Alternative INTRA

Group 18

Sean Brereton
Aoife Anderson
Conchúr Ó Floinn
Peace Ayomide
Ruth Leavey

Design Portfolio

Date: 26/06/2020

Declaration on Plagiarism
Assignment Submission Form

Name(s): Aoife Anderson, Sean Brereton, Ruth Leavey, Conchúr Ó Floinn, Peace Ayomide
Module Code: INTRA
Assignment Title: Design Portfolio
Submission Date: 26/06/2020

I/We declare that this material, which I/We now submit for assessment, is entirely my/our own work and has not been taken from the work of others, save and to the extent that such work has been cited and acknowledged within the text of my/our work. I/We understand that plagiarism, collusion, and copying are grave and serious offences in the university and accept the penalties that would be imposed should I engage in plagiarism, collusion or copying. I/We have read and understood the Assignment Regulations. I/We have identified and included the source of all facts, ideas, opinions, and viewpoints of others in the assignment references. Direct quotations from books, journal articles, internet sources, module text, or any other source whatsoever are acknowledged and the sources cited are identified in the assignment references. This assignment, or any part of it, has not been previously submitted by me/us or any other person for assessment on this or any other course of study.
I/We have read and understood the referencing guidelines found at http://www.dcu.ie/info/regulations/plagiarism.shtml , https://www4.dcu.ie/students/az/plagiarism and/or recommended in the assignment guidelines.

Name(s): Aoife Anderson, Sean Brereton, Ruth Leavey, Conchúr Ó Floinn, Peace Ayomide
Date: 26/06/2020

Table of Contents

List of Figures

1. Introduction

In light of the current Covid-19 climate, the design brief for the alternative INTRA project was to design an intelligent, automated dispenser for hand sanitization purposes. The following constraints were included in the brief and considered throughout the design process:

- The device must be touch-free with proximity sensors.
- It should be controlled and monitored via network communications, incorporating a Raspberry Pi.
- Refills should be easily accessible and replaced with warning issued when refills are needed
- A flashing beacon to encourage usage with 2 metre proximity sensing.
- The device should record usage statistics and store data for data analysis to be conducted.

This design portfolio shows the design process over the eight week period and includes all of the documents completed throughout. The feedback received from mentors and peers was taken into account and changes and improvements were made accordingly. During the eight week design period, two INTRA teams were merged.

The team consists of students from different areas including mechanical engineering, electronic engineering and computer science. For this reason, the brief was divided into mechanical, electronic and software sections utilising skills from each area and merging them where there was an overlap.

The team used a number of different methods to ensure that team work was evenly distributed and being managed efficiently. At the start of the project all group members got in contact and had a Zoom meeting. Throughout the duration of the project the team had at least 2 Zoom meetings a week (1 with mentors and 1 without) to discuss assignments and divide out the work accordingly. Regular contact was also maintained through a Messenger group chat. A Google document was created for meeting minutes, keeping a record of everything discussed at each meeting including decisions made and feedback given from

mentors. A shared Google Drive folder was created to make collaboration on projects easier by allowing everyone to see what each team member was contributing.

2. Product Design Specification (PDS)

Performance

Physical

The dispensing device must be:

- Mountable on a wall or stand, as well as the option to free-stand on a desk or table.
- Modular
- Composed of parts that are easily accessible and replaceable.
- Easily refilled.

Sensors

The smart dispenser must have the ability to:

- Flash a beacon when a person is within 2 metres of the device to encourage usage.
- Dispense the sanitizing fluid without the necessity to be touched using proximity sensing.
- Signal a refill warning when the fluid level has fallen below a predetermined level.
- Signal a warning when the battery level is low.

Data

The device must have the ability to:

- Record usage statistics such as the time of use and the volume of fluid dispensed at each use.
- Record other details such as times when the beacon flashed but there was no solution dispensed.

Network

The smart dispenser product must be able to be controlled remotely by a Raspberry Pi.

- The Pi will be the interface to the internet for the product.
- The Pi must be able to read sensors, and change and set the volume of solution dispensed over the network.

Environment

Indoor Area

- The product will allow for simple installation in general public settings like:
    - Hospitals
    - Schools/ Universities
    - Shopping centres
    - Businesses
    - Offices

- The dispenser should be stored in secure locations that do not experience extremely hot temperatures due to the flammability of the sanitisation fluid.
- The dispenser should not be located in corridors, exits, or open areas that lead to them.[1][1]
- The dispenser will be unaffected by dirt, dust or insects.

Safety Precautions

- The product should be stored away from all heat and ignition sources.
- The smart dispenser should not be allowed to come in contact with any type of oxidizing agent (such as acetyl chloride) or reducing agent.
- Children should not be allowed to use or access hand sanitizer unless properly supervised by an adult.

Life Span

Short-Term

- The product will be used daily.
- Considering the flashing beacon will encourage usage, the dispenser must be suitable for heavy usage.
- Modular parts must be replaced immediately when/if parts are subject to faults from incorrect use or wear and tear.
- The alcohol solution must be replaced when the Raspberry Pi detects that the solution has fallen below a predetermined level e.g. 5% capacity.

Long-Term

- The product's solution must be changed when the alcohol level in the solution has dropped below 60%.[2][2]
- An expiry date of 3 years must be placed on the smart dispenser after the first use.

Maintenance

- Minimal maintenance is required when the Raspberry Pi sends a warning message.
- Spare parts for the product must be in secure storage.
- Spare alcohol solution must also be kept in storage at room temperature.
- The outer housing should be designed with few crevices for bacteria/dust to gather. It should be sleek and easily cleaned.

[1] https://safety.lovetoknow.com/Hand_Sanitizer_Fire_Hazard
[2] https://www.insider.com/does-hand-sanitizer-expire

Target Product Cost

- The costs of the product will realistically be between €100 and €200.
- This price is strongly affected by the intended market.
- This price is based on the costs of materials, manufacturing, networking, and competitor prices.
- The device should be composed of modular parts to allow them to be manufactured at a large scale and assembled into an end product. This will promote economies of scale and make it more affordable.
- A plastic material will not affect the cost greatly.
- Manufacturing a unique design to match all the criteria will affect the price the most.

Competition

Various automated 'no touch' hand sanitiser and soap dispensers exist on the market. Many brands have adopted and trademarked different touch free technologies including 'TFX' by Purell and 'Intuition' by Tork. Although some competitor products use signal lights for maintenance purposes, there is an opportunity for the reminder beacon light in this design brief to fill a niche in the market. Several existing products are described below:

Micronova™ Touch-Free Dispenser and Soaps[3][3]

- This competitor uses a no-touch dispenser. The dispenser spouts sanitizer when the electronic beam is broken. It is easy to install and easily replaceable.

- However, this product is not smart. It does not flash a beacon on close proximity or send a low-capacity warning sign.

Renergise Automatic Hand Sanitizer Dispenser[4][4]

- This product is a cheap no-touch dispenser that is wall mounted. It uses an anti-drip pump.

- Again, this competitor does not use smart sensors or a Raspberry Pi.

Blue KUB Basic Package[5][5]

- This smart dispenser uses an auto-alert system. It sends notifications to a connected smartphone whenever the product needs to be refilled or when the battery level is low. The product is free standing and must be placed on a counter top at waist level.

---

[3] https://www.fishersci.ie/shop/products/micronova-touch-free-dispenser-soaps-4/p-100333
[4] https://renergise.ie/shop/soap-dispenser/automatic-hand-sanitizer-dispenser-1-100ml/
[5] https://touchland.com/products/blue-kub-basic-package

- The product does not take advantage of data analytics and statistics.

Shipping

The product could be shipped internationally without any problems.

Packing

- The smart dispenser will come with a user manual.
- Depending on the transportation used, fluid-filled refills may have to be packaged separately.
- The product will not come with a floor stand.
- The product will be surrounded by bubble wrap and placed securely in a sturdy cardboard box.

Quantity

Initially, for the purpose of this design project, one proto-type dispensing device will be made.

Manufacturing

There is no machining available to teams, the only fabrication methods available are:

1. Bending
2. Drilling
3. Guillotine
4. Hand tools
5. Punching

There is also injection moulding for the outer shell, which will make the manufacturing process much quicker and easier as once the mould is done, it can be used over and over again as the parts are needed for new hand sanitizer units or to replace broken parts.

Size

- The overall system should be small and kept within a volume of $1m^3$.
- The system should be discrete and not protrude too much from the wall or stand so that it is not an eyesore for any installation.
- The device should be compact so that it can be wall mounted without interfering with surrounding objects.
- For the purpose of hand sanitisation in public settings, several small dispensers can be more effective than one large dispenser.

Weight

The device should be lightweight so that it can be supported on a wall whilst holding a container of sanitizing fluid (approx. 1 litre) without risk of it falling.

Aesthetics, Appearance, Finish

- The device should have a sleek finish that can be easily cleaned so it is desirable to users.
- The device should be easily visible and recognisable. In this case, the smart dispenser does not need a refill viewing window but it should be considered to make the device more identifiable as a sanitiser dispenser.
- It should have no small parts or sharp edges.
- Ideally, it should come in different colours and finishes to suit different settings.

.

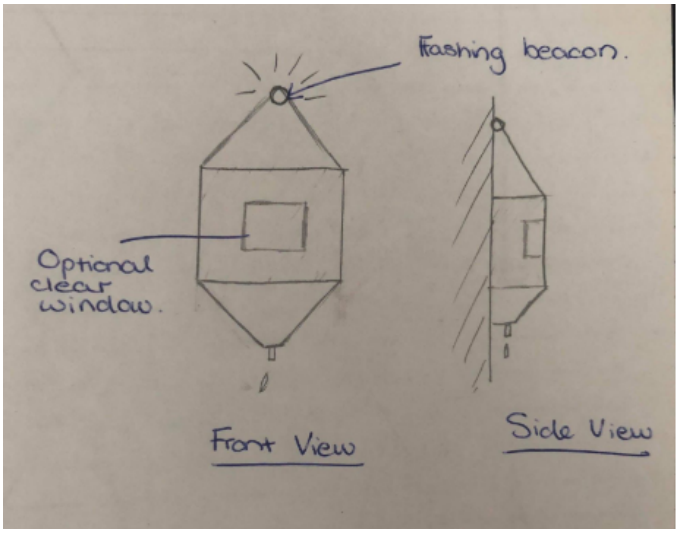
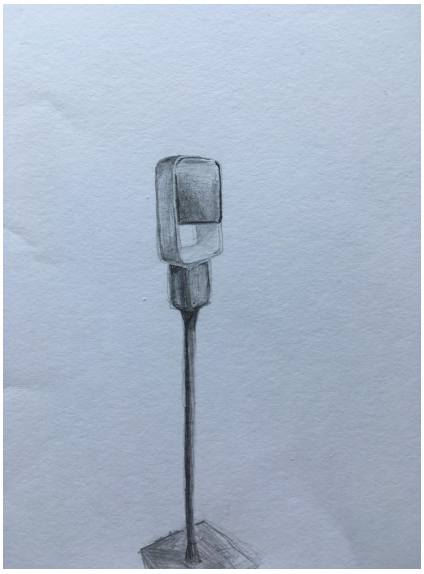


Fig 2.1 Initial design sketch



Materials

- Acrylonitrile Butadiene Styrene (ABS) will be used to manufacture the outer housing cover of the dispenser. This thermoplastic is a suitable material for it's high strength, durability and corrosion resistance. It is also suitable for injection moulding.

- The replaceable refill containers will be composed from Polyethylene (low to medium density). This thermoplastic is suitable for it's excellent chemical resistance to agents like alcohol.
- Other components such as battery packs, sensors and the raspberry pi will be purchased separately for assembly.

Product Life Span Standards and Specifications

Solid modeling will be used from the concept stages to the final design. Full engineering drawings will be made of the system and all individual part drawings in 3rd angle projection will be done for the final design. The full system will be fully dimensioned.

Ergonomics

- Simple installation.

- Dispensers can be free standing or wall mounted. An option of a floor stand of adjustable height will also be made available.
- The product will dispense the sanitizing solution when a user's hand is approximately 50 mm away from the bottom of the dispenser within 1 second of sensor activation.
- A Raspberry Pi board computer will help achieve a man-machine interface which will allow the product to be monitored and controlled remotely.
- Product will be designed to produce a warning signal to notify when it needs to be refilled

Customer

- Predicted buyers are general public utilities like hospitals, businesses and schools.
- Predicted users are the general public who work in these areas as well as pass through.

Quality and Reliability

- Product will be designed so that it will act reliably on a regular basis.
- Manual intervention is permitted in relation to refilling the dispenser when needed and replacing any parts if damaged.
- Modular design assures easy replacement of parts if not working.
- Design to ensure refill possible without difficulty.

Processes

- Fabrication
- Injection moulding
- 3D printing.
- Electronic proximity sensing.
- Networking.
- The recording of usage statistics.
- Data analysis.

Time-scales

The design portfolio, which must include PDS, background review, conceptual analysis, choice of optimum solutions, detailed design work and FMEA, to be completed and submitted at the end of 8 weeks.

Testing

Regular testing methods will be carried out throughout the building and design stages of the project to check for and correct any possible failure as well as ensuring full functionality.

Safety

- Product will be designed to ensure utmost safety regarding operation and maintenance.
- The nature of the device means that electronic components and circuitry will be within close proximity to fluid. This should be considered in the fabrication of the device.
- The device should have no removable parts (unless for maintenance) and no sharp or unfinished edges.
- Appropriate safety practices will be undertaken when assembling/building the product.
- Product to be labelled accordingly to indicate the presence of an alcohol product.
- Safety instructions will be provided to ensure proper and safe operation and disposal.

Company Constraints

- For the construction of the product, some technical supervision will be required by a trained professional during the build phase as well as the testing phase.
- The company will need access to specific resources and equipment for these 4 stages. Accessing these resources and the required facilities will be difficult given the current health emergency and the economic climate in the aftermath of the pandemic.

Market Constraints

- Due to the current health emergency, the product will have to compete with larger companies in terms of pricing which might reduce the profit margin for the company.
- Given the current economic climate, acquiring the necessary funds for the promotion of the product could prove difficult.

Patents, Literature, Product Data

- Since there are numerous versions of this type of product, the design of the product can't conflict with any other patented designs.
- Since this product is going to be used in the interest of health, precautions will have to be followed in what materials are safe to be used with the anti-bacterial liquid to be dispensed.
- A background review document will be completed and consist of in-depth research into each element of the design including parent research, literature reviews and analysis of existing designs.

Legal

- The product has to be given the all clear to by the team in order or it to be built. It must meet the requirements of the Biocidal Products Regulation (BPR) outlined by the HSE in order for production to commence.
- Aspects of the product that are not considered under the BPR must abide by the legislation on Classification, Labelling and Packaging of substances and Mixtures (CLP) as well as other relevant product safety regulations.[6][6]

Installation

- The product must be able to support itself freely.
- A trained personnel may be necessary for the installation of the device within a centralised network. After installation, the user should be able to refill and change batteries as necessary.
- It should have a simple installation process and the materials should be easily replaceable if the product is subjected to any damages.

Documentation

- After the 8-week project timeline has run its course, the team will have a design portfolio establishing the proposed design.
- The group will also give a presentation of the product and each team member will have a document containing all of their individual work in regard to the development of the hand sanitizer product.
- If the device were to be manufactured, it should include a full list of instructions, warnings and disposal advice.

Disposal

- Each product will have multiple uses so the product life will be multiple years.
- The hand sanitizer disposal unit will be made up of various materials, mainly plastics so these plastics will have to be recycled when the product does become damaged beyond repair.
- Ideally, most of the individual parts should be recyclable or reused where possible (fluid container).
- Electronic components should be recycled where possible as per regulations.

3. Background Review

This section consists of the in-depth research carried out on existing designs, patents and literature relevant to the project.

[6] https://www.hse.gov.uk/news/hand-sanitiser-manufacture-supply-coronavirus.htm

Research Methods

The following methods were used to obtain the information gathered in this deliverable:

- Existing product research.
- Review of appropriate patents via DCU library, google scholar.
- Review of available journals and articles which correspond to the project via DCU library, google scholar.

Existing Products

1. **Brand:** Purell
   Name: TFX Touch-Free Dispenser
   Price: €29.80

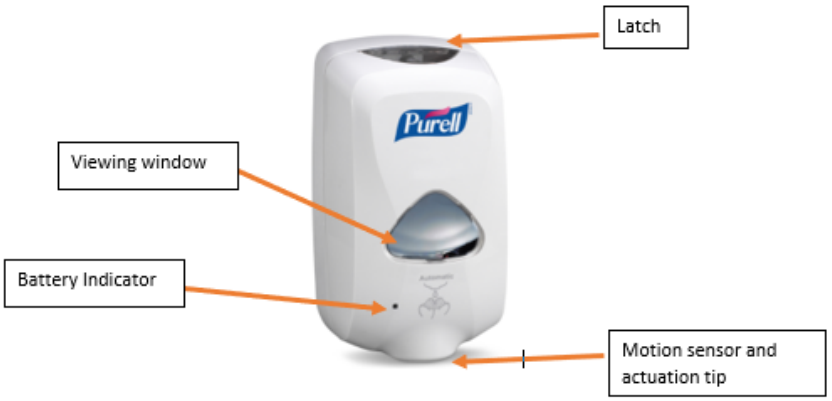


Figure 3.1: Purell TFX Touch-free dispenser

Product Features:

- 1200ml capacity refills
- 1.2 ml doses (1000 doses per refill)
- Trade-marked TFX™ touch-free technology- motion sensors.
- Battery operated- 3 C-size alkaline batteries
- 30,000 cycles before battery requires changing- LED indicator for low battery.
- Sight window
- Sanitary sealed refill.
- Mountable on a wall using a sticker or screws. Compatible with a mobile hand-sanitising station.
- Available in white or metallic finish.
- Dimensions(mm): H-266.7 L-154 W-106
- Net weight: 0.79kg [7]

Pros:

- Large refills for less maintenance
- Touch-free dispensing prevents contamination.
- Recyclable refill bottles.
- Variety of mounting options for different spaces.
- Different finishes for different settings eg. metallic may be suitable for professional settings like offices.

Cons:

- No locking system to prevent misuse.
- The neutral colours may be passed by and overlooked.

2. **Brand:** Purell
   Name: LTX-7 Touch-Free Dispenser
   Price: €49.80

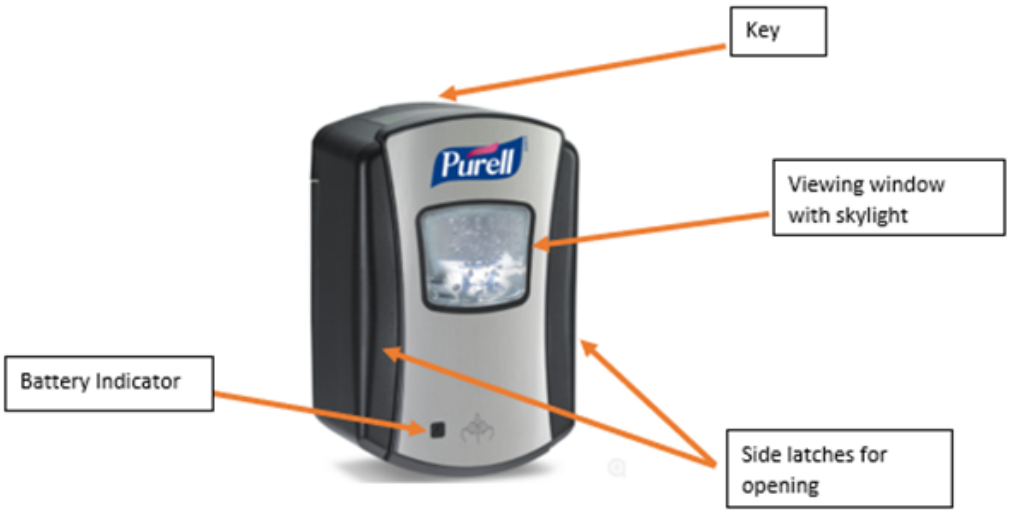


Figure 3.2: Purell LTX-7 Touch-free dispenser

Product Features:

- 700ml capacity refills
- Sight window with light.
- Uses Trade-marked GOJO LOCK OR NOT™ making the dispenser lockable with a key.
- Collapsible refills.
- Dimensions(mm): H-219 L-145.8 W-101
- Net weight: 1.34kg [8]

Figure 3.3: GOJO LOCK OR NOT™

Pros:

- Compact design with smaller refill capacity for areas with limited space
- Locking system to prevent tampering.
- Recyclable, collapsible refill bottles.
- Available in LTX-12 with larger, 1200ml capacity for areas with more traffic.

Cons:

- Cost- product is more expensive than other dispensers with a higher capacity.
- Despite 700ml refill capacity, the dispenser weighs significantly more than other dispensers (TFX touch-free 1200ml dispenser).

3. **Brand:** Purell
   Name: CS8 Touch-free Hand Sanitiser Dispenser
   Price: Not specified

Figure 3.4: CS8 Touch-free hand sanitiser dispenser

Product Features:

- Battery source integrated with every refill.
- Large sight window
- 1200ml capacity refills
- Mountable by adhesive stickers or screws.
- Constructed of ABS plastic.
- Drip tray to protect walls and floors from splashes.
- Employs GOJO LOCK OR NOT™ locking system.

Pros:

- Integrated battery/refill can cut down on maintenance i.e. replace both at the same time.
- Lock and key system prevents misuse.

Cons:

- If the refill becomes empty before the battery dies and vice versa, the battery would have to be replaced along with the refill which would lead to wastage.

4. **Brand:** Tork
   Name: Foam Soap Dispenser with Intuition™ sensor (Elevation Line)
   Price: Not specified

Figure 3.5: Tork foam soap dispenser with Intuition™ sensor

Product Features:

- Compatible with 800ml or 1000ml soap refills
- Trade-marked Intuition™ sensor to automatically dispense soap.
- LED refill indicator
- Available in black or white.
- Dimensions(mm): H-278 L-128 W-112
- Net weight: 0.69kg [9]

Pros:

- Compatible with different size refills.
- Suitable for foam soaps and foam hand sanitisers.
- Lightweight in comparison to other dispensers
- Sleek design for easy cleaning.
- Compatible with Tork EasyCube® facility management software (See section below).

Cons:

- May be overlooked due to the simple design and muted colours.

- Although suitable both soap and hand-sanitising fluid, the dosage for each may vary to be effective. If the volume of each dose of hand sanitiser is too small like that of soap, it will not be effective.

5. **Brand:** Tork
   Name: Foam Soap Dispenser with Intuition™ sensor (Image Line)
   Price: Not specified

Figure 3.6: Tork foam soap dispenser with Intuition™ sensor

Product Features:

- Trade-marked Intuition™ motion sensor for automatic dispensing.
- Stainless steel finish with anti-fingerprint coating.
- Sealed refill bottles.
- Dimensions(mm): H-278 L-130 W-116
- Net weight: 0.937kg [10]

Pros:

- Smooth surface for easy cleaning with stainless steel finish for a modern aesthetic.
- Versatile i.e suitable for foam soaps and foam hand sanitiser.
- Compatible with Tork EasyCube® facility management software(See section below).

Cons:

- The overall design is very simple and could camouflage into surroundings, not encouraging people to use it.
- The dispenser is suitable for foam soaps and foam sanitising fluid. Again, the volume of each substance varies depending on their use. If hand sanitiser is not dispensed in the recommended doses, it can be ineffective.

6. **Tork EasyCube software**

Brand: Tork

During this research, minimal existing product examples were found which are currently being manufactured and sold while also incorporating the technical aspect of this project; making use of a microcontroller, networking, data analysis and storage. However, the existing software named Tork EasyCube, a smart rest-room system offered the most relevance regarding these technical areas. The Tork EasyCube has sensor technology which tracks the number of passer-byers and refill levels of dispensers and bins before sending this information to the Tork EasyCube software. This information is collected by the cloud and sends data to a user's Tork EasyCube account, available on phones, tablets and computers.

There are six sensors incorporated. Of these six, the following are of most relevance to the project at hand. A level sensor and sensor communication unit which measures refills in dispensers, built in sensors in dispensers measuring foam soap level and people counter sensors which counts the number of people passing.

Each dispenser connects wirelessly to the information system. The sensors send data and communicate with the gateway via radio on a 2.4GHz frequency band. The gateway is configured for dispensers with specific status types such as 'full', 'empty', 'time to refill'. If the dispenser's status changes, the gateway forwards the info to the system. All data is processed in the application.[11]

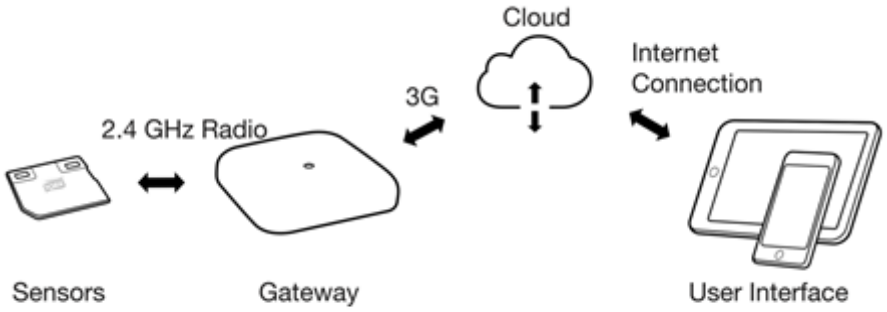


Figure 3.7: Different components of EasyCube

Patents/Literature

Title: Systems For Monitoring Hand Sanitization

Area: Computing and software

This patent details the invention of a hand sanitizer system focusing on monitoring sanitation compliance. It consists of a dispenser along with proximity detector, microprocessor and an alarm which will sound when someone comes into close proximity and continues to alert the user until the dispenser has been used. The microprocessor will receive input from the dispenser switch and proximity sensor and provide appropriate output to the alarm. The

dispenser switch is attached to the dispenser and when this switch is closed, it signifies that the sanitizer has been/is being used.

In this instance the microprocessor mentioned is a Microchip 12F508 microcontroller. It is programmed so that if there is a high signal from the proximity detector and the dispenser switch is not closed, the alarm will be initialized. A delay is also incorporated so that the user has time to reach the sensor before the alarm sounds.

To monitor the usage of the sanitizer successfully, all data associated should be stored in and sent to another device for analysis and recording. The microcontroller would be programmed to count the number of times that someone walks past the dispenser, the number of times the sanitation fluid is dispensed and the number of times the alarm is sounded. [12]

Pros:

- This patent invention makes use of a delay, giving the user time to reach the dispenser before the alarm sounds unnecessarily. This prevents user dissatisfaction with the device.

Cons:

- The use of an alarm noise. This may be very disruptive in certain areas of a workplace or building. It is particularly problematic if someone passes the dispenser without approaching it in a less busy area causing the alarm to sound repeatedly until approached by somebody else.

Title: Sanitation tracking and Alerting system

Area: Computing and software

This is another patent describing an invention focused on the tracking and alerting of a hand sanitizer system aimed at hospitals and healthcare workers. In this invention, there are additional personal devices to be carried by the healthcare workers at all times. The system consists of a communication network which allows contact between a sanitizer dispenser and a central computer containing a database.

The dispensing system should be able to identify the particular worker via worker ID identification on their approach. The monitoring of sanitation usage is based on the specific workers who carry these devices and looks at their individual data as well as the broad usage data of the dispenser itself. A status is created for each worker, either being ‘Clean’ or Unclean” which should correlate to particular business rules regarding how often one must use the dispenser. All information regarding the worker is stored on their personal devices and downloaded periodically to the central database for further analysis.

The patent invention also involves the locations of each dispenser and whether or not this location needs to be taken into account when storing data e.g. if one particular worker is avoiding a particular dispenser.

Wireless technology is used for the workers' devices seeing as they are most likely to be mobile throughout the day. A RFID (radio frequency identification) device is strongly considered to achieve this. The communications network is helpful because it can allow the sanitizer to send information such as the worker identifier to the central computer. This will give a broad idea of which workers are using the dispensers more than others. The software components in the central computer can also present reports on compliance from the information gathered from the database for example stats on the average times the dispenser is used.[13]

Pros:

- The Location identification of this invention is a good way to look at data, especially if dispensers are installed over a large area.
- The use of personal devices which connect to the dispenser is also good as it can look at people individually and increase sanitation use among them .

Cons:

- Privacy issues could appear regarding all healthcare workers' individual dispenser usage being available to others.
- This invention also deals with a smaller target audience of just healthcare workers and doesn't take into account usage which may come from the public.
- This invention also does not seem to be touch-free which doesn't deal with sanitation as effectively as it would have had it been.

Title: Predictive maintenance with sensor data analytics on a raspberry pi-based experimental platform.

Area: Computing and software

This journal documents an experiment regarding predictive maintenance of a raspberry pi-based device involving sensor data analytics. This was undertaken to highlight the importance of predicted maintenance in relation to a product's performance and successful functionality. In addition to this, depending on the results of predictive maintenance, it also allows for much more efficient treatment of the device.

One can either postpone upcoming maintenance or pre schedule it based on the data gathered. In this experiment, a dehumidifier was used alongside various accompanying sensors to help gauge the functionality of the device. An analysis system was also implemented to analyse the recorded data and make model correction for failure predictions as fast and as effectively as possible.

A Raspberry Pi and a PIC18F4525 microcontroller were used as they act as control kernels while also supporting features such as WIFI connectivity, online storage, integration of multiple sensors and a powerful CPU for the processing and analysis of data. The PIC18F4525 microcontroller was included to provide an analog-to-digital conversion interface.

Sensors were used to detect things such as the temperature, vibration and sound.For data analytics the SAS software suite was applied to the experiment to help spot variables and relationships that were deemed important among the data. Interesting information was discovered while carrying out this experiment. One discovery detailed a sensor identified as being too close to drum blades inside the device after noting dramatic changes in wind speed. Appropriate measures were taken to fix this quickly. The results which were provided showed the prediction of the device's future condition as a success, and therefore possible issues can indeed be prevented from happening to the project at an early stage.[14]

Pros:

- This invention highlights the importance of data analytics within a smart device which incorporates a microcontroller and sensors. Although this patent invention differs in device functionality, it is important to be aware of possible problems which may come in the future and relates to any notifications our project's dispenser will send regarding sanitation volume and alcohol level.

Title: Real-time and location-based hygiene monitoring and notification

Area: Computing and software

A smart dispenser is discussed here to send notifications to another component in the system over a WIFI network once a sanitizing substance is dispensed. It is also integrated with real-time location systems (RTLS). The real-time information correlation system mentioned here is a developed system which joins all areas together in real time. It needs to provide immediate data analysis. The smart dispenser is designed to communicate directly to the central system. In the instance that the RTLS unit is not functioning, the smart dispenser should still be able to communicate with the system independently.

An Arduino Mega 2516 microcontroller board is used as well as an Arduino WIFI shield board and an on/off switch. The switch is connected to the Arduino microcontroller and is there to send electronic signals when pushed. The Arduino Mega is programmed using the Arduino development environment. The Arduino WIFI shield sends and receives to and from the Arduino Mega board. These components work together to send WIFI signals over a local area network making use of the transmission control protocol (TCP) to a server.

A database management system is also incorporated into this. Many tables are provided, a table regarding dispenser information among them which stores dispenser IDs, confirmation of dispense, counter and dispenser location IDs. This table is to be updated in real time in relation to the experiment at hand. Other interesting static tables are produced providing information on rooms, different sections inside rooms and healthcare providers.[15]

Pros:

- Good use of database tables to provide beneficial information.
- Good that the smart dispenser can function independently from the RTLS unit.

Title: Automatic dispenser for hand-sanitizer lotion

Area: Mechanical engineering

The patent is for an automatic hand sanitizer dispenser that is powered using a rechargeable battery pack, recharged by a solar panel located on top of the device. The way in which the dispenser assures that the hand sanitizer fluid is pushed out of the fluid bag is by putting pressure on the bag using the battery pack. The battery pack is pivotally mounted above the fluid bag which allows it to move linearly, putting pressure down on the bag due to its weight, and forcing the fluid out of the nozzle once activated.

This is one problem that can arise if the fluid being used in the hand sanitizer dispenser is viscous. Without pressure forcing the liquid down to the nozzle then the fluid would stay in the bag and be wasted. This solution is a good and easy one that utilizes objects that are already in the dispenser.

Pros:

- Initiative solution to the problem that utilizes objects already in the dispenser.
- Cuts down on waste as it makes sure more of the hand sanitizer fluid is used.

Materials

There are two main materials that the majority of hand sanitizer dispensers are made from, ABS plastic and stainless steel. Examples of devices that are on the market today made out of stainless steel can be seen below:

Fig 3.8

Pros:

- Very durable and strong material and is also corrosion resistant.
- It is easy to keep stainless steel clean, which will be a necessity in a hospital environment for example.
- Stainless has a very clean look and would fit the look of a hospital environment.
-

Cons:

- Stainless steel is hard to work with as it is a strong material. It would be difficult to get any intricate shapes. This can be seen in the examples above where there are only simple bends and curves.
- The stainless steel hand sanitizer dispensers that are on the market, from my research, have been up to five times more expensive than the ABS dispensers.
- Stainless steel isn't easily recyclable, it can't be put in a household bin, it must be dropped off at a metal recycler.

This material would be suitable for the hand sanitizer dispenser however there may be better materials that fit the requirements better.

Below are examples of dispensers that are made out of ABS plastic:

Fig 3.9

Pros:

- ABS plastic is a cheap material that can be made in multiple different colours.
- As ABS can be injection moulded or vacuum formed, it can be made into very intricate shapes.
- It is easily recyclable in a common household bin.

Cons:

- ABS is made out of oil, therefore it's more damaging to the environment.
- Deforms easily if it is not on a heated surface while being printed.
- Emits fumes when it is being 3D printed.

Comparing both stainless steel and ABS plastic, it seems that ABS is more suitable as it is cheaper, easily workable, environmentally friendly and has the right aesthetic.

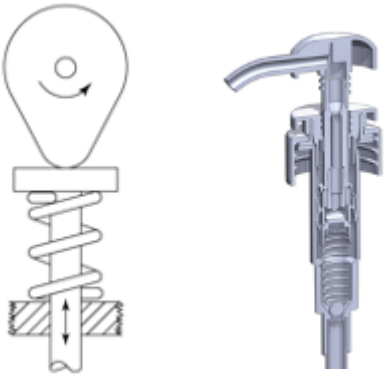
Dispensing Mechanism

There are many different dispensing mechanisms on the market. The majority of hand sanitizer dispensers on the market today are not automatic, and they utilize a hand pump. A few of the automatic hand sanitizer dispensers mechanisms that I found are displayed below.

Fig 3.10

This is one method that is being used in hand sanitizer dispensers. The dispenser is using a peristaltic pump motor. It squeezes the tube that is connected to the hand sanitizer tank, which forms a small vacuum and forces the liquid or gel up the tube to the nozzle. The different dispensers that I have seen on the market are all powered by four AA batteries. I found dispensers being taken apart on youtube to find out how they worked.

Another solution to this dispenser is to use a cam and a regular soap pump. The cam can be driven by a servo so specific amounts of pumps can be done in quick succession if needed or different pumps are available so that the right amount of fluid can be dispensed in each pump.



Another pump that is available on the market is a foam pump. The design is very simple, A container full of hand sanitizer is attached to the top, once the pump is pressed it releases foam. This can be automated by using a cam or some sort of piston for example. The breakdown of the hand sanitizer was found on youtube, as it is difficult to find the individual dispensers without the housing.

Fig 3.11

Fluid Volume

There is evidence to suggest that the efficacy of hand-sanitising fluid and hand-sanitising dispensers is influenced by the volume of fluid used per dose. The volume of hand sanitiser necessary may vary depending on the format of the fluid whether it is gel, liquid or foam. Other factors that may influence the anti-microbial properties of hand sanitiser is the composition i.e. percentage of alcohol and drying time.

Most dispensers contain predetermined doses depending on the fluid, the most common being hand sanitising fluid or soap. Several hand sanitising dispensers on the market, for example,Tork's automated dispensers with Intuition™ sensors (see above) claim to be suitable for both hand-sanitising gel refills and soap refills. If the dispenser gives the same doses for both gel and soap, there is an argument to be made on whether or not the dose can be effective for both formats.

A study carried out on the influence of volume of hand sanitiser on hand-coverage and bactericidal efficacy suggested that doses under 2ml were not as effective as doses above 2ml. The study used three hand sanitisers: 2 gels (70% and 85% ethanol respectively) and 1 foam (70% ethanol). Each fluid was distributed in doses of 1.1ml, 2.0ml and 2.4ml. Fifteen subjects used the different sanitisers aiming to cover their hands entirely and then tested using ultraviolet light. The study concluded that volumes less than 2ml suggest more incomplete coverage than volumes above 2ml. The content of alcohol in each case reduced contamination but only if hands were largely covered. [18]

Sensors

There are a few electrical components needed in the design of our product.
One of the components needed are sensors. This is needed for two different functions. One for alerting individuals to the presence of the dispenser. Another for triggering the mechanism to dispense liquid.

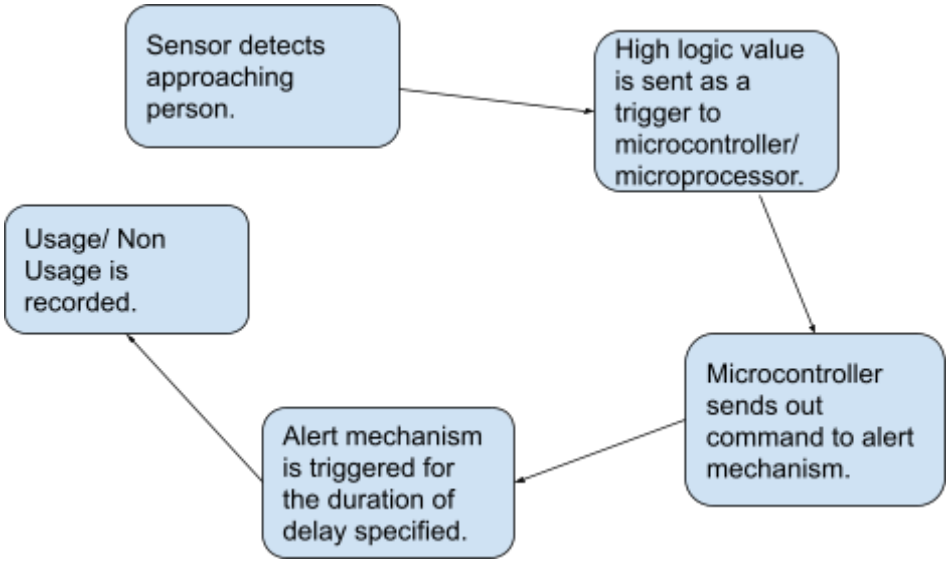


Fig 3.12

There are various sensors that could be used such as:

- Photoelectric sensors
- Passive infrared sensor
- Distance Sensor

1. **Photoelectric sensors(e.g. E3JK)**

Fig 3.13

Technical description

Photoelectric sensors detect objects, changes in surface conditions, and other items through a variety of optical properties. For this particular project we will be considering a diffused sensor. The light source and the receiver are housed in the same device.

Diffused sensors detect objects when the light beam, emitted towards the target, is reflected back to the sensor. The purpose of the sensor would be to send out an infrared LED and based on how much light is bouncing back off an object it outputs a value for the distance to the microcontroller.

Pros:

- Has a built-in amplifier which accepts a wide supply voltage range.
- Slim, space-saving construction measures only.
- Relay outputs with long life expectancy.
- Sensing distance is 5m.

Cons:

- Would be difficult to wire onto a raspberry pi as it would need an adapter for the board.
- The E3JK will malfunction if installed in the following places.
    - Places where the E3JK is exposed to a dusty environment.
    - Places where corrosive gases are produced.

This is not ideal as hand sanitizer contains corrosive substances.

Places where the E3JK is directly exposed to water, oil, or chemicals

2. **Passive infrared sensor(HC-SR501):**

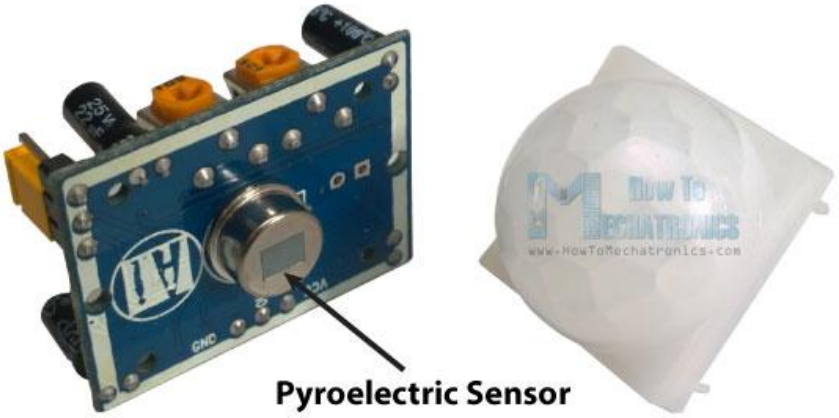


Fig 3.14

Technical description

Passive infrared sensor is a pyroelectric sensor that detects motion by measuring the infrared radiation(heat) emitted naturally by living objects. It is often used in security alarm systems. The word passive means that the sensor is not using any energy for detecting but works with the energy given off by another body. The sensor module consists of a lens called the Fresnel lens which focuses the infrared radiation unto the pyroelectric sensor.

The passive infrared sensor would be ideal for triggering the mechanism that alerts people to the presence of the dispenser.

Pros:

- Sensor is housed in a hermetically sealed metal can to improve noise/temperature/humidity immunity.
- Low power consumption.
- Covers a distance of about 120 degrees and 7 metres.
- Able to distinguish between object movement and human movement.
- Low cost: Price ranges between €1- €10 each depending on choices of retailer e.g. AliBaba, IndiaMart.
- Easily adjustable for various distances.

Example of a circuit schematic that can be used:

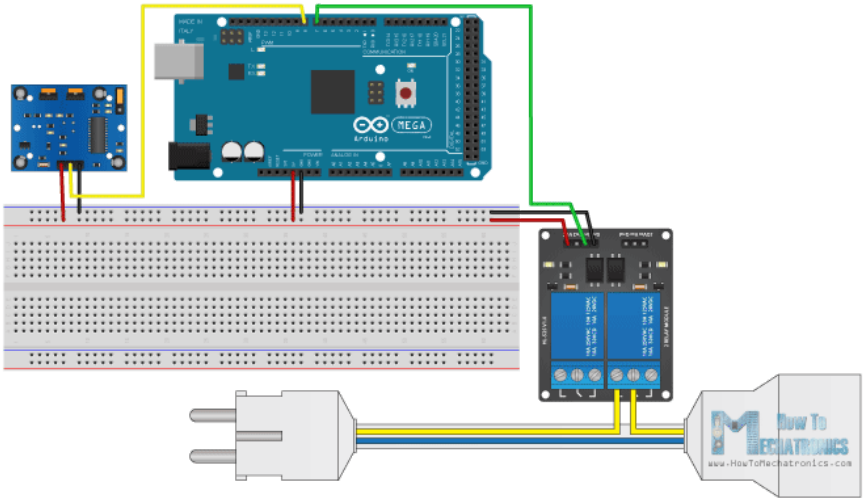


Fig 3.15

In this example the sensor is wired to an arduino board which will trigger a high voltage when an object(e.g. hand) is detected.

The sensor module has three pins: Ground and VCC which is used for powering the module, and an output pin which gives logic level if an object is detected.

The module also consists of two potentiometers, one is for adjusting the sensitivity of the sensor and the other is for adjusting how long the output signal stays high when an object is detected.

There are two modes in which the sensor can be set: Repeatable, Non-Repeatable.

In repeatable mode the lamp will turn off once object is no longer being detected

In non repeatable mode the lamp will turn off based on the length of the delay from the moment it detects IR radiation.

This module is most ideal as it has all the features required in the design for our dispenser.

3. Distance Sensor (e.g. HC-SR501)

Fig 3.16

Technical Description

In particular the HC-SR04 is the most ideal distance sensor. It works as an ultrasonic sensor which emits an ultrasound at 40 000 Hz which travels through air and if an object us in its path will bounce back to the module. It calculates the distance in relation to time and speed of the sound.

This would be ideal to trigger the mechanism that releases the sanitizer fluid from the dispenser.

Pros:

- Has sensing capabilities to sense all material types.
- Not affected by atmospheric dust, rain, snow etc so would be great for outdoor use.
- Can work in any adverse conditions
- It has a high sensing distance.
- Provides good readings in sensing large sized objects.
- Low cost: Price ranges between €2 - €5 each depending on retailers e.g. Digi-Key Ireland, RobotShop.

Cons:

- Sensitive to variation in temperature.
- Difficulties in reading reflections from soft, curved, thin and small objects.

Example of a circuit schematic that can be used:

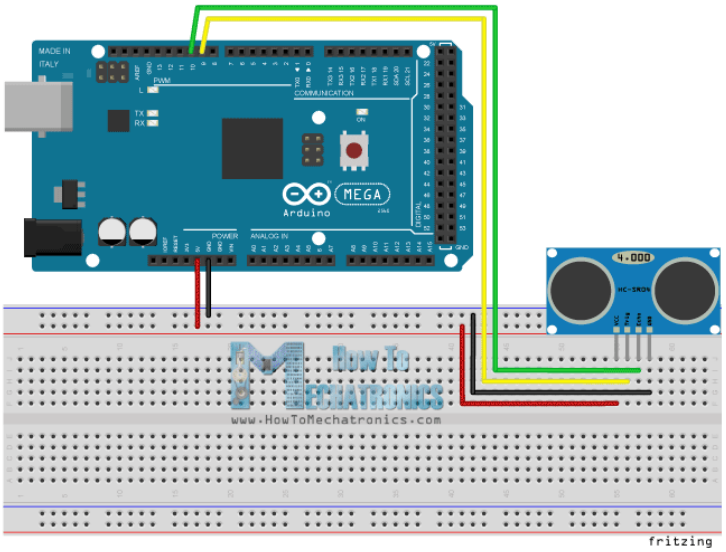


Fig 3.17

- The HC-SR04 module has 4 pins, Ground, VCC, Trig and Echo. The ground and VCC pins are connected to the ground and power pins on the microcontroller respectively. The trig and echo pins are connected to any I/O pin on the board.
- In order to generate the ultrasound you need to set the Trig on a High State for 10 µs. That will send out an 8 cycle sonic burst which will travel at the speed sound and it will be received in the Echo pin. The Echo pin will output the time in microseconds the sound wave traveled.

Computing

Network API

An API (Application Programming Interface) defines the interactions between a client and a server. An API can handle different types of requests. The developed API for the project should be able to:

- Receive data from the Raspberry Pi located in the dispenser.
- Add this data to the appropriate section of the chosen method of data storage.
- Access this data from the database in a readable way and be able to update the database.

There are a number of different web service API's. REST (Representational State Transfer), SOAP (Simple Object Access Protocol), XML-RPC and JSON-RPC are some of the most common types of API types.

During our research on network API's the most popular and most suitable type was a REST API. A REST API communicates with a client and a server mainly using HTTP requests.

The main HTTP methods used for REST API's are GET, POST, PUT and DELETE.

- GET requests data from a web server. This would be used to
- POST is used to upload data to the web server.
- PUT is used to update data already stored.
- DELETE is used to delete data.

For the purpose of this system it is likely that POST and GET will be used the most out of the four methods.

There are a number of design standards regarding API development that should be adhered to. According to Microsoft Azure documentation [25] there are a few main design principles to follow when developing a REST API:

- They are designed around resources (any object, data or service) that can be accessed by the client.
- Clients interact with a service by exchanging representations of resources (mainly using JSON).
- REST APIs use a uniform interface. If the REST API is built on HTTP, use the standard HTTP methods.
- REST APIs use a stateless request model. Any server should be able to handle any request from any client. This increases scalability as there is no unique link between a particular client and server.
- REST APIs are driven by hypermedia links that are contained in the representation.
-

Programming Language

There are many different languages to write API's with. However two of the most popular are python and JavaScript. Both languages have a number of built in frameworks and libraries to help speed up the development of the API. These include Flask for python and NodeJS for JavaScript. In general, an API that is developed in python will be fast, scalable and readable. JavaScript will also provide fast and scalable APIs with easy HTTP support. Either language is a good choice for development.

Data Storage

Data storage is a very important aspect of the project. It is essential for storing data that will be analysed to determine the effectiveness of the dispenser. When researching data storage the two most common options for data storage were cloud based database and physical in house database. A database is used to store data. This data is usually stored in columns, rows and tables. During this research there were a number of pros and cons of each.

A cloud based database runs on a server that is hosted remotely by a service provider. These servers can be all around the world and run together. There are many different companies that provide cloud based server services. Many providers also offer extra services to make database management easier. Some of the most popular are AWS(Amazon Web Services), Microsoft Azure, and Google Cloud. There are many different cloud based service companies available. These services vary in price a lot. In house servers are servers that are physically owned and set up by the owner of them.

Cloud Storage

Pros:

- Very scalable. More servers can be added as needed to increase storage.
- Data backup can be done more frequently depending on the service used.
- No need for a large initial investment into physical servers.
- Easier to set up than a physical server.
- You only pay for how much storage you use.

Cons

- Cloud data storage can be more expensive over longer periods of time for services such as Amazon Web Services (AWS).
- Data storage companies may go bankrupt or close for a number of reasons, making it possible for data to be lost.
- There may be an extra cost on data recovery.
- There may be data security threats from hackers.
- Requires an internet connection to access data.

In House Storage

Pros:

- Data can be accessed without an internet connection.

- Data cannot be accessed by hosting companies.
- You have physical control over your data.

Cons:
- It requires a larger initial investment.
- You are at higher risk of losing data from a technical fault.
- Requires someone for installation.
- Requires someone to maintain it.
- Increasing storage size can be expensive and time consuming.

Data Analysis

The collection and storage of data to help find possible patterns and relationships provide answers to specific relevant questions and allow us to gain insight into the device functionality. In order to obtain this information, the data must all be analysed. To do this, certain methods must be developed which provide a good representation of all the data collected. There are two different methods of analysing data. Data is analysed in either a quantitative and/or a qualitative approach.

Quantitative Data Analysis

Quantitative data analysis collects data, records it and carries out the computation of specific findings using statistical methods. This type of data analysis focuses on looking for the occurrence of certain things happening.

Descriptive statistics, inferential statistics are two methods of quantitative data analysis. Descriptive analysis is a beneficial way to gather absolute figures which help outline specific variables and identify possible existing patterns. Examples of this analysis include mean, mode, median, percentage, frequency and range calculations. Inferential analysis is a method carried out to identify relationships between many variables. Examples of this include correlation, regression and variance analysis.

Graphs are one of the best ways to represent and analyse data. Most quantitative data analysis discoveries can be shown on graphs and provide a much easier visualisation for people to look at and understand. Regarding quantitative data, most graphs either represent the exact data itself or the certain quantities that the data has produced. Data can be displayed in different ways on graphs depending on the scenario. It may be necessary to have a display of all raw data, or to display a particular data set, to display the relationship between many datasets for comparison or to provide probability plots to identify possible future assumptions.[26]

Pros:

- Quicker
- Cost effective
- Allows for generalization

Cons:

- Lack of specific details gathered. E.g. “I like that colour.”

Qualitative Data Analysis

Qualitative data analysis is significantly different to quantitative. This method focuses more on observations and is made up of text. Methods which carry out qualitative analysis are some of the following; Content analysis which looks at responses from users or interviewees and narrative analysis which looks at peoples’ experiences and interaction.[27]

In the computing aspects of this INTRA project, a quantitative data analysis approach is of larger relevance. The dispenser to create will need to analyse specific numerical data corresponding to the amount of sanitizer dispensed, the number of times the dispenser was ignored, etc.

Pros:

- Provides an easily changeable format, in case of useless insight being recorded.
- Provides opportunity to analyse data on a deeper and personal level.

Cons:

- Larger chance of being deemed biased use of subjectivity.

Data analysis examples

The employment of data analysis can be seen throughout many products and systems which exist today.

Many existing wearable devices make use of data analysis. The ‘Fitbit’ is an example of one of these wearables. The Fitbit can be purchased as either an activity tracker or a smart watch. Both are wireless wearable devices which measure data such as steps, heart rate, quality of sleep, calorie intake, calories burned and weight. Up to 31 day’s worth of this data is made available to the user at a time and can be accessed to view their progress.[28]

The delivery company UPS started gathering sensor data in 2003 regarding the routes taken by their drivers. They track their vehicles routes, travel times and safety protocols and send this data into an algorithm called ORION. Major data analysis is then undertaken to establish the most efficient routes and avoidance of traffic. An example of one finding they have come across is that taking a left turn is proven much better than a right turn. Through their data analysis it was proven that right turns in fact take longer and also involve going against traffic so lead to an increased risk of having an accident. [29]

These examples highlight the importance and relevance that data analysis presently has and provides examples of how very beneficial knowledge relating to a particular product, device or system can be discovered through the data analysis process.

4. Conceptual Analysis

This section shows the breakdown of the problem into components. Each component is analysed individually and various concepts are presented before deciding on the one which will be developed further in the final assembly. Most options were evaluated using a weighted matrix table under headings relevant to each concept. The highest ranking option in each case would continue to be used in the final design.

The following concepts will be covered in this section:

- Outer Housing Design
- Material
- Dispenser Mechanism
- Hand Sanitiser Format
- Sensors
- Raspberry Pi
- Power Source
- Network API Approach
- Programming Language
- Data Analysis/Storage

Outer Housing Design

Concept 1

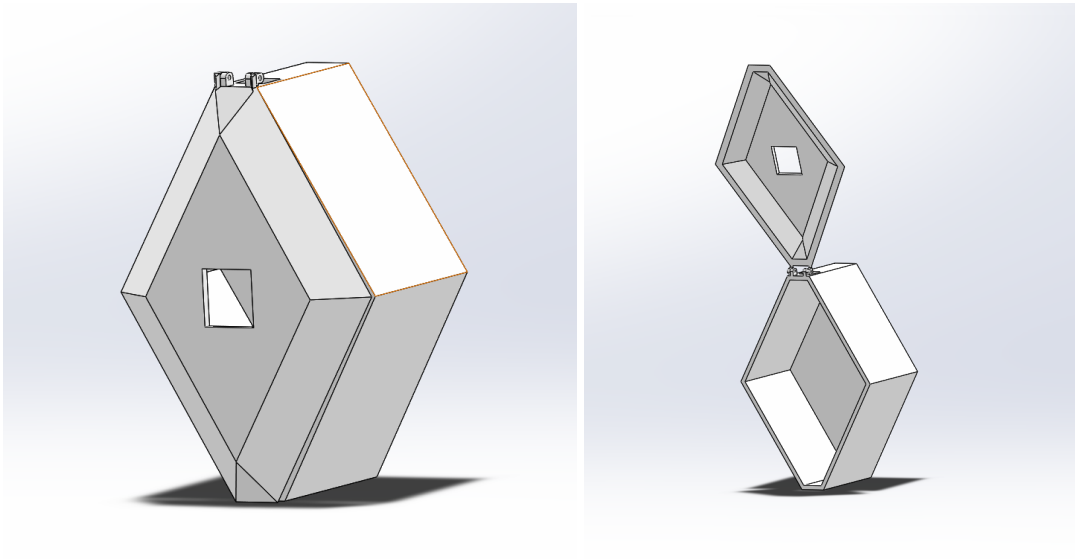


Figure 4.1: Concept design 1

- Diamond shape outer housing design.
- Unique shape for aesthetics.
- Refill viewing window.
- Opening at the top for the flashing beacon.

Concept 2

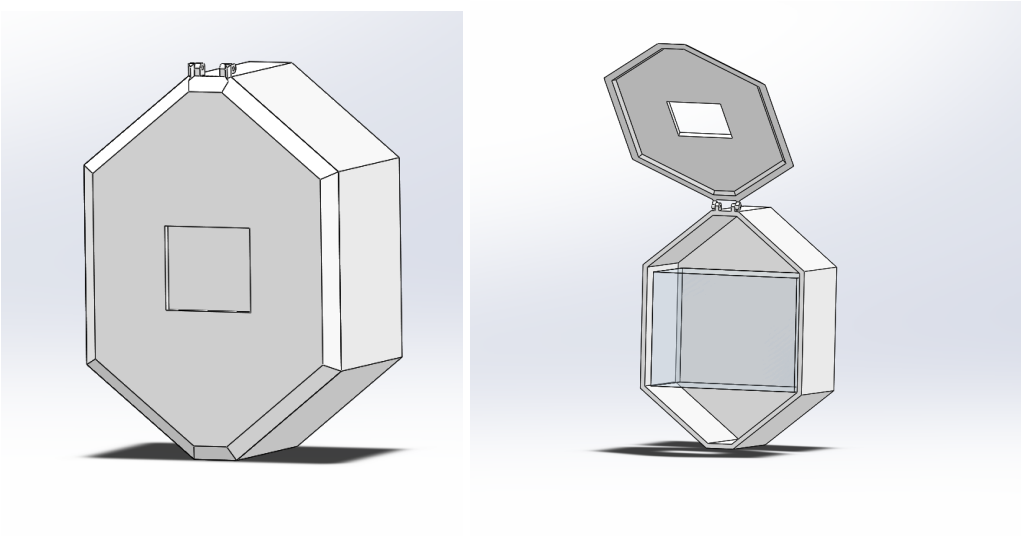


Figure 4.2: Concept design 2

- Similar design to concept 1 with a wider body creating more internal space for refills and other components.
- Refills can be a simple cube shape for ease of manufacturing.

Concept 3

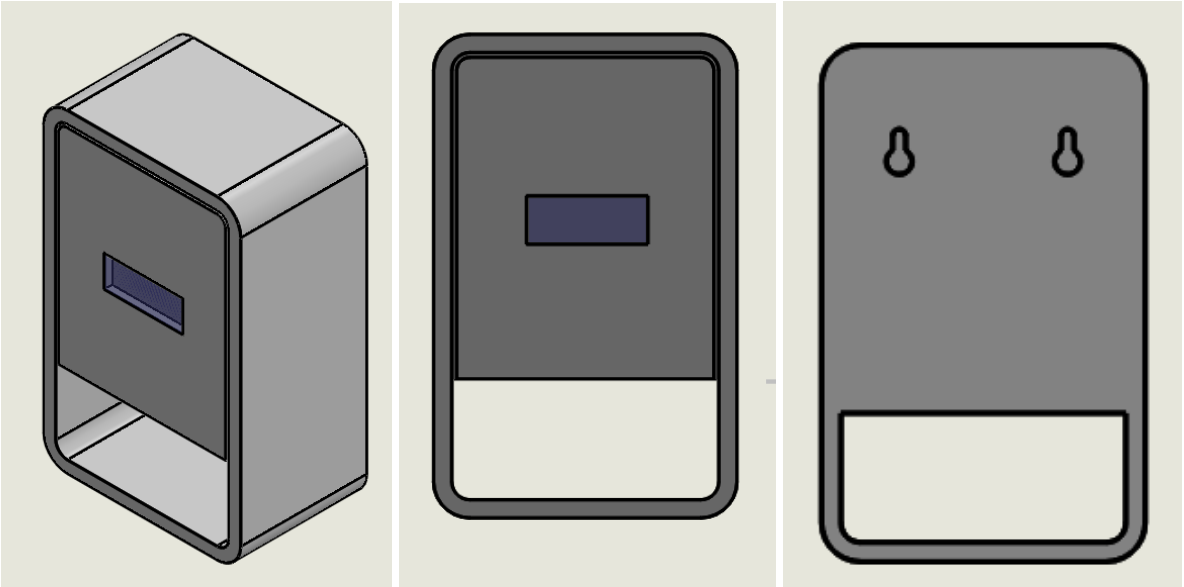


Figure 4.3: Concept design 3

- System has a modern and simple design.
- Can be injected moulded or 3D printed.
- Drip tray underneath to catch any fluid spillage.
- Can be wall mounted, mounted onto a stand or stand alone on a desk or table.
- Internal cavity is small compared to the overall form factor.

Reasoning

The weighted matrix table below evaluates each outer housing design, considering factors such as adjustability, weight, refills, aesthetics, manufacturing ease and simplicity of design. Adjustability accounts for the highest weight as it is important for components including batteries and refills to be easily accessible and replaced. Concept 3 ranks highly due to its simple design, suitability for various locations (wall-mounted, desk) and ease of manufacturing.

	Weight	Concept 1	Concept 2	Concept 3
Adjustability	30%	3	4	4
Weight	20%	4	3	4
Refills	20%	2	4	3
Aesthetics	15%	5	4	5
Manufacturing Ease	10%	3	4	4
Simplicity of Design	5%	3	4	4
Score		3.3	3.8	3.95
Continue		**No**	**No**	**Yes**

Figure 4.4: Concept matrix table

Material

The two materials most suitable for application are Acrylonitrile Butadiene Styrene (ABS) and stainless steel. These materials are commonly used in existing dispensing designs for their advantageous features which can be seen in the table below:

Material 1	Material 2
ABS Plastic	Stainless Steel (Grade 304)
• Thermoplastic • Very durable- high strength and tough. • Good machinability- ABS has a low melting point making it ideal for injection moulding and use in 3D printing. • Aesthetics- ABS can have a high-gloss finish in various colours, for example, yellow dispensers to symbolise Covid-19. • Relatively cheap material and very accessible. • Good dimensional stability. • Recyclable.	• Corrosion resistance- suitable for outdoors unlike ABS. • Extremely durable- very tough with high impact resistance. • Fire resistance- beneficial when housing a highly flammable fluid like hand sanitizer. • Expensive to purchase but long-lasting. • Attractive finish for washrooms and professional settings including offices. • 100% recyclable.

Figure 4.5: Material comparison

Reasoning

The weighted table below evaluates each material based on various factors and their importance based on design requirements. ABS plastic received a higher score in machinability and adjustability, two headings that carry the highest weight at 25%. This is because ABS plastic is easily accessible and straightforward to manufacture through injection molding. This means various shapes and fits can be made easily for the dispenser outer housing etc. There is also the opportunity for the dispenser housing to be designed with different colours and finishes which is not possible with stainless steel. Stainless steel received a higher score in reliability for its durability and corrosion resistance but ultimately, ABS ranked higher overall and will be incorporated into the team's final design.

	Weight	Material 1: ABS	Material 2: Stainless Steel
Machinability	25%	5	3
Adjustability	25%	5	2
Weight	15%	4	3
Reliability	15%	3	5
Cost	10%	4	2
Aesthetics	10%	5	5
Score		4.45	3.15
Continue		YES	NO

Figure 4.6: Material Matrix Table

Dispenser Mechanism

Concept 1- Soap Pump Dispenser with Linear Actuator

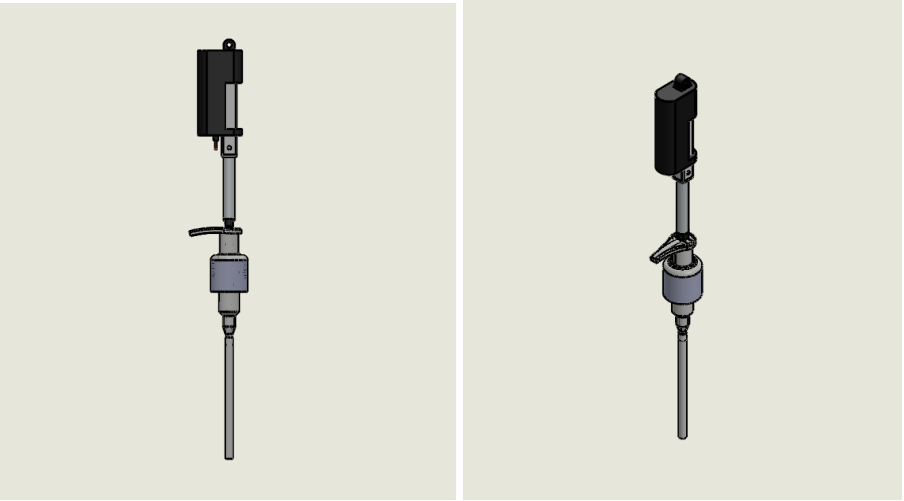


Figure 4.7: Dispenser 1

Pros:

- Simple design with linear actuator applying pressure on hand pump.
- There are very few, simple components that should not cause any issues.

Cons:

- The actuator takes up a lot of space.
- The operation of the actuator is slow, therefore multiple pumps in quick succession will take a considerable amount of time.

Concept 2- Soap Pump Dispenser with Cam

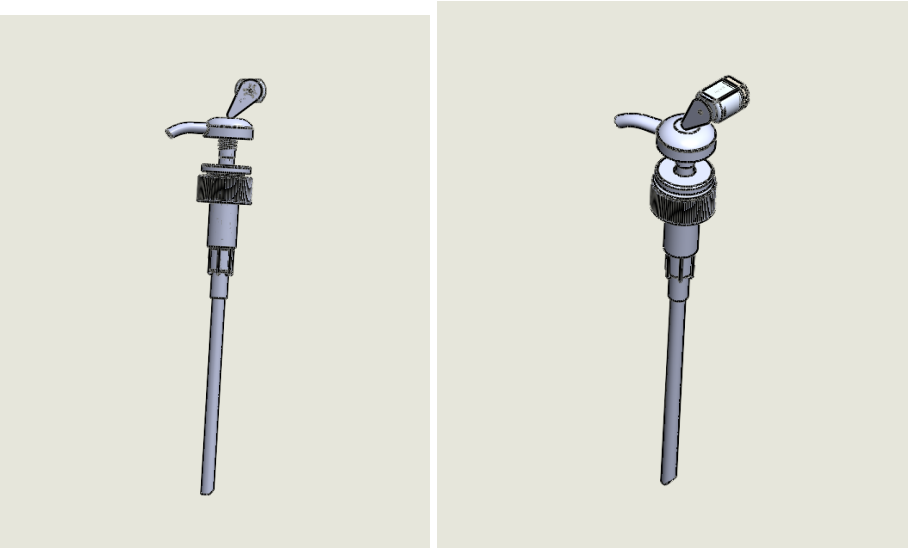


Figure 4.8: Dispenser 2

Pros:

- Compact design that could be easily implemented into any hand sanitizer dispenser.
- Very cheap components to be purchased and manufactured.

Cons:

- Manufacturing of the cam would have to be done.
- The cam could slip from the top of the soap dispenser and cause it not to function.

Concept 3- Motor Driven Piston

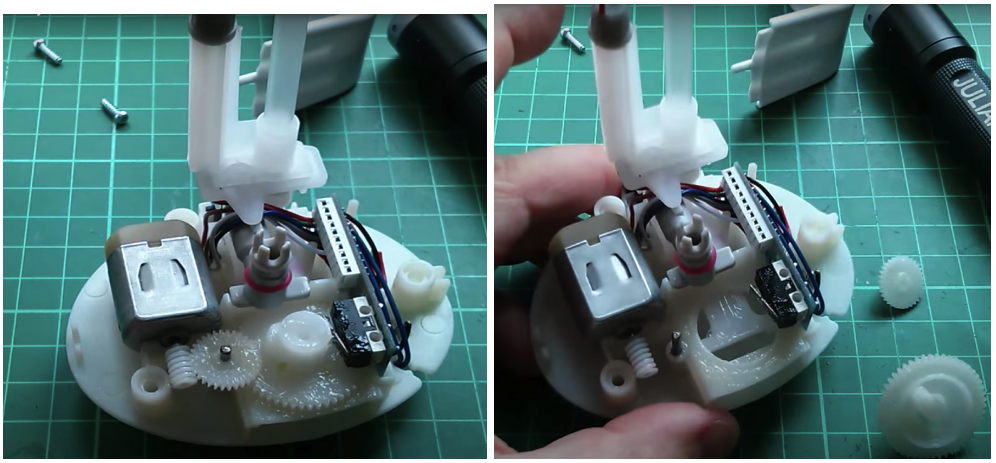


Figure 4.9: Dispenser 3

Pros:

- The motor can build up pressure using a piston in quick succession for multiple uses.
- Compact system that could be easily implemented into any hand sanitizer dispenser.

Cons:

- Many moving parts with a high margin for error.

- Time consuming manufacturing.

Reasoning

The matrix table below analyses each dispensing mechanisms under headings such as manufacturing ease, simplicity of design and effectiveness. The table concluded that concept 1, the soap pump dispenser with linear actuator was the most suitable. This dispenser is the easiest to manufacture as it is not composed of lots of parts. It is effective in dispensing the fluid and because of it's simple design, it is easy to maintain.

	Weight	Concept 1	Concept 2	Concept 3
Manufacturing Ease	15%	5	4	2
Weight	10%	4	5	3
Simplicity of Design	15%	4	4	2
Effectiveness	25%	4	4	5
Reliability	20%	4	3	2
Adjustability	15%	5	5	5
Score	100%	4.3	4.05	3
Ranking		1	2	3
Continue		**YES**	**NO**	**NO**

Figure 4.10: Dispensing mechanism matrix table

Hand Sanitiser Format

When choosing a hand sanitiser fluid format, the team's preference was a gel format. The table below highlights the main features of each format. Although foam and liquid are commonly used and have benefits, gel was chosen for its efficacy for sanitisation purposes, cost-effectiveness and suitability for the chosen dispenser pump. It is also the most commonly used in healthcare settings.

Format 1: Gel	Format 2: Liquid	Format 3: Foam
• Accurate doses • Minimal spillage • Effective coverage • Opportunity to use as a soap dispenser if suitable for all gels. • Less expensive than foam. • Uses more product than foam per 'dose' • Recommended for healthcare settings.	• Does not cling to hands as well as gel or foam which decreases efficacy. • Will inevitably splash onto clothes and surrounding surfaces. • Can be wasteful if it spills and drips a lot. • Longer drying time.	• Clings to hands well for full coverage. • No spills or splashes. • Most expensive option. • Delivers less product per dose because air is added. • The addition of air means less active ingredient (alcohol) is dispensed. • Recommended for educational settings, offices etc. • Fastest drying time.

Figure 4.11: Sanitiser comparison

Dispensing Sensor

Concept 1- E3JK (Photoelectric Sensor)

- Fast response time
- Senses various materials
- Quite expensive costing approximately €100.
- Can malfunction when exposed to dust or corrosive substances.
- Over a long period of time lenses can become contaminated.

Concept 2- VCNL4010 (Proximity Sensor)

- Sensor is easy to use with the Raspberry Pi due to its i2c capabilities.
- Inexpensive, costing approximately €3.
- It is effective for detecting objects within a short distance.
- Unable to detect objects over 200mm away.

Concept 3- GP2Y0D21YK0F (Distance Sensor)

- Inexpensive, costing approximately €10.
- Effective for detecting objects directly in front of it.
- Unable to read objects at angles.
- Unable to detect objects over 200mm away.

	Weight	E3JK	VCNL4010	GP2Y0D21YK0F
Manufacturing Ease	15%	5	5	5
Safety	5%	2	4	4
Simplicity of Design	5%	2	5	4
Effectiveness	40%	5	5	5
Reliability	20%	3	5	4
Cost	15%	2	5	5
Score		3.17	4.83	4.5
Ranking		3	1	2
Continue		NO	YES	NO

Figure 4.12: Dispensing sensor matrix table

Alert Sensor

1. **HC-SR04 (Ultrasonic Sensor)**
    - Has sensing capabilities to sense all material types.
    - Not affected by atmospheric dust, rain, snow etc so would be great for outdoor use.
    - Can work in any adverse conditions.
    - It has a high sensing distance.
    - Inexpensive, costing approximately €5.

2. **HC-SR501 (Pyroelectric)**
    - Sensor is housed in a hermetically sealed metal can to improve noise/temperature/humidity immunity.
    - Low power consumption.
    - Covers a distance of about 120 degrees and 7 metres.
    - Ability to distinguish between object movement and human movement.
    - Low cost sold for approximately €5.
    - Sensor is easy to use because of it's 3.3V TTL logic.
    - Sensitive to variation in temperature.
    - Difficulties in reading reflections from soft, curved, thin and small objects.

3. **OPT3101RHFR (Proximity Sensor)**
    - Can work in any adverse conditions.
    - It has a high sensing distance.
    - Sensor is easy to use with the Raspberry pi due to its i2c capabilities.
    - Quite bulky in size.

	Weight	HC-SR04	HC-SR501	OPT3101RHFR
Manufacturing Ease	15%	5	5	3
Safety	5%	4	4	5
Simplicity of Design	5%	4	3	4
Effectiveness	40%	2	5	4
Reliability	20%	2	4	3
Cost	15%	5	5	1
Score		3.67	4.33	3.33
Ranking		3	1	2
Continue		NO	YES	NO

Figure 4.13: Alert sensor matrix table

Power Source

1. **Kumann Lithium Battery Pack**
   It is very cost effective coming in at €20. It is a portable power supply made for the Raspberry Pi. It's capable of giving the dispenser power for 9 hours before needing a recharge. It can easily be attached unto the Raspberry PI with a case.

2. **PiJuice HAT**
   PiJuice HAT is the easiest one of the concept power supplies to use as it is made for the raspberry pi. The PiJuice HAT is expensive with a 12000mAh version coming in at €33.58.

3. **9 Volt Battery**
   This concept would be very easy to come by and implement, however many 9V batteries would need to be purchased to power the hand sanitizer dispenser. This would lead to a lot of waste and would cost a considerable amount of money in the long run.

	Weight	Kumann Lithium Battery Pack	PiJuice HAT	9 Volt Battery
Manufacturing Ease	15%	5	5	5
Weight	5%	5	4	1
Simplicity of Design	5%	4	5	3
Effectiveness	40%	5	5	2
Reliability	20%	5	5	2
Cost	15%	5	3	2
Score		4.83	4.5	2.5
Ranking		1	2	3
Continue		YES	NO	NO

Figure 4.14: Power source matrix table

Raspberry Pi

The table below compares the potential Raspberry Pi's for the dispenser. It compares each one with features in mind including ethernet, storage type, processor speed, wi-fi, RAM, and voltage.

Features	Raspberry pi 3 model B	Raspberry pi 2 model B	Raspberry pi model B	Raspberry pi model A+
Ethernet	Yes	Yes	Yes	No
Storage	Micro SD	Micro SD	Micro SD	Micro SD
Processor Speed	1.2GHz Quad-core processor	900GHz Quad-core processor	700MHz single core processor	700MHz single core processor
Wi-Fi	Built-in	No	No	No
RAM	1 GB SDRAM of 400MHz	1 GB SDRAM of 400MHz	512 MB SDRAM of 400MHz	256 MB SDRAM of 400MHz
Max power draw/voltage	2.5 amps/5 Volts	1.8 amps/5 Volts	1.8 amps/5 Volts	1.8 amps/5 Volts

Figure 4.15: Raspberry Pi comparison

Reasoning

The weighted table below analyses each Raspberry Pi and ranks each one on the basis of their suitability for this project. The Raspberry Pi 3 Model B was chosen mainly because of its processor speed, WIFI capabilities and RAM size which can be seen in the comparison table below. While requiring the most power draw of the raspberry pi compared, all these

factors combined show that this Raspberry Pi would be most efficient for the functions needed in our dispenser.

	Weight	Raspberry pi 3 model B	Raspberry pi 2 model B	Raspberry pi model B+	Raspberry pi model A+
Manufacturing Ease	15%	5	5	5	5
Weight	5%	3	3	4	4
Simplicity of Design	5%	4	5	5	3
Effectiveness	40%	5	3	2	2
Reliability	20%	5	2	2	2
Cost	15%	3	3	2	3
Score		3.83	3.53	3.33	3.17
Ranking		1	2	3	4
Continue		YES	NO	NO	NO

Figure 4.16: Raspberry Pi matrix table

Network API Approach

	Weight	SOAP	REST
Design	10%	4	4
Approach	15%	4	4.5
Statefulness	10%	4	3
Caching	15%	1	4
Security	10%	4	3
Performance	25%	3	4.5
Message format	15%	3	4
Score		3.15	3.7
Ranking		2	1
Continue		NO	YES

Figure 4.17: API matrix table

REST API Approach

Some of the following factors of REST API helped us in making this decision:

- Limited bandwidth and resources contributes to a more positive performance which is important for this project.
- Completely stateless operations which induces visibility, reliability and scalability.
- Caching
- JSON format which is less verbose, faster and readable than xml.

Programming Language

1. **Python**
    - Open Source
    - Generally quite quick to write
    - Very readable and easy to maintain
    - Flask and Django frameworks

2. **Java Script**
    - Open Source
    - Quick run speeds.
    - Not as readable as Python
    - Node Js Express framework.

	Weight	Python	JavaScript
Library Support	25%	5	4
Performance	20%	3	5
Ease of use	15%	5	3
Scalability	20%	4	3
Maintainability	20%	4	3
Score		4.2	3.65
Ranking		1	2
Continue		Yes	No

Figure 4.18: Programming language matrix table

Data Storage

1. **In-House Server**
    - Physical control over data
    - Data accessed without internet connection.
    - Prevents data from being accessed by hosting companies.
2. **Cloud Storage**
    - Scalable
    - Frequent backup of data.
    - Payment based on how much storage is used.

	Weight	Cloud	In-house
Scalability	30%	5	2
Reliability	20%	4	3
Cost	15%	3	3
Ease of use	15%	3	4
Security	20%	4	4
Score		4	3.05
Ranking		1	2
Continue		**YES**	**NO**

Figure 4.19: Data Storage matrix table

Data Analysis

Data Analysis Techniques

Quantitative

- Data required established i.e. used/ignored/location/refill
- Applicable:
    - Statistical Analysis
    - Diagnostic Analysis
    - Predictive Analysis

Data Analysis Tools

1. **SAS**
    - Large range of statistical functions.
    - Good GUI
    - Quick to learn
    - Expensive

2. **R**
    - Open source.
    - New techniques can be uploaded quickly.
    - Cost effective.

3. **Python**
    - Open source.
    - Supports libraries/functions for statistical purposes.
    - Quick to learn.
    - Cost effective

	Weight	SAS	R	Python
Cost/Availability	20%	3	5	5
Learning	10%	3.5	3	4
Data handling capabilities	30%	4	4	4
Graphical capabilities	25%	3	4.5	4.5
Advancement in tools	15%	4	4.5	4.5
Score		3.5	4.3	4.4
Ranking		3	2	1
Continue		**NO**	**NO**	**YES**

Figure 4.20: Data analysis matrix table

5. Selection of optimum component solutions

Software

Network

A REST API was the chosen type to design and develop for our project.

How will it work in our system?

For the smart hand sanitizing dispenser being designed for this project, REST API will be utilized to provide communication between the client - the Raspberry Pi - and the server. Data communicated will contain useful information regarding the dispenser that can be later analysed to provide efficient usage statistics. This data will be represented in JSON. Each dispenser will have a set of information shown below.

```
{
    "dispenser_id": 12,
    "dispenser_location": [
        {
        "building": "Henry Grattan",
        "room": "G14",
        "start_date": 1-10-2020
        }
    ],
    "dispenser_status": [
        {
        "date": 12-11-2020,
        "time": 9.00
        "Empty": False
        }
    ],
    "usage_count": 0,
    "detection_count": 0,
    "detection": [
        {
        "date": 12-11-2020,
        "time": 9.00,
        "used": True
        }
    ]

}
```

Figure 5.1 Dispenser information

Line 2: dispenser_id: This represents an identifier for each dispenser. The dispenser_id should never change as it is how one dispenser is known from another.

Lines 3-9: dispenser_location: An array of locations for the dispenser. Each object in the array includes the building, the room in that building and the installation date. This may be useful for knowing the location of busier dispensers and so could lead to placement improvements. This information is changed when the location of the dispenser is changed.

Lines 10-16: dispenser_status: The dispenser_status contains an array of statuses which represent the time and date that the dispenser has been identified as empty. Each object contains a Boolean “empty” which changes to TRUE when usage count has reached a predetermined number that signifies the dispenser needs to be refilled along with the respective date and time to show when exactly this was detected.

Line 17: usage _count: This represents the total number of times that the dispenser has been used. As mentioned above, it also helps to notify when the dispenser solution needs to be refilled. This number will be incremented each time the hand dispenser is triggered.

Line 18: detection_count: This counts the total number of times that the dispenser has detected someone within 2m of the device. This number will be incremented each and every time the proximity sensor is triggered.

Lines 19-25: detection: This an array of detection details of the dispenser. Each object includes the date and time of each detection as well as whether the dispenser was successfully used. A new object should be added to this array if the proximity sensor alone is triggered, or if the proximity sensor and the hand sensor are triggered with changes made appropriately.

REST Endpoints

GET /dispensers/ - returns a list of all dispensers and their information.
GET /dispenser/<int: dispenser_id>/ - Fetches all available information for the specific dispenser.
POST /dispenser/<str:name>/ - Creates a new dispenser. The body of the POST request is a JSON object with fields shown in fig 2.1.
DELETE /dispenser/<int:dispenser_id>/ - Deletes the information for the specific dispenser.

Basic Class Diagram

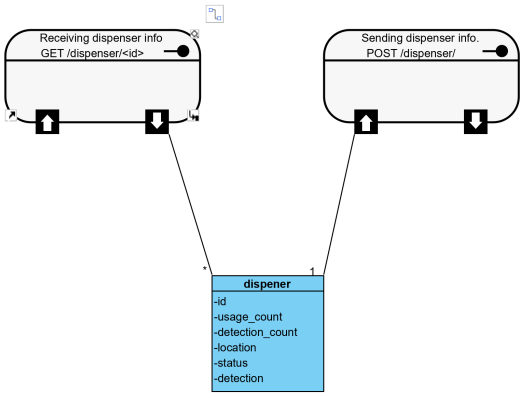


Figure 5.2: Class diagram

Sequence Diagrams

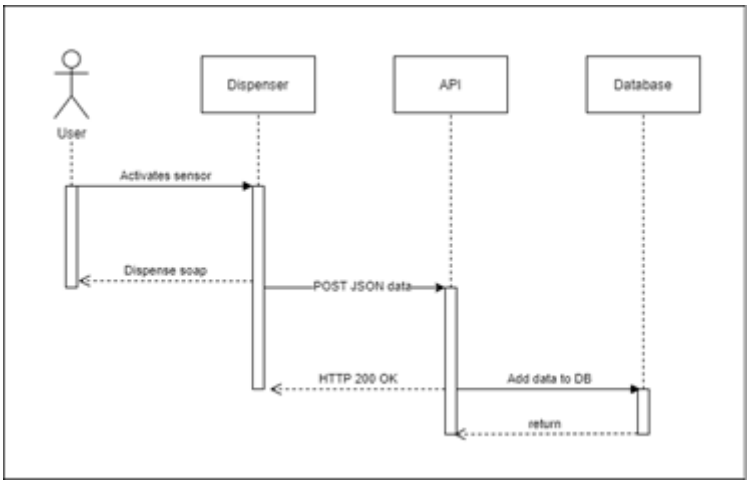


Fig. 5.3 Dispenser being used

This Sequence diagram shows what happens within the software system when sanitizer is dispensed for a user. When a user triggers the sensor hand sanitizer will be dispensed. The dispenser then sends JSON data to the API. The API sends this to the database and adds it to the proper columns.

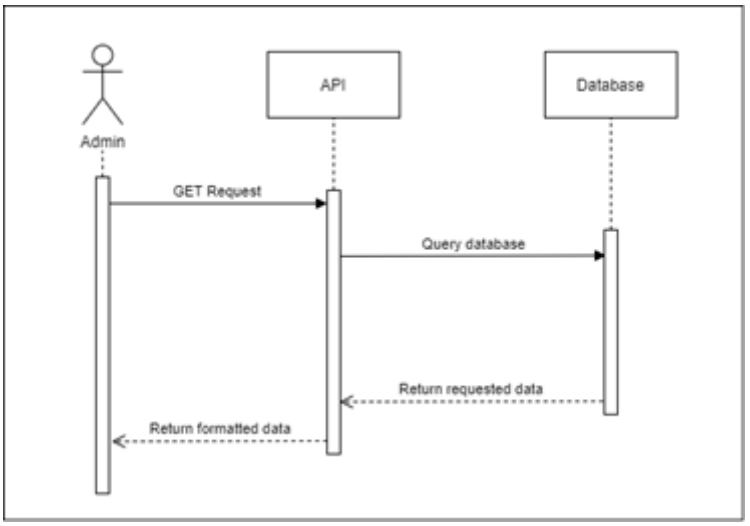


Fig.5.4 Admin viewing data

This Sequence diagram shows the process of an administrator accessing information from the database, using the RESTful API. The administrator can send a request for data to the API. The API uses this request to query the database. The API returns the queried data to the administrator.

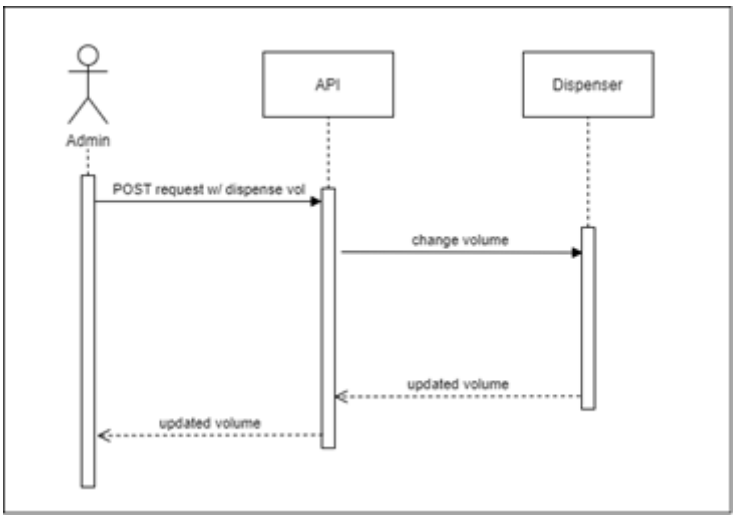


Fig.5.5 Admin changing dispensing volume.

This diagram shows the sequence of an administrator remotely changing the amount of solution that is to be dispensed for a dispenser. The administrator sends a volume amount and dispenser ID to the API. The API will then update a file on the Raspberry Pi. After it has been changed the administrator receives a message that the amount has been successfully updated.

Data Storage

Chosen Data Storage

- A database hosted on a cloud based server was chosen.
- The database will be done using SQL.
- The server will be hosted on a platform such as AWS or Microsoft Azure.

Alternatives

- The alternative is buying physical servers and setting them up.

Reasoning

See Matrix table under ‘Data Storage’ in the conceptual analysis section.

Data Analysis

Data Required

We are aware that in order to conduct efficient data analysis we need to firstly know exactly what data we want to collect.
As a group, it was decided this data includes:

- The location of the dispenser
- How often the dispenser is used
- How often the dispenser is ignored

- Whether or not the dispenser needs to be refilled

For this product and considering the data we want to analyse, there will be a larger focus on quantitative data analysis.
Many analysis techniques were then looked at to establish ones which we believe to be most applicable to the analysing of our data. Techniques decided on include:

- Statistical analysis
- Diagnostic analysis
- Predictive analysis

Analysis Tools

Data Analysis tools make it easier to process and manipulate data, analyse relationships/correlations between data sets and help identify patterns/trends for interpretation. We considered SAS, R and Python as three possible data analysis tools we could use for our product.

- SAS - SAS has been the undisputed market leader in commercial analytics space. The software offers lots of statistical functions and it has a good GUI for people to learn quickly. However, it ends up being the most expensive option and is not always up to date with latest statistical functions.
- R - R is the Open source counterpart of SAS. Because it's open source, latest techniques get released quickly. There is a lot of documentation available over the internet and it is a very cost-effective option.
- Python – Python is an open source scripting language, it also has libraries (numpy, scipy and matplotlib) and functions for almost any statistical operation / model building you may want to do. It is lightweight, stays up to date with latest techniques and also another cost effective option.

These tools were evaluated and compared using the following table concerning areas such as learning, graphical capabilities and advancement in tools. After this comparison, Python was ranked the highest and therefore chosen as the primary tool for our data analysis.

6. Functional Specification

1. Introduction

1.1 Overview

The automated dispensing system will allow the user to access the sanitising fluid without the need to touch the dispenser. Using electronic proximity sensing, the dispenser will sense the user’s hand and deliver a pre-set volume of fluid per dose. It will utilize a flashing beacon as a reminder for people to use. The dispenser will be smart, meaning that is controlled and

monitored remotely. It will gather various usage statistics for data analysis to be carried out offline.

Non-automated hand sanitizing dispensers are often used today. These can become fomites as they require each user to manually dispense the fluid by touching a button or lever. With the current pandemic, touch-free dispensers are extremely important as they prevent the risk of cross-contamination.

1.2 Glossary

API : An application programming interface is a set of routines and protocols that specifies how software components should interact.

REST : A REST (Representational State Transfer) API is an architectural style that uses HTTP requests to POST, GET, PUT and DELETE data.

HTTP : HTTP (Hypertext Transfer Protocol) gives users a way to interact with web resources by transmitting hypertext messages between clients and servers.

POST : The HTTP POST request method requests that a web server accepts data enclosed in the body of the request message.

GET : HTTP GET request is only used for receiving data.

PUT: HTTP PUT request method creates a new resource or replaces a representation of the target resource.

JSON : JavaScript Object Notation is a file format, and data interchange format, that uses human-readable text to store and transmit data objects

2. General Description

2.1 Product / System Functions

This product is a networked hand sanitizer dispenser connected to a centralised management system. The product must monitor when someone is within 2m distance of the device and flash a light when successfully detected. The dispenser must be no-touch; therefore, a user should be able to receive some hand sanitizing solution from the dispenser by simply placing their hand underneath it. A finite volume of solution should be dispensed, and this volume must be able to be changed remotely. When the sanitizing solution is empty, an appropriate notification must be sent via network communications. The device should report on usage statistics of the dispenser to a networked service and successfully store this information so that it can be efficiently analysed.

2.2 User Characteristics and Objectives

Regarding the dispenser itself, the user community consists of the general public. Due to the current Covid-19 pandemic, hand hygiene is more important than ever hence this product is

envisioned to be placed in almost any public utility e.g. schools, universities, hospitals, etc. and be utilized by whomever passes it. Limitations may exist toward anyone with a disability which affects their visual or movement capabilities.

For the access of data and the update of product features e.g. volume-change, the user community would consist of appropriate employees/employers of said public utilities. The device will be designed so that a low level of computer expertise will be acceptable to operate the features available.

2.3 Operational Scenarios

USE CASE 1	Initialise dispenser	
Goal in Context	Turn on dispenser successfully	
Scope & Level	System, Core.	
Preconditions	Dispenser has been installed correctly	
Success End Condition	Dispenser is turned on.	
Failed End Condition	Dispenser is not turned on.	
Primary, **Secondary Actors**	DispenserInstaller. Dispenser, DispenserSystem	
Trigger	DispenserInstaller plugs dispenser into power-source	
DESCRIPTION	**Step**	**Action**
	1	User connects the dispenser to its power-source.
	2	Dispenser powers on.
	3	System initialises default dispenser info.

EXTENSIONS	**Step**	**Branching Action**
	1a	
	2a	Dispenser does not power on: signifies low battery of power source
VARIATIONS		**Branching Action**
		n/a

USE CASE 2	Trigger proximity light	
Goal in Context	User triggers proximity sensor which causes a light to flash	
Scope & Level	System, Core.	
Preconditions	Dispenser installed successfully. Dispenser initialised successfully.	
Success End Condition	Dispenser light flashes.	
Failed End Condition	Dispenser light does not flash.	
Primary, **Secondary Actors**	ApproachingUser. ProximitySensor, DispenserLight, DispenserSystem	
Trigger	ApproachingUser walks within 2m of the dispenser	
DESCRIPTION	**Step**	**Action**
	1	User walks within 2m of the dispenser

	2	Proximity sensor triggered
	3	Dispenser light flashes
	4	System updates detection details
	5	Delay time ends
	6	Light stops flashing
EXTENSIONS	**Step**	**Branching Action**
	3a	Dispenser light doesn't flash.: signifies that the connection needs to be checked
VARIATIONS		**Branching Action**
	1	ApproachingUser passes the dispenser without using it.
	2	ApproachingUser ignores the dispenser and doesn't use it.

USE CASE 3	Dispense of sanitizer
Goal in Context	User receives hand sanitizer
Scope & Level	System, Core.
Preconditions	Dispenser has been installed successfully. Dispenser has been initialised successfully. User has approached the dispenser.
Success End Condition	Hand sanitizer solution is dispensed into the user's hand.

Failed End Condition	Hand sanitizer solution is not dispensed into the user's hand.	
Primary, **Secondary Actors**	DispenserUser. DispenserSensor, DispenserSystem.	
Trigger	DispenseUser places a hand underneath the dispenser sensor.	
DESCRIPTION	**Step**	**Action**
	1	User presents a hand underneath the dispenser.
	2	Dispenser sensor is triggered.
	3	Hand sanitizer solution is dispensed.
	4	Usage details updated.
	5	Appropriate information sent to the server.
EXTENSIONS	**Step**	**Branching Action**
	2a	Dispenser sensor not triggered: maintenance necessary
	3a	Hand sanitizer solution not dispensed: usage details updated, dispenser status details updated.
VARIATIONS		**Branching Action**
		n/a

USE CASE 4	Access data.
Goal in Context	User accesses usage data.

Scope & Level	System, Core.	
Preconditions	Dispenser installed successfully. Dispenser initialised successfully.	
Success End Condition	Usage data accessed successfully.	
Failed End Condition	Usage data not accessed successfully.	
Primary, **Secondary Actors**	DispenserAdministrator, DispenserOwner DispenserSystem	
Trigger	Request made for retrieval of data.	
DESCRIPTION	**Step**	**Action**
	1	User accesses the web interface.
	2	User requests for desired dispenser data.
	3	Correct data returned.
EXTENSIONS	**Step**	**Branching Action**
	2a	Dispenser info requested is invalid: User informed that the dispenser info requested is invalid and asked to try again.
VARIATIONS		**Branching Action**
	1	User requests for all dispensers' data.
	2	User requests a specific dispenser's data by id.

	3	User requests dispenser status data.
	4	User requests dispenser location data.
	5	User requests dispenser detection data.
	6	User requests dispenser use data.

USE CASE 5	Change dispenser location.	
Goal in Context	User changes the dispenser location.	
Scope & Level	System, Core.	
Preconditions	Dispenser installed successfully. Dispenser initialised successfully.	
Success End Condition	Dispenser location successfully changed.	
Failed End Condition	Dispenser location not successfully changed.	
Primary, **Secondary Actors**	DispenserAdministrator, Dispenser Owner DispenserSystem	
Trigger	User sends a request to change the dispenser location.	
DESCRIPTION	**Step**	**Action**
	1	User accesses the web interface.
	2	User sends a request to change the location containing the dispenser id and new location name.

	3	New location updated.
EXTENSIONS	**Step**	**Branching Action**
	2a	Location name invalid: User asked to try again
VARIATIONS		**Branching Action**
		n/a

USE CASE 6	Change volume remotely.	
Goal in Context	User changes the dispenser volume.	
Scope & Level	System, Core.	
Preconditions	Dispenser installed successfully. Dispenser initialised successfully.	
Success End Condition	Dispenser volume successfully changed.	
Failed End Condition	Dispenser volume not successfully changed.	
Primary, **Secondary Actors**	DispenserAdministrator, DispenserOwner DispenserSystem, Dispenser	
Trigger	User sends a request to change dispenser volume	
DESCRIPTION	**Step**	**Action**
	1	User accesses the web interface.

	2	User sends a request containing the dispenser id and the desired volume amount.
	3	Volume change updated
	4	Dispenser dispense mechanism altered to change to new volume.
EXTENSIONS	**Step**	**Branching Action**
	2a	Volume entered is outside the predetermined range and is invalid: User notified and asked to try again.
VARIATIONS		**Branching Action**
		n/a

USE CASE 7	Receives refill notification.
Goal in Context	User receives a notification that the dispenser needs to be refilled.
Scope & Level	System, Core.
Preconditions	Dispenser installed successfully. Dispenser initialised successfully. Dispenser solution used up.
Success End Condition	Refill notification received successfully.
Failed End Condition	Refill notification not received successfully.
Primary,	DispenserAdministrator, DispenserOwner

Secondary Actors	DispenserSystem	
Trigger	Dispenser use reaches the predetermined number signifying that the dispenser needs to be refilled.	
DESCRIPTION	**Step**	**Action**
	1	User receives notification with the dispenser id, location and a message to say it needs to be refilled.
EXTENSIONS	**Step**	**Branching Action**
		n/a
VARIATIONS		**Branching Action**
		n/a

2.4 Constraints

Time: Due to the limited time of two months assigned to this project, it is unlikely a finished design and implementation will be successfully completed for the device. This time constraint will force us to focus on the more critical areas of the design process.

Existing Knowledge: There are aspects of this project that the group have not had thorough experience with before. Raspberry Pi is the required microprocessor for this project. As nobody in the group has worked with a Raspberry Pi before, it is possible that constraints regarding lack of extensive knowledge may arise. The same applies for a Network API implementation. This will require more time to be spent on the learning and understanding process.

Financial: The product would cost a lot of money in materials such as plastics and metals. Electronics would be expensive as the dispenser also needs to be a number of sensors, a battery, and a Raspberry Pi. In terms of software server costs to host the API and database would be a high cost.

3. Functional Requirements

3.1 Dispense soap

Description

- When a user of the dispenser puts their hand under the sensor the chosen amount of hand sanitizer should be dispensed. When the soap has been dispensed usage statistics such as time and location are recorded by the Raspberry Pi as a JSON file.

Criticality

- This is the most critical function of the system. Without this function the system will not work and there is no use for any other functions.

Technical Issues

- There are a few possible issues, mainly one of the dispensers sensors failing or malfunctioning. A dispenser with no sanitizer could also cause issues.

Dependencies with other requirements

- There are no dependencies on any other functional requirements as this is the main function that the rest of the software system is based around.

3.2 Remotely set dispensing volume

Description

- The administrator or owner of the hand sanitizer dispenser should be able to set the volume of hand sanitizer that is dispensed remotely. They can access a web interface for the REST API and send a PUT request to the Raspberry Pi with the dispenser ID and the volume amount they would like to set.

Criticality

- It is important for the owner to have the ability to set the volume remotely so there is an accurate count of the number of doses per refill.

Technical Issues

- The administrator needs to know the ID of their dispenser to properly set dispensing volume. There is a possibility that the PUT request to the Raspberry Pi will fail. There will also need to be a volume range to prevent the administrator setting the volume to invalid or too large amounts.

Dependencies with other requirements

- There is a relationship between the pre-set dispensing volume function and the refill warning function. The refill size and the volume dispensed per use will allow the system to count how many doses the dispenser can deliver before the refill is low or empty. This will then trigger the refill warning.

3.3 Saving data to the database

Description

- When the soap dispenser is used, information with usage statistics is sent to the RESTful API as a JSON file using a HTTP POST request. The API then sends this data to a database and adds it in the correct columns.

Criticality

- This is important for the administrator to be able to see their data. This is necessary for the administrator to be able to view data analytics.

Technical Issues

- The POST request to the API may fail and return a 400 bad request code. The API may also have issues adding data to the database.

Dependencies with other requirements

- This function depends on the Raspberry Pi’s ability to correctly send JSON data to the API.

3.4 Refill warning

Description

- When the hand sanitizer levels are low the on-board Raspberry Pi should send a notification to the administrator that the hand sanitizer levels are low, along with its ID and location.

Criticality

- This is important for the owner of the dispenser system to ensure the dispensers do not run empty.

Technical Issues

- The liquid level dispensers may not always be correct and can give wrong information. Also the dispenser must send the warning to the owner of the device.

Dependencies with other requirements

- This has some dependence on the pre-set dispensing volume to accurately count the amount of volume dispensed for each dose before a refill is needed and a warning signal is displayed.

3.5 Accessing data for data analysis

Description

- An administrator can access data for analysis from the database through a web interface. The web interface should display the data in an easy to understand format.

Criticality

- This critical aspect of the system is not very critical as it requires saving to the database to work properly. It also has no effect on whether or not the hand sanitizer dispenser will work.

Technical Issues

- Incorrect or incomplete data in the database could cause poor data analysis.

Dependencies with other requirements

- A system administrator being able to access data for analysis completely depends on the system's ability to properly add data to the database in

3.6 Alerting user to the dispenser

Description

- When a potential user is within 2 metres of the hand sanitizer dispenser, the sensor sends this data to the Raspberry Pi and this triggers an LED to alert the user of the presence of the dispenser.

Criticality

- This aspect isn't critical as it has no effect on the primary function of the hand sanitizer dispenser.

Technical Issues

- Failure to wire the circuit involving this sensor correctly could cause the circuit to short out causing the sensor to fail.

Dependencies with other requirements

- This function depends on the correct voltage to be supplied from the Raspberry Pi.

4. System Architecture

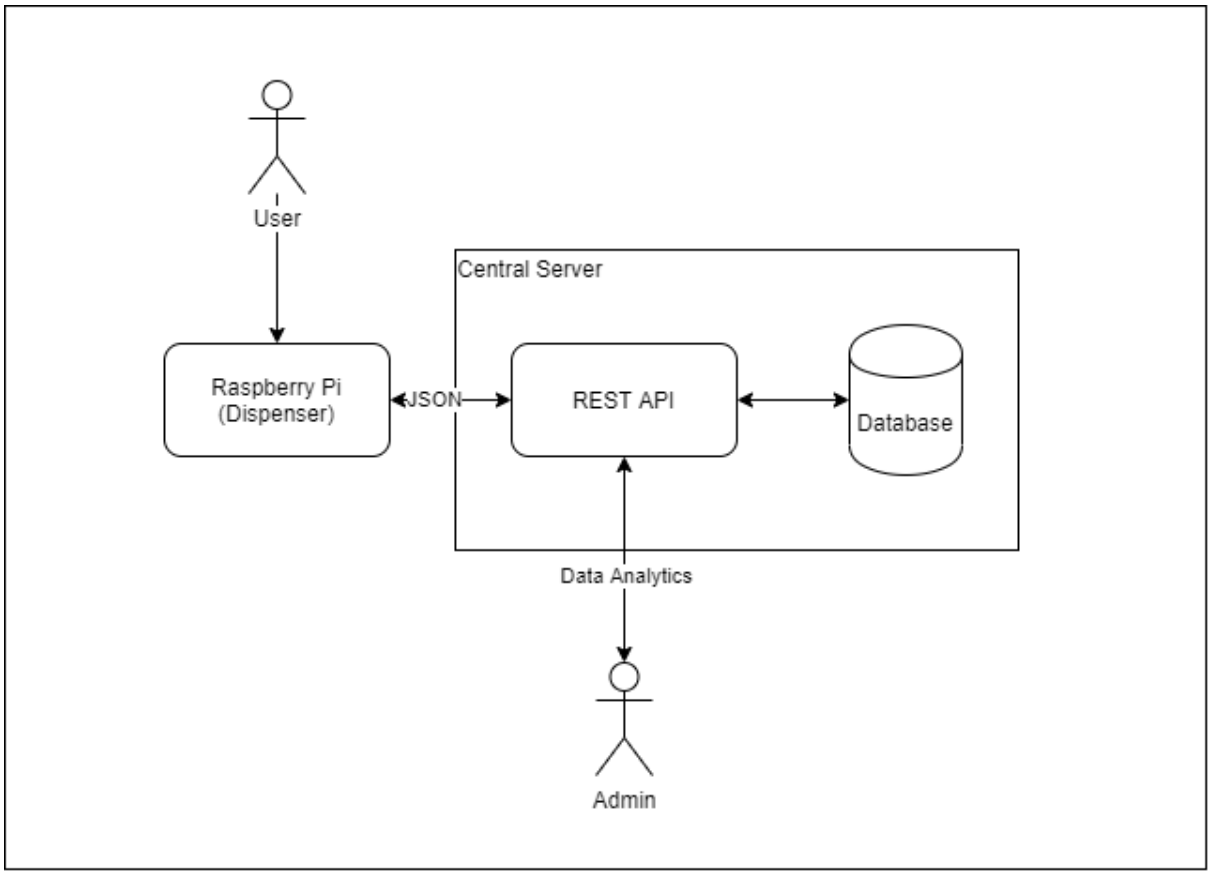


Fig 6.1: System architecture diagram

5. High-Level Design

5.1 Top Level Context Diagram

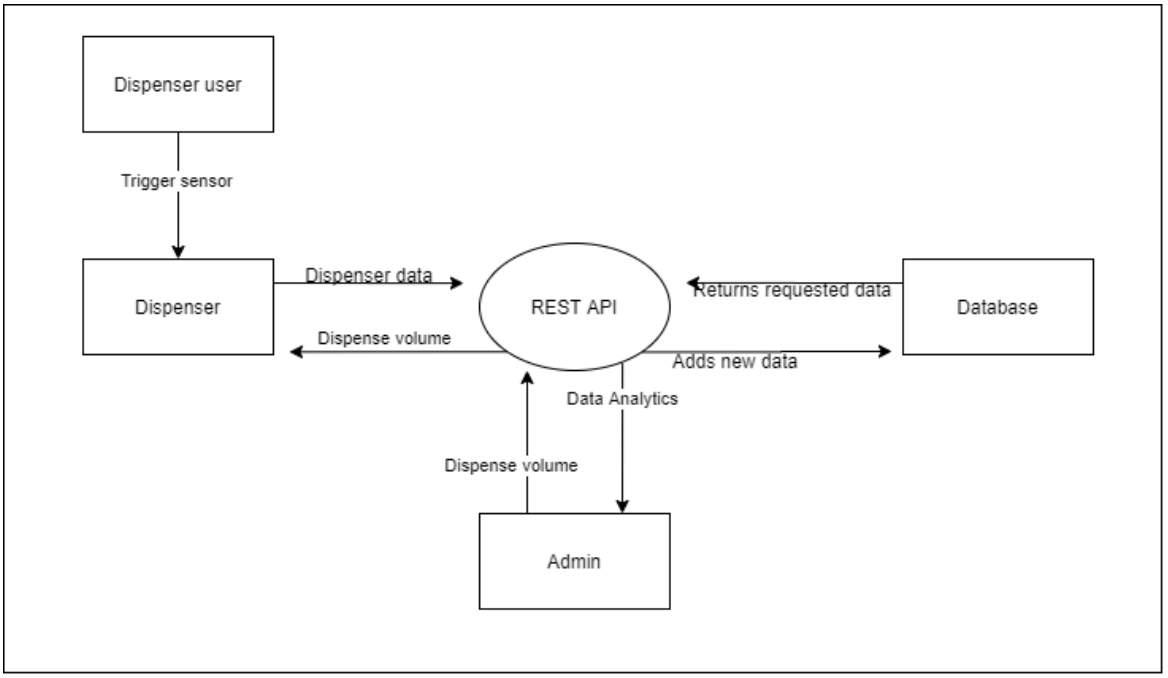


Fig 6.2: Top Level Context Diagram

5.2 DFD

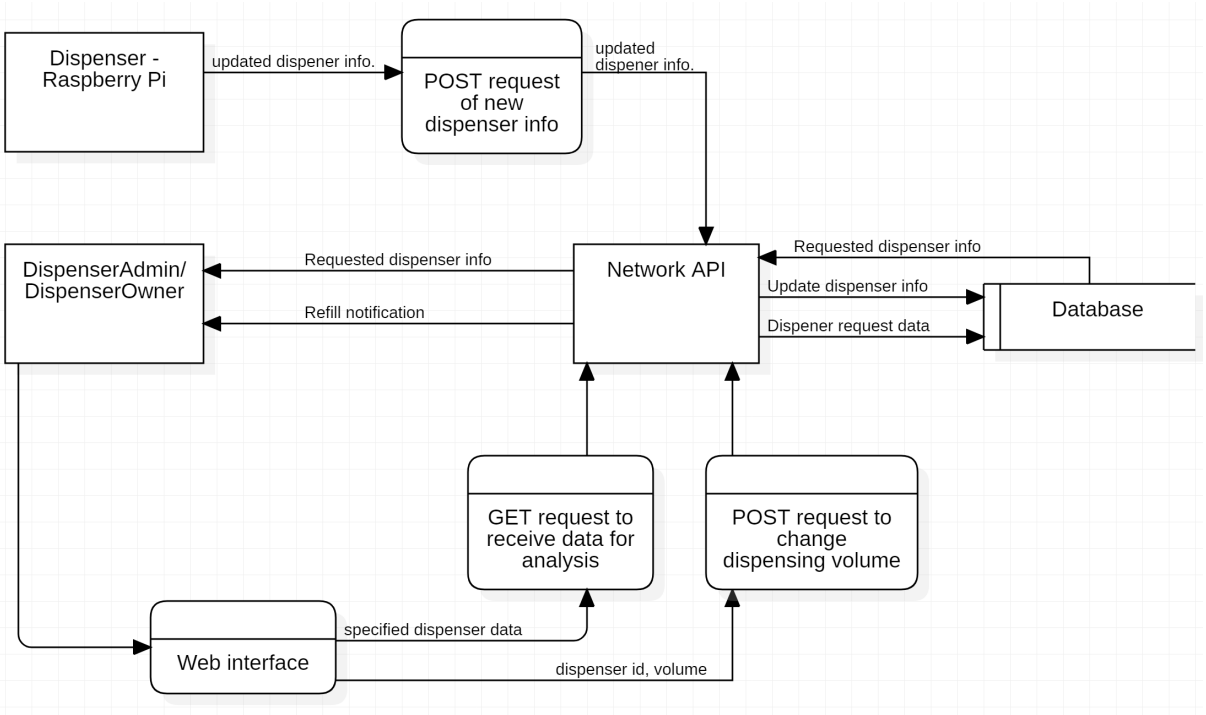


Figure 6.3: DFD

7. Detailed design

Assembly (Using SolidWorks)

The final design had a few changes after it went through the peer review process. The dimensions of the unit were changed as there was unused space inside. This would waste material and waste money. There were two locks added to the unit, one on the front panel to lock the door of the unit and one on the top of the unit to lock it to the wall bracket. The location of the fluid tank and dispensing mechanism were switched so that the fluid tank would be easier accessible to remove and refill. There was a speed controller added so that the speed of the linear actuator could be controlled. Below are isometric drawings of the finished hand sanitizer dispenser.

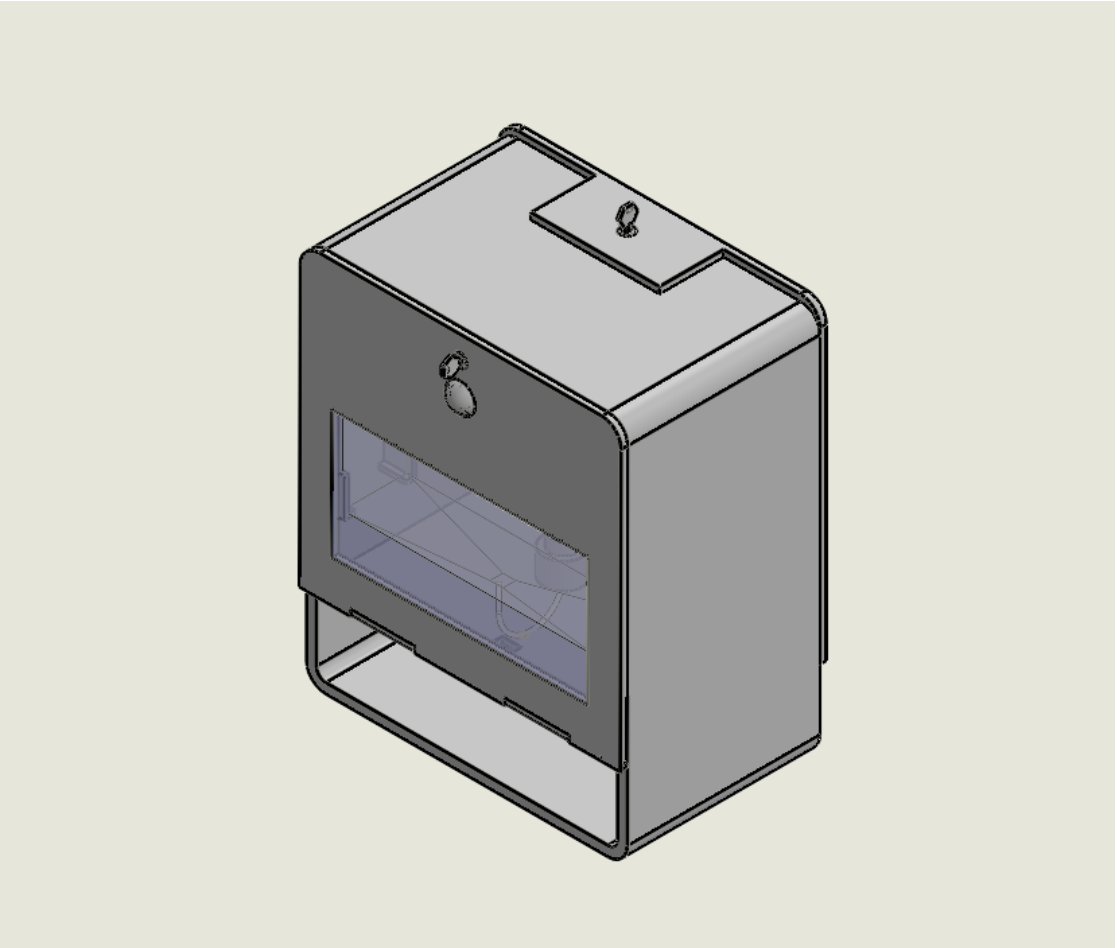


Figure 7.1 Isometric View

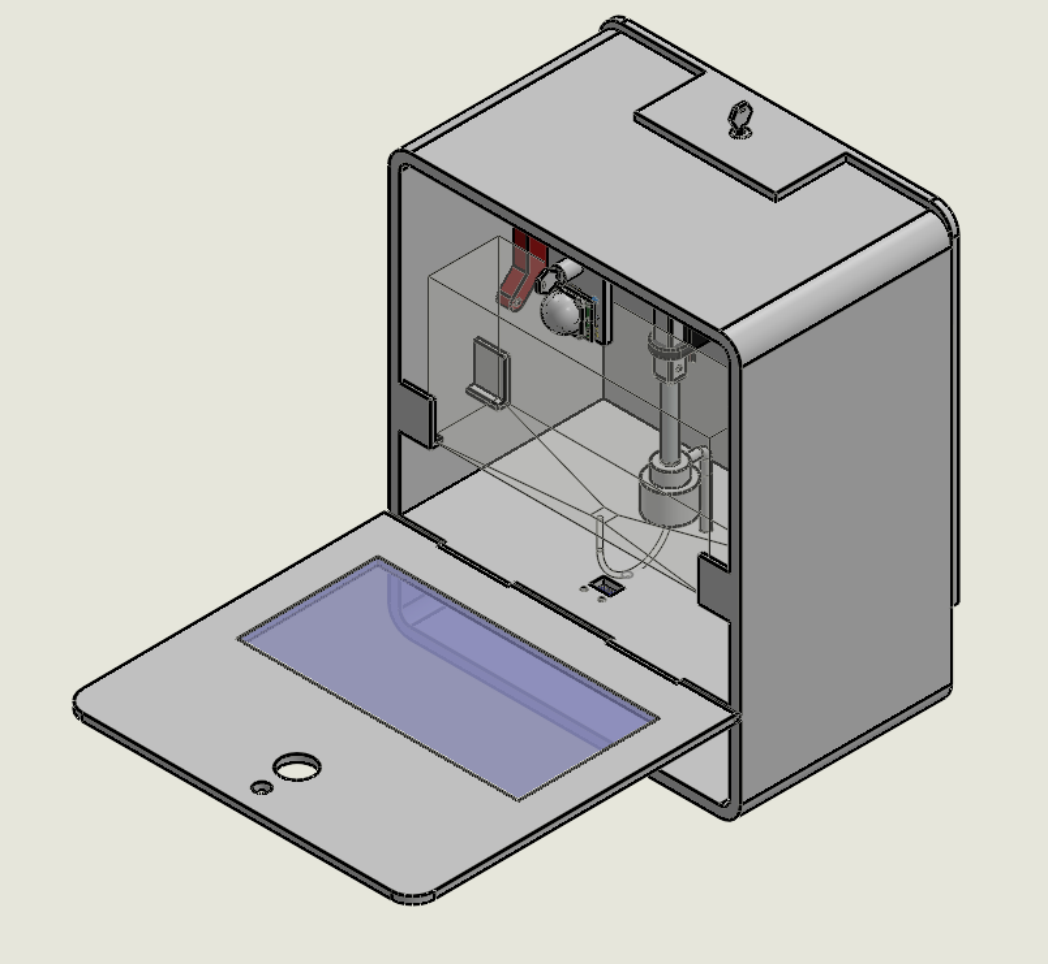


Figure 7.2 Isometric View With Front Panel Open

The images below are all the dimensions of the outside and inside of the hand sanitizer dispenser.

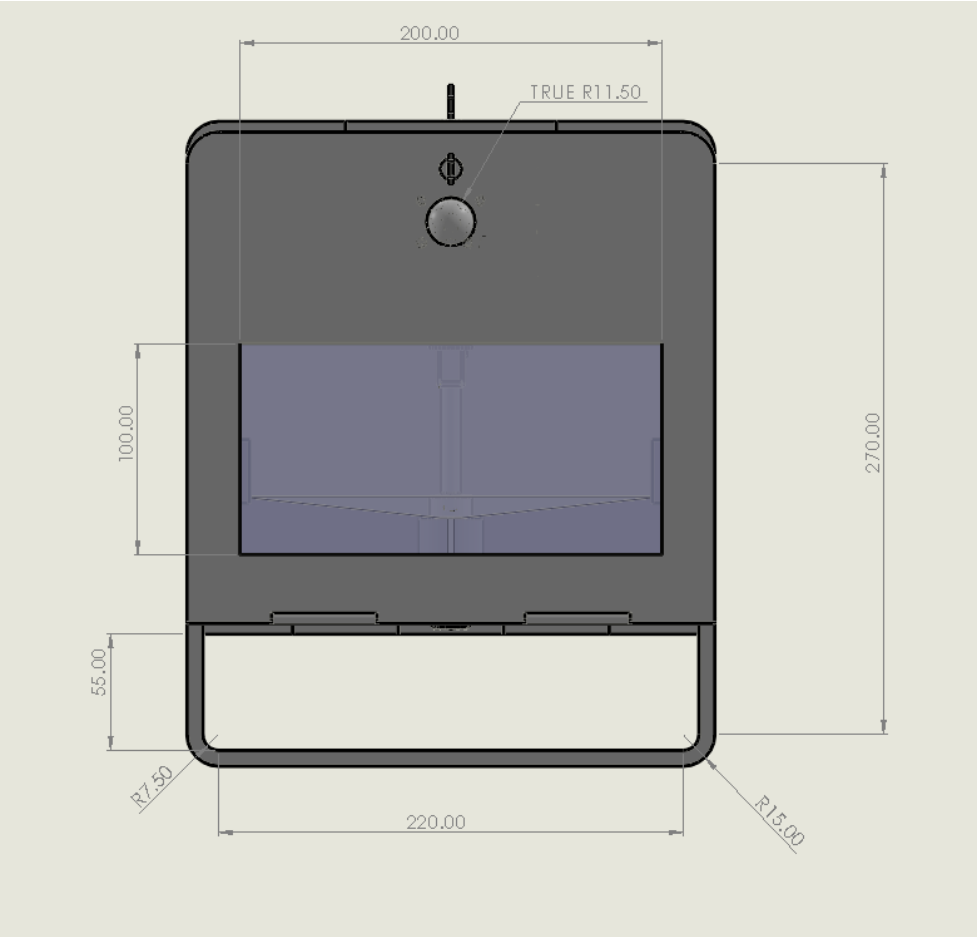


Figure 7.3 Front View With Dimensions

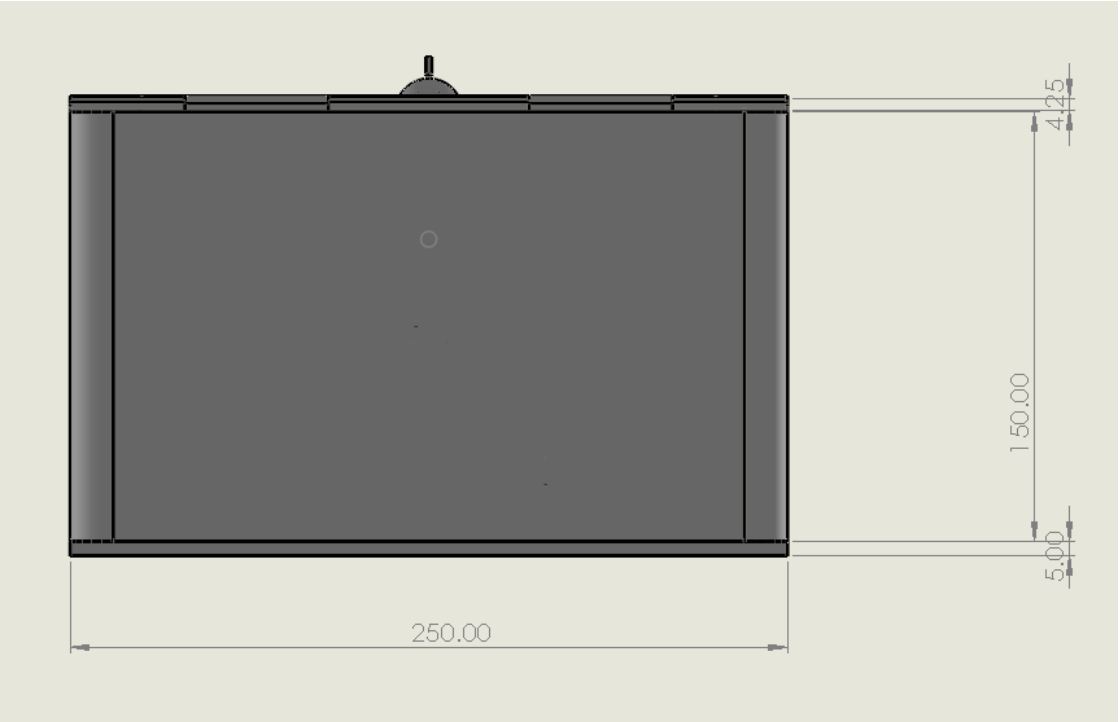


Figure 7.4 Top View With Dimensions

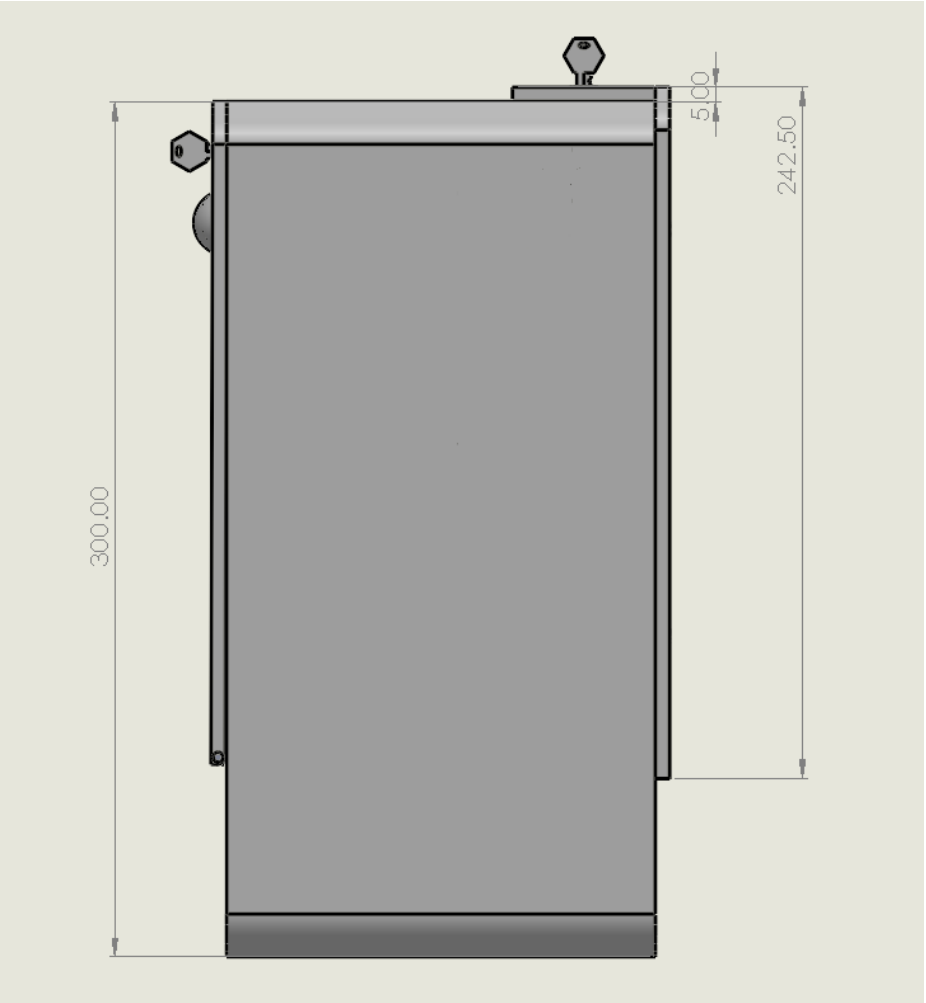


Figure 7.5 Side View With Dimensions

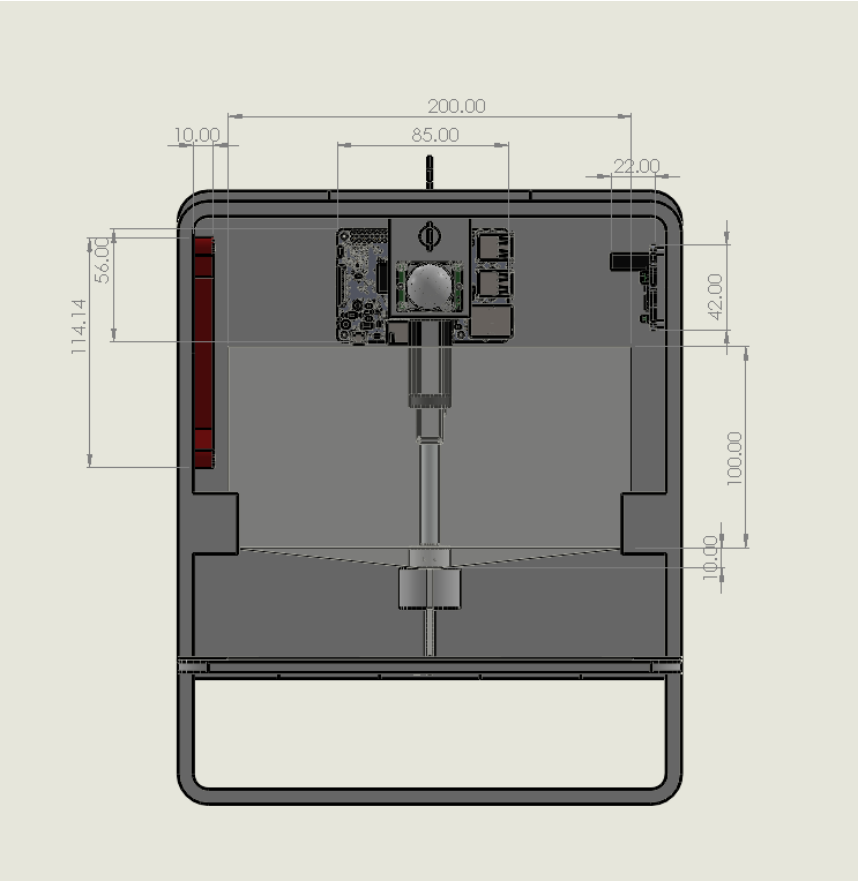


Figure 7.6 Front Inside View With Dimensions

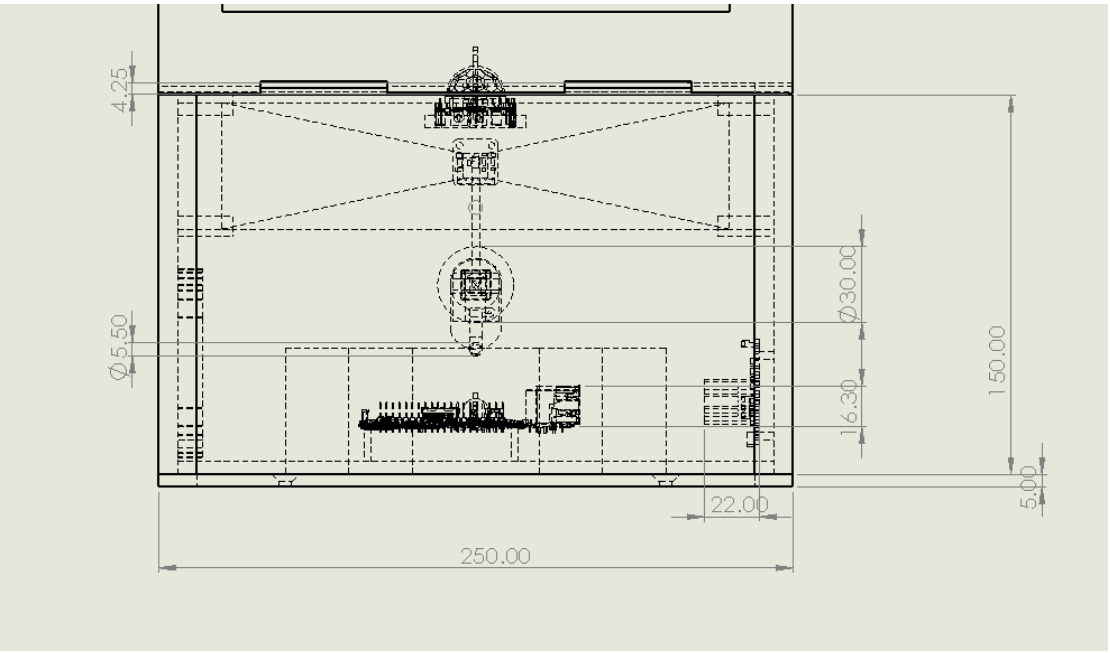


Figure 7.7 Top Inside View With Dimensions

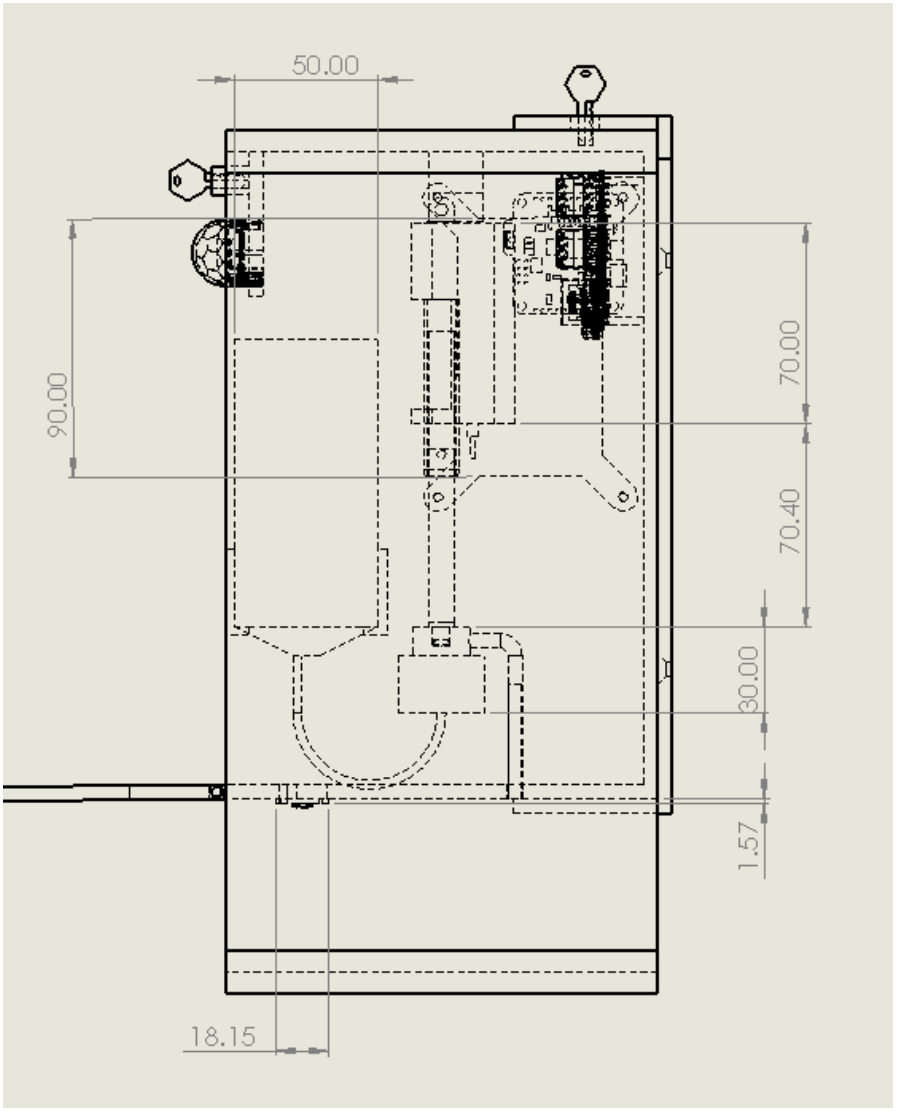
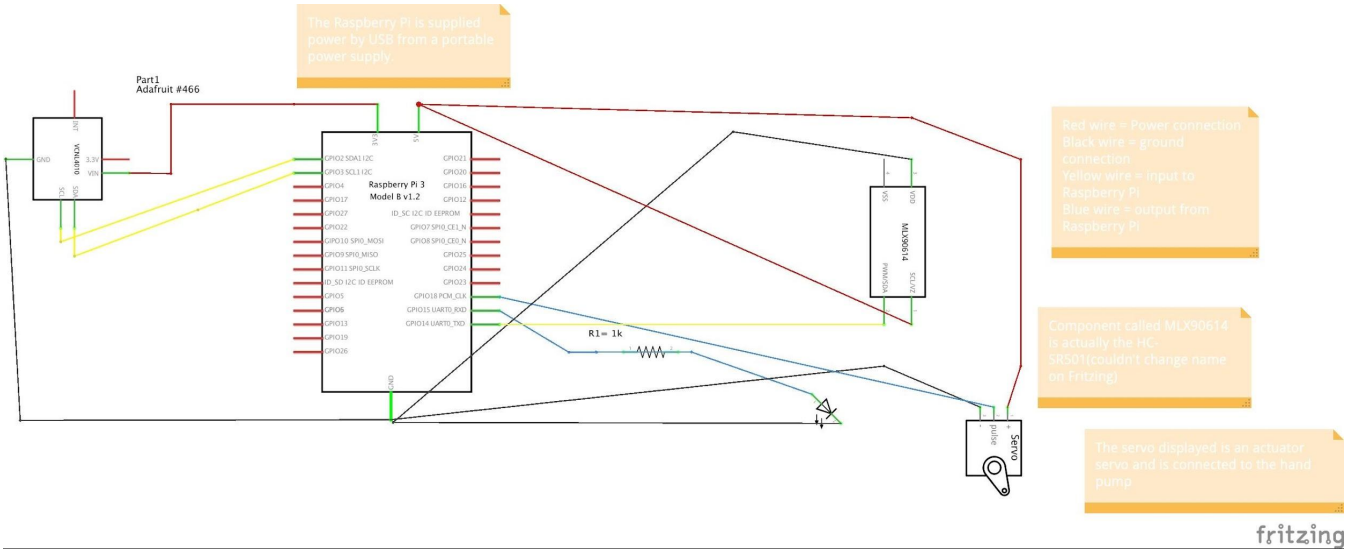


Figure 7.8 Side Inside View With Dimensions

Circuit

The screenshot below is the layout of the circuit created using fritzing. It includes the raspberry pi, all of the sensors, the linear actuator for the dispensing mechanism displayed as a servo as it was all that was available with the software, the power supply and how they all connect together.



Detailed design for network API

Implementation

For the implementation process of the network API design detailed above, a simple API application was developed using Flask and python. All code for this can be accessed through the GitLab link in the appendices at the end of this document. It includes user authentication, and access to all dispenser information via REST network requests. Further description is detailed below.

appl.py

Lines 1-4: In order to start up the Flask application, Flask, Api, JWT and db imports are included so to tell python that we wish to use them throughout our project.
Flask_jwt.JWT: JWT stands for json web token and is an obfuscation of data.
Db.db: this is importing the db variable from the db.py file created for this project.

```
1  from flask import Flask
2  from flask_restful import Api
3  from flask_jwt import JWT
4  from db import db
```

Lines 16: Here an app named ‘app’ is created from the Flask class. ‘__name__’ is a unique python variable and gives each file a unique name. It is used so that Flask knows that the application is running in a specific unique place.
Line 17: Used to tell sqlalchemy where to find the db file. The sqlalchemy database is going to live at the root folder in our project data.db.
Line 18: Specifies a configuration property. It turns off the flask-sqlalchemy modification tracker but does not turn off the sqlalchemy modification tracker which is used in order to know when an object has been changed but not saved to the database.
Line 19: This is the decryption key that the app needs to encrypt and understand what was been encrypted.
Line 20: Here we create the api. ‘Api’ is imported from flask_restful and it allows us to add resources to it with ease.

```
16  app = Flask(__name__)
17  app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///data.db'
18  app.config['SQLAlLCHEMY_TRACK_MODIFICATIONS']=False
19  app.secret_key = 'group18'
20  api = Api(app)
```

Line 22-23: This allows sqlalchemy to create tables in a database. A flask decorator is used (@app.before_first_argument), which will run the create_tables() method below it before the first request to this app.

```
22  @app.before_first_request
23  def create_tables():
24      db.create_all()
```

Line 28-36: This is of all of the endpoints for this application. Each line tells our api that the location resource which we’ve created is now accessible via our api and includes the endpoint which will be called by whoever is using the api.

```
28  api.add_resource(Location, '/location/<string:name>', '/location/<int:id>')
29  api.add_resource(Status, '/status/<string:name>')
30  api.add_resource(Detection, '/detection/<string:name>')
31  api.add_resource(Dispenser, '/dispenser/<string:name>', '/dispenser/<int:id>')
32  api.add_resource(DispenserList, '/dispensers')
33  api.add_resource(LocationList, '/locations')
34  api.add_resource(StatusList, '/statuses')
35  api.add_resource(DetectionList, '/detections')
36  api.add_resource(UserRegister, '/register')
```

Db.py
This file hosts the sqlalchemy object. The sqlalchemy object is imported from flask_sqlalchemy. The variable db is initialised as SQLAlchemy().
The object SQLAlchemy() will link to our flask app and look at all of the objects it is told to look at. It will then allow us to map those objects to rows in a database.

db.py

```
1  from flask_sqlalchemy import SQLAlchemy
2
3  db = SQLAlchemy()
```

Security.py

Used to authenticate a user, given a username and password. It finds the user by username. If the username exists it will check to see if the password is correct and if it is, return the user.

Identity function is unique to flask-jwt and takes in a payload which is the contents of the jwt token. The user_id is extracted from the payload and once we have the user_id, we then can retrieve the specific user matching the payload.

security.py

```
1  from werkzeug.security import safe_str_cmp #compares strings safelfy
2  from models.user import UserModel
3
4  def authenticate(username, password):
5      user = UserModel.find_by_username(username)
6      if user and safe_str_cmp(user.password, password):
7          return user
8
9  def identity(payload):
10     user_id = payload['identity']
11     return UserModel.find_by_id(user_id)
```

User.py – models

Users will be stored in a database and when the user calls the /auth endpoint it will retrieve their username and password from it. The username and password will be compared to the ones sent in the request, checked if they match and if they do, a jwt token will be sent back.

Line 5: We tell sqlalchemy the table name, where we wish our objects to be stored. We then tell it what columns we wish the table to contain.

user.py — models | user.py — resources

```
1  import sqlite3
2  from db import db
3
4  class UserModel(db.Model):
5      __tablename__ = 'users'
6
7      id = db.Column(db.Integer, primary_key=True)
8      username = db.Column(db.String(80))
9      password = db.Column(db.String(80))
10
11
12     def __init__(self, username, password):
13         self.username = username
14         self.password = password
15
16     def save_to_db(self):
17         db.session.add(self)
18         db.session.commit()
19 #store of data
20     @classmethod # using current User class
21     def find_by_username(cls, username):
22         return cls.query.filter_by(username=username).first()
23
24     @classmethod # using current User class
25     def find_by_id(cls, _id):
26         return cls.query.filter_by(id=_id).first()
```

User.py – resources

Line 6: UserRegister(Resource) is a resource so it can be added to the api using flask-restful.

Parsing ensures only some elements can be passed in through the json payload. Flask-restful uses reqparse to do this.

Line 8: initialises a new object which can be used to parse the request.
Lines 9-18: arguments defined for the parser.
Lines 20-29: If data (the username and password) is posted to the register, it checks to see if the username exists using the find_by_username method and if it doesn't it adds the new user and its password to the database.

user.py — models | user.py — resources

```python
import sqlite3
from flask_restful import Resource, reqparse
from models.user import UserModel


class UserRegister(Resource):

    parser = reqparse.RequestParser()
    parser.add_argument('username',
        type=str,
        required=True,
        help="This field can't be blank"
    )
    parser.add_argument('password',
        type=str,
        required=True,
        help="This field can't be blank"
    )

    def post(self):
        data = UserRegister.parser.parse_args()

        if UserModel.find_by_username(data['username']):
            return {"message": "A user with that username already exists"}, 400

        user = UserModel(**data)
        user.save_to_db()

        return {"message": "User created successfully"}, 201
```

Dispenser.py – models

The dispenser model is created with a table called 'dispensers' and its columns consisting of id, name, location_id, staus_id, usage_count, detection_count, detection_id, locations, statuses and detections.

Lines 14-16: back-references which allows a dispenser to see which locations, statuses and detections with their respective ids equal to the dispenser's own id. Tells sqlalchemy that it has a relationship with LocationModel, StatusModel andDetectionModel. Lazy='dynamic' allows locations,statuses,detections to be query builders that have the ability to look into 'locations','statuses' and 'detections' tables accessed in the json method below.

```python
from db import db

class DispenserModel(db.Model):
    __tablename__ = 'dispensers'

    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String(80))
    location_id = db.Column(db.String())
    status_id = db.Column(db.String())
    usage_count = db.Column(db.Integer())
    detection_count = db.Column(db.Integer())
    detection_id = db.Column(db.String())

    locations = db.relationship('LocationModel', lazy='dynamic')
    statuses = db.relationship('StatusModel', lazy='dynamic')
    detections = db.relationship('DetectionModel', lazy='dynamic')
```

Line 27: A json method is used to return a json representation of the model.

```
18      def __init__(self, name, location_id, status_id, usage_count, detection_count, detection_id):
19          self.name = name
20          self.location_id = location_id
21          self.status_id = status_id
22          self.usage_count = usage_count
23          self.detection_count = detection_count
24          self.detection_id = detection_id
25          # self.location_id = location_id
26
27      def json(self):
28          return {'id': self.id, 'disp': self.name, 'locations': [location.json() for location in self.locations.all()],
29          'statuses': [status.json() for status in self.statuses.all()], 'usage_count': self.usage_count, 'detection_count': self.detection_count,
30          'detections': [detection.json() for detection in self.detections.all()]}
```

Line 33-34: find_by_name method which runs a query to the database where name=name.
Line 36-37: Same as above except it looks for id instead of name.
Line 39-45: methods that both add data or delete data to/from the database.

```
32      @classmethod
33      def find_by_name(cls, name):
34          return cls.query.filter_by(name=name).first() #SELECT * FROM items WHERE name=name
35      @classmethod
36      def find_by_id(cls, id):
37          return cls.query.filter_by(id = id).first()
38
39      def save_to_db(self):
40          db.session.add(self)
41          db.session.commit()
42
43      def delete_from_db(self):
44          db.session.delete(self)
45          db.session.commit()
```

Dispenser.py – resources

Parsing ensures only some elements can be passed in through the json payload. Flask-restful uses reqparse to do this. These elements are building, room, start-date and dispenser_id.

Line 6: initialises a new object which can be used to parse the request.
Lines 8-32: arguments defined for the parser.

```
1   from flask_restful import Resource, reqparse
2   from models.dispenser import DispenserModel
3
4
5   class Dispenser(Resource):
6       parser = reqparse.RequestParser()
7
8       parser.add_argument('location_id',
9           type=int,
10          required=True,
11          help="This field can't be left blank!!"
12      )
13      parser.add_argument('status_id',
14          type=int,
15          required=True,
16          help="This field can't be left blank!!"
17      )
18      parser.add_argument('usage_count',
19          type=int,
20          required=True,
21          help="This field can't be left blank!!"
22      )
23      parser.add_argument('detection_count',
24          type=int,
25          required=True,
26          help="This field can't be left blank!!"
27      )
28      parser.add_argument('detection_id',
29          type=int,
30          required=True,
31          help="This field can't be left blank!!"
32      )
```

Line 39-43: The get method which sends a GET request for the dispenser. If the name included in the endpoint is a name in the database, the corresponding dispenser is returned in json format. Otherwise a message is returned that it could not be found along with a 404 error.

Line 45-49: Same as above except the argument id is used instead of name.
Line 51-64: The post method which sends a POST request to add a new dispenser to the database. The required arguments must be entered. Error handling is used when saving the data to the database. If unsuccessful a message displays along with a 500 error signifying an internal server error.
Line 66-71: A delete method to send a DELETE request to the database including the dispenser name. If the name is found in the database, the dispenser is deleted.

```
39      def get(self, name):
40          dispenser = DispenserModel.find_by_name(name)
41          if dispenser:
42              return dispenser.json()
43          return {"message": "Dispenser not found"}, 404
44
45      def get(self, id):
46          dispenser = DispenserModel.find_by_id(id)
47          if dispenser:
48              return dispenser.json()
49          return {"message": "Dispenser not found"}, 404
50
51      def post(self, name):
52          if DispenserModel.find_by_name(name):
53              return {"message": "A dispenser named '{}' already exists".format(name)}, 400
54
55          data = Dispenser.parser.parse_args()
56
57          dispenser = DispenserModel(name, data['location_id'], data['status_id'], data['usage_count'], data['detection_count'],
58              data['detection_id'])
59          try:
60              dispenser.save_to_db()
61          except:
62              return {"message": 'An error occurred while creating the location.'}, 500
63
64          return dispenser.json(), 201
65
66      def delete(self, name):
67          dispenser = DispenserModel.find_by_name(name)
68          if dispenser:
69              dispenser.delete_from_db()
70
71          return {'message': 'Dispenser deleted'}
```

Line 73-75: This method is called when the GET request is sent for all dispensers. It returns all existing dispenser information.

```
73  class DispenserList(Resource):
74      def get(self):
75          return {'dispensers': [dispenser.json() for dispenser in DispenserModel.query.all()]}
```

Location.py – models (Same applies for status.py-models and detection.py-models)
Line 12: The column dispenser_id is created as an integer as it must match exactly the dispenser id type in dispenser.py-models. The foreign key is the id column from the ‘dispensers’ table shown above in dispensers.py-models. The foreign key in location of the dispenser_id creates a link between both dispenser and location.
Line 13: sees that we have a dispenser_id and therefore a dispenser can be found in the database that matches the dispenser_id. Now every location model has a property dispenser that is the dispenser which matches the dispenser_id in its id.

location.py — models × location.py — resources ×

```
1  from db import db
2
3  class LocationModel(db.Model):
4      __tablename__ = 'locations'
5
6      id = db.Column(db.Integer, primary_key=True)
7      name = db.Column(db.String(80))
8      building = db.Column(db.String(80))
9      room = db.Column(db.String(80))
10     start_date = db.Column(db.Integer())
11
12     dispenser_id = db.Column(db.Integer, db.ForeignKey('dispensers.id'))
13     dispenser = db.relationship('DispenserModel')
14
15     def __init__(self, name, building, room, start_date, dispenser_id):
16         self.name = name
17         self.building = building
18         self.room = room
19         self.start_date = start_date
20         self.dispenser_id = dispenser_id
21
22     def json(self):
23         return return {'id': self.id, 'dispenser_name': self.name, 'building': self.building, 'room': self.room,
24         'start_date': self.start_date, 'dispenser_id': self.dispenser_id}
25
26     @classmethod
27     def find_by_name(cls, name):
28         return cls.query.filter_by(name=name).first() #SELECT * FROM items WHERE name=name
29
30     def save_to_db(self):
31         db.session.add(self)
32         db.session.commit()
33
34     def delete_from_db(self):
35         db.session.delete(self)
36         db.session.commit()
```

Location.py – resources (Same applies for status.py-resources and detection.py-resources)
Parsing ensures only some elements can be passed in through the json payload. Flask-restful uses reqparse to do this. These elements are building, room, start-date and dispenser_id.
Line 7: initialises a new object which can be used to parse the request.
Lines 8-27: arguments defined for the parser.

location.py — models × location.py — resources ×

```
1  from flask_restful import Resource, reqparse
2  from flask_jwt import jwt_required
3  from models.location import LocationModel
4
5
6  class Location(Resource):
7      parser = reqparse.RequestParser() #NEW
8      parser.add_argument('building',
9          type=str,
10         required=True,
11         help="This field can't be left blank!!"
12     )
13     parser.add_argument('room',
14         type=str,
15         required=True,
16         help="This field can't be left blank!!"
17     )
18     parser.add_argument('start_date',
19         type=int,
20         required=True,
21         help="This field can't be left blank!!"
22     )
23     parser.add_argument('dispenser_id',
24         type=int,
25         required=True,
26         help="Every location needs a dispenser id!!"
27     )
```

Line 29: This ensures that user authentication is successful before any requests can be made.

Line 30-34: The get method which sends a GET request for the location. If the name included in the endpoint is a name in the database, the corresponding location is returned in json format. Otherwise a message is returned that it could not be found along with a 404 error.
Line 36-49: The post method which sends a POST request to add a new location to the database. The required arguments must be entered. Error handling is used when saving the data to the database. If unsuccessful a message displays along with a 500 error signifying an internal server error.
Line 51-56: A delete method to send a DELETE request to the database including the location name. If the name is found in the database, the location is deleted.

```
29      @jwt_required()
30      def get(self, name):
31          location = LocationModel.find_by_name(name)
32          if location:
33              return location.json()
34          return {'message': 'Location not found'}, 404
35
36      def post(self, name):
37          # if DispenserModel.find_by_name(name):
38          #   return {'message': "A dispenser with name '{}' already exiists".format(name)}, 400
39          # data = Dispenser.parser.parse_args()
40          data = Location.parser.parse_args()
41
42          location = LocationModel(name, data['building'], data['room'], data['start_date'], data['dispenser_id'])
43          #exception is what python runs when an error occurs
44          try:
45              location.save_to_db()
46          except:
47              return {"message": "an error occured inserting the data"}, 500 #internal server erro, somethinf went wrong that not the users fault.
48
49          return location.json(), 201
50
51      def delete(self, name):
52          location = LocationModel.find_by_name(name)
53          if location:
54              location.delete_from_db()
55
56          return {'message': 'Location deleted'}
```

Line 75-77: This method is called when the GET request is sent for all dispensers. It returns all existing dispenser information.

```
75  class LocationList(Resource):
76      def get(self):
77          return {'locations': [location.json() for location in LocationModel.query.all()]}
```

8. Failure Mode & Effects Analysis (FMEA)

Risk Estimation

The Risk Priority Number (RPN) gives a numerical estimate of the risk each failure carries in any design. It is calculated using three values:

- Severity Rating (S)
- Occurrence Rating (O)
- Detection Rating (D)

$$RPN = Severity \times Occurrence \times Detection$$
$$= S \times O \times D$$

$$No\ Risk = 1 \qquad Most\ Risk = 125$$

Scales (1-5)

Severity Rating (S)	Severity	Effect
5	Catastrophic	Product recall
4	Harmful	Permanent damage, irreparable
3	Significant	Fixable damage
2	Minor inconvenience	Complaint from users
1	None	

Detection Rating (D)	Detectability	Description
5	Impossible	Impossible to detect
4	Rare	Undetectable except by specific highly skilled person
3	Possible	Detectable by skilled person
2	Likely	
1	Certain	Detectable by anyone

Occurrence Rating (O)	Likelihood	Description
5	Expected	1/10
4	Likely	1/100
3	Occasional	1/10,000
2	Unlikely	1/100,000
1	Improbable	1/1,000,000

Risk Level	
1-10	Inconvenient, small risk
11-20	Action required
20+	Urgent action required

Figure 8.1: FMEA Scales

FMEA Tables

The below FMEA tables take into consideration the possible hazards that the dispensing device may be subject to in an attempt to reduce, if not eliminate the risk of them occurring. Each table highlights the possible failures relating to each area of the device: mechanical, electronic and software.

MECHANICAL							
Number	Failure	Effect	Cause	Severity	Detection	Occurrence	RPN
1	Blockage in dispensing pump or tubing.	No fluid dispensed	Solidification of excess gel due to evaporation	3	2	3	18
2	Cracked or chipped outer cover	Sharp edge exposed – potential to injure a user	Obstructed by an external object, misuse.	3	1	2	6
3	Kink in tubing	Obstruction to fluid flow – little or no fluid dispensed	Crowding of other components and pump	3	3	2	18
4	Unintentional dismounting of device	Potential to injure somebody and/damage surrounding objects.	Incorrect installation, loose screws.	4	3	2	24
5	Refill leakage	Damaged components (battery, sensors)	Not sealed correctly, improper replacement	5	3	2	30

Figure 8.2: Mechanical FMEA table

ELECTRONIC							
Number	Failure	Effect	Cause	Severity	Detection	Occurrence	RPN
1	Beacon stops working	Users will not notice device	Defective LED, external impact	3	2	3	18
2	Power supply runs out sooner than expected	Device would be inactive until battery is replaced	Unexpected heavy usage of device, old battery	4	1	3	12

Figure 8.3: Electronic FMEA table

SOFTWARE							
		Rating					
Number	Failure	Effect	Cause	Severity	Detection	Occurrence	RPN
1	Raspberry pi cannot communicate with API	Cannot add data to database or change dispenser volume	Failed HTTP request	4	3	3	36
2	API cannot add data to database	Missing data	Invalid data or failed API	3	3	2	18
3	Cannot access database for data analysis	User cannot analyse data	Failed API request or invalid SQL query	4	1	2	8

Figure 8.4: Software FMEA table

9. Conclusions

To conclude, this design portfolio has discussed the development of an intelligent automated dispenser for hand sanitising purposes. The objectives of this project were to design a physical dispensing device that is either free standing or wall mounted, incorporate a Raspberry Pi to control sensors in the dispenser and to design and develop a network API to record usage statistics that can be used for data analysis. Throughout this document these objectives were discussed and solutions to these design questions were provided.

The design for the dispenser was chosen based on a number of factors such as effectiveness and reliability. The final design was chosen based on these chosen factors. Electrical components such as the Raspberry Pi and other sensors were chosen based on things like their price, reliability and compatibility. All chosen components were deemed to be the best for the design. For the network API a REST API was chosen. It was developed

using Python and Flask using a SQLAlchemy database. Technical drawings and source code for this can be found below in the Appendices.

10. Appendices

10.1 Technical drawings

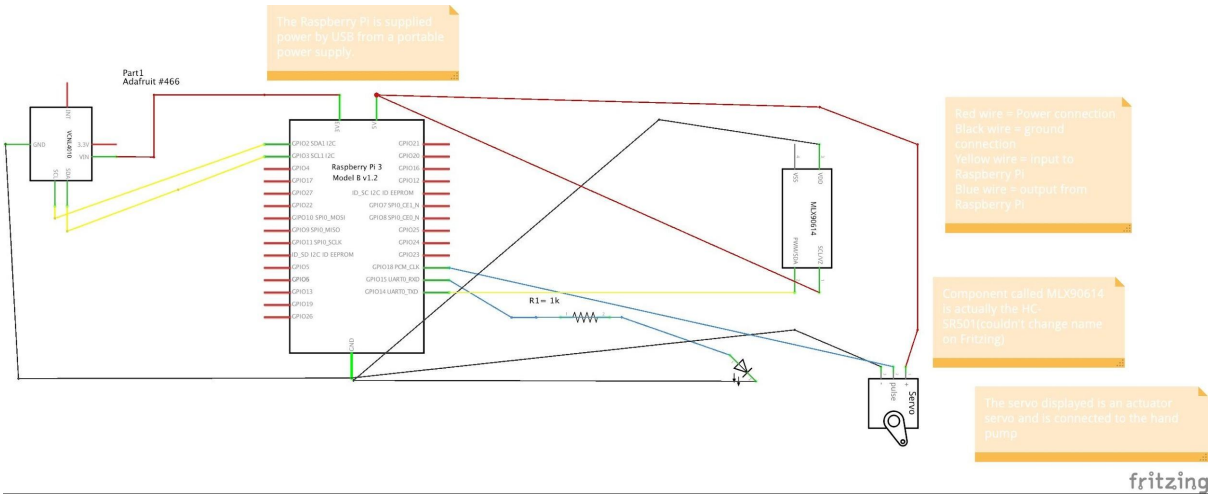
Mechanical

Technical diagrams can be found at the start of section 7.

Software

Sequence diagrams can be found in section 5.Selection of optimum component solutions.
Network basic class diagram can be found in section 5.Selection of optimum component solutions.
System architecture diagram can be found in section 6. Functional Specification - 4. System Architecture.
Top-level context diagram can be found in section 6. Functional Specification - 5. High-level design.
DFD can be found in section 6. Functional Specification - 5. High-level design.

10.2 Electronic circuit drawings



10.3 Software code

Source code for the network API can be found on Gitlab through the following link https://gitlab.computing.dcu.ie/leaveyr2/alternative_intra .

10.4 bill of materials

Other materials needed apart from the components are the ABS needed for the outer shell and the wall mount and the hand sanitizer fluid that will go inside the unit.
The ABS can be sourced from a local supplier based on where the product is being manufactured. However to look at the pricing of the material a website was found that sells ABS granules in the desired color for our hand sanitizer dispenser for a good price.[30]

10.5 component list and suppliers

- Linear actuator - Powerlift actuator servo available on amazon31]
- Soap pump available at parfum dreams[32]
- Speed controller available from robotshop.com[33]
- Dispensing sensor available from the company website adafruit.com[34]
- Alert sensor is sold on wish[35]
- Power source is the kumann Lithium battery pack available on amazon[36]
- The raspberry pi is sold from links off the raspberry pi website[37]
- Small lock sold on AliExpress.com[38]

10.6 Costings

Component Pricing

Below is a list of all the components used in the hand sanitizer and the pricing.

- Linear actuator - Powerlift actuator servo : €8.75
- Soap pump - €0.40
- Speed controller - €14.67 or 100 for €12.96 each
- Dispensing sensor - VCNL4010 - €3
- Alert sensor - €1
- Power source - Kumann Lithium Battery pack - €20.99
- Raspberry pi - €32.34 or €28.16 each if 150+ bought
- ABS costs approximately €1.50 per 0.5kg.
- Small lock for the front and top of the hand sanitizer unit €1.34 with 10% discount on the purchase of 50 locks or more.

AWS Server Pricing

To host the API and database using AWS, Amazon RDS is needed for a database, API Gateway is used to host the REST API and AWS Lambda is used to execute queries to the database. Below is a breakdown of estimated monthly expenses that we got using the AWS pricing website.

- Amazon RDS for MySQL - €9.54 per month (db.t3.micro instance with 10 GB storage)
- AWS Lambda - €0.97 per month (with 30,000 requests per month)
- Amazon API Gateway - €3.10 per million requests

The total estimated price per month to host the server with a database and API is €13.64. To cover development and maintenance costs, we would charge customers €15 per month for a subscription. This subscription price is subject to change depending on whether or not a customer uses more server space.

References

[1] https://safety.lovetoknow.com/Hand_Sanitizer_Fire_Hazard

[2]https://www.insider.com/does-hand-sanitizer-expire

[3] https://www.fishersci.ie/shop/products/micronova-touch-free-dispenser-soaps-4/p-100333

[4] https://renergise.ie/shop/soap-dispenser/automatic-hand-sanitizer-dispenser-1-100ml/
[5] https://touchland.com/products/blue-kub-basic-package

[6] https://www.hse.gov.uk/news/hand-sanitiser-manufacture-supply-coronavirus.htm
[7]"PURELL® TFX™ Touch-Free Dispenser 1200mL, white", *Purell.eu*, 2020. [Online]. Available: https://www.purell.eu/en/purell-tfx-touch-free-dispenser-1200ml-white. [Accessed: 14- May- 2020].

[8]"PURELL® LTX-7™ Touch Free Dispenser 700mL, chrome/black", Purell.eu, 2020. [Online]. Available: https://www.purell.eu/en/purell-ltx-7-touch-free-dispenser-black. [Accessed: 14- May- 2020].

[9]*Tork.ie*, 2020. [Online]. Available: https://www.tork.ie/product/561600/dispenser/hand-sanitiser. [Accessed: 14- May- 2020].

[10]*Tork.ie*, 2020. [Online]. Available: https://www.tork.ie/product/460009/dispenser/hand-sanitiser. [Accessed: 14- May- 2020].

[11]Az745204.vo.msecnd.net. 2020. *Tork Easycube User Manual*. [online] Available at: <https://az745204.vo.msecnd.net/docs-c5/203/189203/original/466200-image-design-easycube-foam-skincare-automatic-dispenser-manual.pdf> [Accessed 15 May 2020].

[12]Hermann, C., 2016. *Systems For Monitoring Hand Sanitization*. US10032359B2.

[13] R. Sinha, S. and Kennedy, C., 2008. *Sanitation Tracking And Alerting System*. US20090224907A1.

[14] Chuang, S., Sahoo, N., Lin, H. and Chang, Y., 2019. Predictive Maintenance with Sensor Data Analytics on a Raspberry Pi-Based Experimental Platform. *Sensors*, 19(18), p.3884.

[15] Baslyman, M., Rezaee, R., Amyot, D., Mouttham, A., Chreyh, R., Geiger, G., Stewart, A. and Sader, S., 2015. Real-time and location-based hand hygiene monitoring and notification: proof-of-concept system and experimentation. *Personal and Ubiquitous Computing*, 19(3-4), pp.667-688.

[16] *Alibaba.com*, 2020. [Online]. Available: https://www.alibaba.com/product-detail/New-Arrival-2000ml-Hospital-stainless-steel_62536647130.html?spm=a2700.7735675.normalList.1.59a77e07P5u1Zg. [Accessed: 15- May- 2020]

[17]"Cleanflow 800ml Automatic Liquid Soap, Disinfectant Solutions* and Hydro Alcohol Gel* Dispenser", *Intelligenthanddryers.ie*, 2020. [Online]. Available: https://www.intelligenthanddryers.ie/products/vivo-800ml-automatic-liquid-soap-disinfectant-solutions-and-hydro-alcohol-gel-dispenser. [Accessed: 15- May- 2020]

[18]*Amazon.com*, 2020. [Online]. Available: https://www.amazon.com/TESECU-Automatic-Dispenser-Countertop-Waterproof/dp/B083DVCG4B/ref=sr_1_7?dchild=1&keywords=automatic+hand+sanitizer+dispenser&qid=1589535328&sr=8-7. [Accessed: 15- May- 2020]

[19] *Alibaba.com*, 2020. [Online]. Available: https://www.alibaba.com/product-detail/Auto-Soup-Dispenser-Sanitizer-Hand-Gel_62580941147.html?spm=a2700.7735675.normalList.9.59a77e07P5u1Zg&s=p. [Accessed: 15- May- 2020]

[20] "Spray Quick Release Wall Mount/floor Stand Non Connect Hand Soap Bottle Automatic Dispenser - Buy Soap Automatic Dispenser,Sensor Hand Soap Dispenser,Soap Bottle Dispense Product on Alibaba.com", *www.alibaba.com*, 2020. [Online]. Available: https://www.alibaba.com/product-detail/spray-Quick-release-wall-mount-floor_1600051277465.html?spm=a2700.7735675.normalList.32.59a77e07P5u1Zg&s=p&bypass=true. [Accessed: 15- May- 2020]

[21]*Amazon.com*, 2020. [Online]. Available: https://www.amazon.com/TESECU-Automatic-Dispenser-Countertop-Waterproof/dp/B083DVCG4B/ref=sr_1_7?dchild=1&keywords=automatic+hand+sanitizer+dispenser&qid=1589535328&sr=8-7. [Accessed: 15- May- 2020]

[22]2020. [Online]. Available: https://www.youtube.com/watch?v=YwKnYKq9gkU. [Accessed: 15- May- 2020]

[23]2020. [Online]. Available: https://www.youtube.com/watch?v=sG4XUtuzWlw. [Accessed: 15- May- 2020]

[24]G. Kampf, S. Ruselack, S. Eggerstedt, N. Nowak and M. Bashir, "Less and less–influence of volume on hand coverage and bactericidal efficacy in hand disinfection", *BMC Infectious Diseases*, vol. 13, no. 1, 2013. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3851816/.

[25] https://docs.microsoft.com/en-us/azure/architecture/best-practices/api-design

[26] Chambers, J., n.d. *Graphical Methods For Data Analysis*.

[27] Heumann, C., Schomaker, M. and Shalabh., 2017. *Introduction To Statistics And Data Analysis*. Cham: Springer International Publishing AG.

[28] Fitbit.com. 2020. *Fitbit Charge 4 Advanced Fitness Tracker*. [online] Available at: <https://www.fitbit.com/ie/charge4?utm_source=&utm_medium=paidsearch&gclid=EAIaIQobChMIxNP2jOi16QIVA-3tCh1oygKdEAAYASAAEgJIGPD_BwE&gclsrc=aw.ds#> [Accessed 15 May 2020].

[29] Holland, C., Levis, J., Nuggehalli, R., Santilli, B. and Winters, J., 2017. UPS Optimizes Delivery Routes. *Interfaces*, 47(1), pp.8-23.

[30]https://www.alibaba.com/product-detail/Injection-Molding-Grade-High-Flow-Abs_62342222459.html?spm=a2700.7735675.normalList.1.4fc5326eLn2Zuq&s=p

[31]https://www.amazon.co.uk/Actuator-Digital-Loading-Ultra-Micro-Aircraft/dp/B07VSNSHB4/ref=asc_df_B07VSNSHB4/?tag=&linkCode=df0&hvadid=441702772780&hvpos=&hvnetw=g&hvrand=7146224882792749256&hvpone=&hvptwo=&hvqmt=&hvdev=m&hvdvcmdl=&hvlocint=&hvlocphy=1007850&hvtargid=pla-852254885904&psc=1&ref=&adgrpid=99265389261

[32]https://www.parfumdreams.ie/Wella/SP-Accessoires/Accessories/Pumpspender-500-ml/index_50117.aspx?AffId=GA_IE2_PLA_Wella&wt_mc=adwords_nonbrand-IE.cpc.pla-IE&gclid=Cj0KCQjwudb3BRC9ARIsAEa-vUsgb1lesDyWhXJRYWkHY9WNKtKd-Bdu_BUwnBFb1w9a9VEaQFSZl_saAsTkEALw_wcB

[33]https://www.robotshop.com/eu/en/dc-motor-driver-module.html?utm_source=google&utm_medium=surfaces&utm_campaign=surfaces_across_google_iren&gclid=CjwKCAjwltH3BRB6EiwAhj0IUKoltvu1ZGAr5tpMTVqr8o_L-5WJ8Q3ulFvkUJlnfeYVdttFfyODFxoCk0YQAvD_BwE

[34] https://www.adafruit.com/product/466

[35]https://www.wish.com/product/5d737d41c784cc4b8e064510?hide_login_modal=true&from_ad=goog_shopping&_display_country_code=IE&_force_currency_code=EUR&pid=googleadwords_int&c=%7BcampaignId%7D&ad_cid=5d737d41c784cc4b8e064510&ad_cc=IE&ad_curr=EUR&ad_price=2.00&campaign_id=9527731323&gclid=Cj0KCQjwudb3BRC9ARIsAEa-vUv5D1FtKo2IBGPQ4bj3D6lhycsG4iWHFNHX7VcnY3FVw4770hx48M0aAgmYEALw_wcB&share=web

[36]https://www.amazon.co.uk/Kuman-Lithium-Battery-Expansion-Raspberry/dp/B06W9FWDSP/ref=asc_df_B06W9FWDSP/?tag=googshopuk-21&linkCode=df0&hvadid=427931137390&hvpos=&hvnetw=g&hvrand=11215809391162035657&hvpone=&hvptwo=&hvqmt=&hvdev=c&hvdvcmdl=&hvlocint=&hvlocphy=1007850&hvtargid=pla-331206144889&psc=1&tag=&ref=&adgrpid=99265830443&hvpone=&hvptwo=&hvadid=427931137390&hvpos=&hvnetw=g&hvrand=11215809391162035657&hvqmt=&hvdev=c&hvdvcmdl=&hvlocint=&hvlocphy=1007850&hvtargid=pla-331206144889

[37]https://cpc.farnell.com/raspberry-pi/raspberrypi3-modb-1gb/sbc-raspberry-pi-model-b-1gb-cn/dp/SC14013

[38]https://www.aliexpress.com/item/32316907046.html