ust=1682805986246000&source=images&cd=vfe&ved=0CAQQjB1qFwoTCJCirfrKzf4CFQAAAAAdAAAAABAE

# Appendices

## Index.html

<!DOCTYPE html>

<!-- M. Gill 2020 -->

<html>

<head>

<title>Sensor JSON MSP</title>

<script src="/socket.io/socket.io.js"></script>

<script src="http://code.jquery.com/jquery-1.11.1.js"></script>

</head><h1>Transport Damage Detection</h1>

<style>

body {

overflow-y: scroll; //enable scrolling through the webpage

padding: 0px;

background-color:#82CAFF ; //blue

}

h1 {

font-size: 28px;

margin-top: 0;

text-align: center

}

p {

margin: 0 0 10px;

font-size: 18px;

}

#log {

margin-top: 20px;

padding: 10px;

background-color: #f2f2f2;

border-radius: 5px;

}

#log ul {

list-style-type: none;

margin: 0;

padding: 0;

}

#log li {

font-size: 16px;

margin-bottom: 5px;

}

#log li.error {

color: #f00;

}

#client\_count,

#Accelerometer,

#PalletNumber,

#EmpName,

#EmpID {

font-weight: bold;

color: #333;

font-size: 18px;z

display: inline-block;

}

#submitNote{

background-color: #00BFFF;

border: none;

border-radius: 25px;

color: white;

padding: 10px 20px;

text-align: center;

font-size: 14px;

margin: 4px 2px;

cursor: pointer;

}

#CEbut, #CPNbut ,#Printbut, #downloadLog, #catLog, #displayLogButton, #downloadEmployeeLog

{

background-color: #00BFFF;

border: none;

border-radius: 25px;

color: white;

padding: 10px 20px;

text-align: center;

text-decoration: none;

display: inline-block;

font-size: 14px;

margin: 4px 2px;

cursor: pointer;

}

</style>

<script>

$(document).ready(function () {

var empName, empID, palletID, logByEmployee = {};;

function authenticate() {

empName = prompt("Please enter your username:");

empID = prompt("Please enter your employee ID number:");

// check username and password

if (empName !== null && empID !== null) {

$('#EmpName').html(empName);

$('#EmpID').html(empID);

// initialize logByEmployee object with empty arrays for each employee

if (!(empName in logByEmployee)) {

logByEmployee[empName] = [];

}

// set initial pallet ID

if (palletID === null) {

palletID = "error";

}

//generate a random pallet at login

palletID=Math.floor(Math.random() \* 100);

//start listening for sensor readings

socket.on("sensorReadings", function (msg) {

// extract values

console.log(msg); // log the entire JSON data received

console.log(msg.accel); // log the accelerometer value extracted

$("#Accelerometer").html(msg.accel);

$("#PalletNumber").html(palletID);

// Check if the acceleration value is classed as excessive

if (msg.accel>=1.5) {

handleExcessiveAcceleration();

}

});

} else {

// authentication failed

alert("Invalid username or password.");

}

}

// connect using socket.io

var socket = io();

//Change Employee details button

$("#CEbut").click(function () {

socket.emit('EmpChange',{}); //command prompt message

// prompt user to re-enter username and password

authenticate();

});

//Change Pallet ID button

$("#CPNbut").click(function() {

socket.emit('PalletChange',{}); //command prompt message

palletID = prompt("Please enter the new pallet ID:");

});

//Submit note to the log button

$("#submitNote").click(function() {

socket.emit('LogNote',{}); //command prompt message

var notes = $("#notes").val();

if (notes !== "") {

var logItem = "\n<li>"+new Date().toLocaleString()+" NOTE: "+ notes+" ( PALLET: "+palletID+" ) - " + empName + ", " + empID + "</li>\n\n";

$("#log").append(logItem);

$("#notes").val("");

}

});

//Print this webpage button

$("#Printbut").click(function(){

socket.emit('LogPrint',{});

window.print();

});

//Download log as displayed to a txt file button

$("#downloadLog").click(function () {

socket.emit('LogDownloaded',{}); //command prompt message

// create a new text file with the log entries

var logText = $("#log").text();

var file = new Blob([logText], {type: "text/plain"});

var a = document.createElement("a");

var url = URL.createObjectURL(file);

a.href = url;

a.download = "log.txt";

document.body.appendChild(a);

a.click();

setTimeout(function () {

document.body.removeChild(a);

window.URL.revokeObjectURL(url);

}, 0);

});

//Download a log that is categorised by employee

$("#downloadEmployeeLog").click(function () {

socket.emit('LogDownloaded',{});

// create a new text file with the log entries categorized by employee

var logText = "";

for (var employee in logByEmployee) {

logText += employee+"\n";

logText += "--------------------------------------------------\n";

for (var i=0; i<logByEmployee[employee].length; i++) {

logText += logByEmployee[employee][i]+"\n";

}

logText += "\n";

}

var file = new Blob([logText], {type: 'text/plain'});

var a = document.createElement("a");

var url = URL.createObjectURL(file);

a.href = url;

a.download = "log.txt";

document.body.appendChild(a);

a.click();

setTimeout(function() {

document.body.removeChild(a);

window.URL.revokeObjectURL(url);

},0);

});

//Display the categroised log in a new window

$("#displayLogButton").click(function () {

// create new window with log entries categorized by employee

var logWindow = window.open();

logWindow.document.write("<h1>Log Entries Categorized by Employee</h1>");

logWindow.document.write("<style>body {background-color: #e6f2ff;}</style>"); // set background color here

for (var employee in logByEmployee) {

logWindow.document.write("<h2>"+employee+", "+empID+"</h2>");

logWindow.document.write("<ul>");

for (var i=0; i<logByEmployee[employee].length; i++) {

logWindow.document.write(logByEmployee[employee][i]);

}

logWindow.document.write("</ul>");

}

});

// authenticate on page load

authenticate();

//when accelerometer gets an acceleration reading of 1.5 or more

function handleExcessiveAcceleration() {

// Prompt the user for a password and store it

handleExcessiveAcceleration.password = prompt("ERROR on pallet "+palletID+"\n\nTo log and stop errror: \n(1) Input required password below [password]\n(2) Hold down the on-board button for 3 seconds");

// Check if the password is correct (for example, "password")

if (handleExcessiveAcceleration.password == "password") {

var logStartTime = new Date().toLocaleString();

$("#log").append("<li>" + logStartTime+ " ERROR: EXCESSIVE ACCELERATION DETECTED ON PALLET "+palletID+"</li>");

// Resume logging

var logItem = "\n<li>" + logStartTime+ " pallet: "+palletID+" - excessive accel handled by "+empName+", "+empID+"\n</li>";

$("#log").append(logItem);

//add log entry to the appropriate employees log

logByEmployee[empName].push(logItem);

delay(2000);

} else {

// Incorrect password

alert("Incorrect password. Logging will not resume.");

var logItem = "\n<li>" + new Date().toLocaleString() + " -logging stopped due to incorrect password entry</li>";

$("#log").append(logItem);

}

}

//empty the text area of its instructional message contents when the user clicks on it

const notesTextArea = document.getElementById('notes');

notesTextArea.addEventListener('click', () => {

notesTextArea.value = '';

});

//update the client count - how many tabs of the program are open at any one time

socket.on("nClientUpdate", function (msg) {

$("#client\_count").html(msg.clients);

});

});

</script>

</head>

<body>

<div class="employee-notes" style="float: right; margin-left: 160px; background-color: #e6f2ff; border-radius: 10px; padding: 10px;">

<p>Employee Notes:</p>

<textarea id="notes" rows="4" cols="60" onclick="this.value=''">enter additional comments about this pallet here....</textarea><br>

<button type="button" id="submitNote">Submit note to the log</button><br><br><br>

</div>

<div class="employee-info">

<p><span id="client\_count">0</span> connected clients</p>

<p>Emp: <span id="EmpName">error</span></p>

<p>EmpID: <span id="EmpID">error</span></p><br>

</div>

<div class="pallet-info">

<p>Pallet Number: <span id="PalletNumber">NULL</span> </p>

<p>Accelerometer: <span id="Accelerometer">error</span></p>

</div>

<button type="button" id="CPNbut">Change Pallet Number</button>

<button type="button" id="CEbut">Change Employee</button>

<div id="log">

<ul>

<li>Log:</li>

</ul>

</div>

<button type="button" id="Printbut">Print this webpage</button>

<button type="button" id="downloadLog">Download Log as displayed (txt file)</button>

<button type="button" id="displayLogButton">Display Shift Log</button>

<button id="downloadEmployeeLog">Download Shift Log (txt file)</button>

</body>

</html>

## JSONserverMSP.js

// Routes readSensors message to serial port and sends back reply

const express =require("express");

const app = express();

const http = require('http').Server(app);

const io = require('socket.io')(http);

const { SerialPort } = require('serialport');

const { ReadlineParser } = require('@serialport/parser-readline');

const port = new SerialPort({ path: 'COM4', baudRate: 9600 }, function (err) {

if (err) {

return console.log('Error on port open: ', err.message);

}

console.log('Serial Port Open');

});

const parser = port.pipe(new ReadlineParser({ delimiter: '\r\n' }));

let activeClients = 0; //number of connected clients

app.get('/', function(req, res){

res.sendFile(\_\_dirname + '/index.html');

});

io.on('connection', function(socket){

//update the client count on the webpage

socket.on('disconnect', function(){

activeClients -=1;

io.emit('nClientUpdate', {clients:activeClients});

});

//print to the command prompt when actions are taken on the webpage

socket.on('EmpChange',function(){

port.write('EmpChange\n', function(err) {

if (err) {

return console.log('Error on port write: ', err.message);

}

console.log('Employee details changed, Showing new pallet');

});

});

socket.on('PalletChange',function(){

port.write('PalletChange\n', function(err) {

if (err) {

return console.log('Error on port write: ', err.message);

}

console.log('Pallet ID changed by employee');

});

});

socket.on('LogNote',function(){

port.write('LogNote\n', function(err) {

if (err) {

return console.log('Error on port write: ', err.message);

}

console.log('Employee added a note to the log');

});

});

socket.on('LogPrint',function(){

port.write('LogPrint\n', function(err) {

if (err) {

return console.log('Error on port write: ', err.message);

}

console.log('Log was printed');

});

});

socket.on('LogDownloaded',function(){

port.write('LogDownloaded\n', function(err) {

if (err) {

return console.log('Error on port write: ', err.message);

}

console.log('Log downloaded as a text file');

});

});

activeClients +=1;

io.emit('nClientUpdate', {clients:activeClients});

});

//Data handler for data received from Serialport via parser

parser.on('data', function(data) {

let dataStr = data.toString();

console.log(dataStr);

if(dataStr.startsWith('Sensors=')){ //accelerometer readings

//extract JSON object after 'Sensors='

let data = JSON.parse(dataStr.substring('Sensors='.length));

io.emit('sensorReadings', data);

}

if (dataStr === 'Button Pressed') { //hardware button

io.emit('buttonReading', {message: 'Button Pressed!'});

}

});

port.on('close', function() {

console.log('Port closed');

})

port.on('error', function(err) {

console.log('Error on port: ', err.message);

})

http.listen(4000, function(){

console.log('listening on \*:4000');

});

## SensorJSON.ino

/\*

SensorJSON modified by MGill 2023

Responds with a JSON string when "readSensor\n" message is

received on the Serial port

It can be tested using Serial Monitor 9600 baud and newline

Based on Serial Event example

When new serial data arrives, this sketch adds it to a String.

When a newline is received, the loop prints the string and

clears it.

\*/

//check if it is a launchpad

#if defined \_\_msp430

#define LED\_BUILTIN GREEN\_LED

#endif

#include <Arduino.h>

//Gravity Acceleration

#include <U8x8lib.h>

U8X8\_SSD1306\_128X64\_NONAME\_HW\_I2C u8x8(/\* reset=\*/ U8X8\_PIN\_NONE);

#include "LIS3DHTR.h"

#ifdef SOFTWAREWIRE

#include <SoftwareWire.h>

SoftwareWire myWire(3, 2);

LIS3DHTR<SoftwareWire> LIS; //Software I2C

#define WIRE myWire

#else

#include <Wire.h>

LIS3DHTR<TwoWire> LIS; //Hardware I2C

#define WIRE Wire

#endif

String inputString = ""; // a string to hold incoming data

boolean stringComplete = false; // whether the string is complete

float accel; // To generate fake temperature value

//led

int ledPin = 4;

bool ledTriggered = false;

//buzzerx

int buzzerPin = 5; // buzzer connected to pin 5

//initialize the button

int buttonPin=6;

String serialData = "";

void setup() {

// initialize the OLED display

u8x8.begin();

u8x8.setFlipMode(1);

// set the initial display message

u8x8.setFont(u8x8\_font\_chroma48medium8\_r);

// initialize serial:

Serial.begin(9600);

// reserve 200 bytes for the inputString:

inputString.reserve(200);

randomSeed(0); //generate same sequence each time for temp

pinMode(LED\_BUILTIN, OUTPUT);

digitalWrite(LED\_BUILTIN, LOW); //led is off

// initialize the accelerometer

LIS.begin(WIRE, 0x19);

delay(100);

LIS.setOutputDataRate(LIS3DHTR\_DATARATE\_50HZ);

// initialize the LED and buzzer

pinMode(ledPin, OUTPUT);

pinMode(buzzerPin, OUTPUT);

//initialize the button

pinMode(buttonPin, INPUT\_PULLUP);

}

void ‘loop()’ {

if (!LIS) {

Serial.println("LIS3DHTR didn't connect.");

while (1);

return;

}

// Read accelerometer values

float x = LIS.getAccelerationX();

float y = LIS.getAccelerationY();

float z = LIS.getAccelerationZ();

float accel=sqrt(x\*x+y\*y+z\*z);

Serial.print("Sensors={\"accel\":");Serial.print(accel);Serial.println("}");

if (accel>= 1.5 && !ledTriggered) { //makes it so the led stays on until button is pressed

Serial.println("ERROR: excessive acceleration detected");

tone(buzzerPin,2000,100);

ledTriggered = true;

digitalWrite(ledPin, LOW);

}

// if the LED has been triggered and the button is pressed...

if (ledTriggered && digitalRead(buttonPin) == HIGH) {

ledTriggered = false;

digitalWrite(ledPin, LOW); //turn off the LED

// Clear OLED display

u8x8.clearDisplay();

//print message to the LED

u8x8.setCursor(0,0);

u8x8.print("button pressed");

Serial.println("Button pressed");

u8x8.setCursor(0,2);

u8x8.print("Hardware");

u8x8.setCursor(0,3);

u8x8.print("resetting...");

delay(2000);

// Print welcome message

u8x8.clearDisplay();

u8x8.setCursor(0,0);

u8x8.print("Welcome!");

delay(500);

}

// if the LED has been triggered, display a different message on the OLED display

if (ledTriggered) {

u8x8.clearLine(0); // clear the first line of the OLED display

//print message to the OLED

u8x8.setCursor(0, 0);

u8x8.print("Excessive Accel!");

digitalWrite(ledPin, HIGH); //on

u8x8.setCursor(0, 1);

u8x8.print("Accel:");

u8x8.print(accel);

Serial.print("Sensors={\"accel\":");Serial.print(accel);Serial.println("}");

delay(2000);

u8x8.setCursor(0, 3);

u8x8.print("Press button for");

u8x8.setCursor(0,4);

u8x8.print(" three seconds");

} else {

//if accel<1.5

u8x8.setFont(u8x8\_font\_chroma48medium8\_r);

u8x8.setCursor(0, 0);

u8x8.print("Accel: ");

u8x8.print(accel);

delay(1000);

}

// clear the string:

inputString = "";

stringComplete = false;

}

//SerialEvent occurs whenever a new data comes in the

//hardware serial RX. This routine is run between each

//time ‘loop()’ runs, so using delay inside loop can delay

//response. Multiple bytes of data may be available.

void serialEvent() {

while (Serial.available()) {

// get the new byte:

char inChar = (char)Serial.read();

// add it to the inputString:

inputString += inChar;

// if the incoming character is a newline, set a flag

// so the main loop can do something about it:

if (inChar == '\n') {

stringComplete = true;

}

}

}