# Product: DeliverED Home

**Team: DeliverED**


## Abstract

Our product aims to eliminate the need for waiting at home for a delivery – we will provide a secure and convenient personal mailbox for contact-less deliveries, controlled by an Android app. In this demo, we have made huge advances in terms of front-end app development, and Webots simulation. We have created an initial app render, with functional navigation and all pages displayed. We also have a Webots simulation showing the physical movement of doors and internal floors in our device. We have also created a Website, with a completed home page and placeholders for the other pages, as well as a plan of the contents of each page. For hardware, we have have an Arduino and a door locking mechanism in Appleton Tower which we are able to interact with.


## 1. Project plan update

Our Demo 2 goals are as follows:

- Design Website UI - Achieved
- Webots Simulation (Physical Factors - dividing floor, elevator, doors) - Achieved
- Webots Simulation (Cameras & Weight Sensors) - Partially Achieved
- Presentable App Front-End - Achieved
- Hardware (EM door for secure compartment, using sensors to determine state) - Achieved
- Hardware (forklift, photon) - Partially Achieved
- Market Research (ethics approval) - Achieved

In terms of the app, we are significantly ahead at this stage – we have a functioning front-end with navigation, and some back-end connectivity, which was a demo 3 goal. In terms of hardware, we have been working with a modular approach due to the restrictions. We have connectivity with a Photon, and have also been working with Garry, who has a physical door controlled by an Arduino, which we are working on controlling the locking mechanisms of. The photon and forklift have been commissioned at this stage. In terms of sensors on the Webots simulation, we decided that using cameras raised too many ethical issues, weight sensors were unnecessary, and didn't get the chance to look into infrared sensors.

The front-end app and design team was made up of Amy, Chris and Huacheng. Amy and Huacheng created the render in Android studio using our App UI designs from the previous demo. Chris was in contact with experts concerning accessibility, and general UI improvements, which led to a list of UI alterations we plan to act on in demo 3. Amy and Chris also created the website. Neo created the simulation in Webots, showcasing the external doors and internal floors moving. Hrichika headlined the conversations with Garry concerning the Arduino and door locking mechanisms, overseeing the hardware aspects, and scheduling meetings for the rest of the group. The app backend was covered by Hallelujah, Yizhuo and Harry, with Harry specifically working on the Photon he owns, and Hallelujah working on connectivity and data ethics.

During this demo we have continued managing tasks using Trello, and have set up GitHub repos for the individual sections of the project, like the app and the website. We have kept in constant contact through Teams. The software and hardware teams respectively kept in contact almost daily, with the two teams coming together for a general update once a week. Lots of communication with experts has occurred, with huge focus on hardware, accessibility, and data ethics. In terms of budget, we have decided that our product must include an installation cost, as it must be securely attached to a wall or the floor to ensure security.

Our demo 3 goals initially were backend app connectivity, a more detailed Webots simulation, an updated website, and progress with hardware. The only alteration that needs to be made for demo 3 is with regards to the app. Setting up backend connectivity is something that we have already done. We wish to add a new goal, which is to alter our App UI with regards to the accessibility advice we received. These alterations are explained in the UI section of this report.

## 2. Technical details

### 2.1. Webots Simulation

During phase two, Neo has implemented all th basic functionalities of DeliverED in Webots simulation. In accord with the previous demo report, our current simulation demonstrate the general use case as the following.

1. The upper chamber is opened, item to be delivered is placed on the one-way-door platform (red). Door is then locked to ensure security.
2. Moving platform (blue) rises up until an IR break beam is triggered, signalling that there is no more room to go up.

3. One-way-door drops item a small distance onto the next platform (blue) which gently handles the item for UV sanitation from the backside of one-way-door.
4. The platform (blue) lowers itself to the bottom of the device.
5. The lower chamber is opened only with authentication, customer can now collect items.

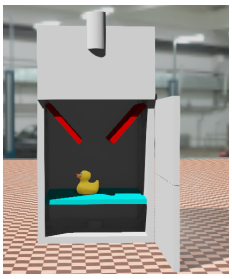

*Figure 1.* Webots Simulation

In this demo, 5 movable parts are rendered, including 3 doors and 2 platforms, which are driven by 5 HingeJoints and 1 SliderJoint. Doors of the cabinet is driven by HingeJoint motors which are attached to the side of the cabinet. The one-way-door platform (red) can be viewed as 2 separate parts but they are moved simultaneously by the HingeJoints which attached to the inner part of the cabinet. The lifting platform (blue) is controlled by a SliderJoint. In order to better associate the simulation with actual hardware implementation, C is chosen to write controllers.

However, there are still some functionalities has not been added into the simulation. Firstly, the magnetic lock which only releases upon authentication is not implemented in webots because there is no such component we could find in the software. Thus, it is completed by our hardware team who used facilities inside Appleton tower. This part is covered in the hardware section. Also the UV light implementation is trivial at this stage, which would be implemented after the physical model is available to us.

### 2.2. Hardware

Since Demo 1, the hardware team have made a lot of progress on the design and implementation behind the mailbox, including advances in communication with the app. The biggest change we made was switching to a mixture of real hardware and simulation. We kept in close contact with Garry for technical advice on the design of the hardware as well as commissions for the creation of individual parts of the device. We are employing a modular approach to our hardware currently, with plans to bring everything together at a later date.

Currently, our hardware exists in three forms – in Appleton we have a commissioned door lock to be used in the construction of the final device, we have simulation to demonstrate how this device would work when put together, and a development/test Particle Photon to experiment with app connectivity. The Photon is a cloud-connected IoT microcontroller that will act as the heart of the device. Bringing these together will create our system, and since the Photon and the Arduino currently being used in Appleton run almost identical code, it will be a seamless fusion. The secure door lock is operated by a solenoid controlled via a signal from our microcontroller. In the event of a power failure, there is a backup key lock for customers to use instead of the app. The delivery section's door is closed and locked during the delivery process in the same way.

Initially we planned on using cameras – one on the outside for security and one inside the delivery section to scan the parcel's dimensions. We decided to remove both of these. The first could be a privacy risk, and there are other products for this purpose. The second is unnecessary overcomplication, and instead we now use an IR break beam on the inside of the secure section, placed such that when it is broken, going any further would interrupt the operation of the door above.

Finally, after conversation with the technicians, we decided to utilise a forklift instead of the previous ideas of a scissor or motor lift. This was mainly because it will allow us to control it with only one motor at the back, and stability issues can be counteracted by placing grooves down the far end of the side walls, potentially with small notches, so that the forks have something on the far side to avoid bending under heavy weights. For future demos, we plan on fully realising app-device communication by ordering in a Photon to Appleton, which will begin the process of bringing our modules together. We have also commissioned the forklift (and will eventually add UV sanitation) to be built, which may be a long process but will lead to our device being one step closer to complete.

### 2.3. User interface

For this demo, we implemented the first revision of our UI in Android Studio. This is discussed in the Software section below. Concurrently, we had a meeting with Ryan Bowler for feedback to improve the UI and UX. We have now started our revision of the design for Demo 3.

Ryan's comments on our design overall were that we have everything needed but that there was a lot of room to simplify and areas to improve inclusivity. His guidance was to aim to be inclusive in our design from the start; we shouldn't design something first and then try to accommodate for people later. Here were his notes:

General UI improvements:

- Reduce the number of screens on the app. There are ways to fit more functionality on each screen. For a lay user, the core functions of the app should be accessible on one or maybe two screens, with more complex tasks accessible through deeper navigation.
- Sign up screen needs to be more unified, e.g. the logo and slogan should be together but separate from the sign in function.

- Device manager should provide core functions such as unlocking/locking the door to the user's delivery. Another note was that custom names of devices would aid in easily identifying different devices.
- The option of adding a passcode for device functions would help reduce accidental actions, e.g. leaving the door unlocked.

Visual inclusivity:

- Modify our blue text on blue backgrounds - they must contrast well tone and colour wise. The 2 shades of blue were quite hard to distinguish between, especially if small. Also, it would be very difficult for those who have tritanopia or tritanomaly colour blindness (blue-yellow colour blindness) to read.
- Icons need to be improved. Some of the icons could be more informative and we should be able to convey the message by just the image without the need of colour.
- There is a lot of whitespace, with this we could increase font size and/or size of elements such as buttons or add functionality.

Physical inclusivity:

- Reduce the actions needed to perform tasks. We could use a QR code to register a device and this would aid those who find typing difficult.

For the website, we have completed the landing page and have set up spaces to fill in the contents of each of the sections. Future plans for the website front-end are to begin the mobile view and - once we have a section's content - design its layout and implement it. See our group space here to view our website.

## 2.4. Software

### 2.4.1. Version Control(GitHub)

We use GitHub repository for version control. The GitHub repository of a certain teammate is taken as the original repository. Android Studio provides convenient version control tools so that we can directly commit changed files to our own repository. The software team have to make sure all local files are up-to-date before they start to working. We use pull requests to evaluate code as a team before it is pushed to the master branch.

### 2.4.2. Device Control System

At this stage we aimed to achieve connectivity between the application(the Device controller class in particular) and a Photon device through the particle cloud. We have had to migrate the project to the latest support libraries(androidx) as it was a requirement and generate OAuth Client(authentication client ID) and a client secret to identify users and requests from our application(DeliverED). We had not anticipated that a significant portion of the project work would have to be completed to fully test the authentication process in connecting to the particle cloud, but as mentioned in the last report the security and automation advantages to be gained from using the particle cloud far outweigh the time that it will take to complete it as the cloud encapsulates common methods and components with many functions also realized without writing code, which greatly improves the development efficiency of networking modules.

We have, for testing purposes, used POST requests with authentication headers to connect to the particle cloud and Photon hardware to send requests to a function LED which takes Strings "on" and "off" as arguments and tests initial connectivity. The major obstacle with this task was that network calls have to run on a different thread than the main one. Therefore, the function being used creates a new thread in the process that executes the network calls. The synchronization method of photon enables our app to do this more efficiently in addition to general message forwarding moreover, Photon provides a load balancing function. Through simple configuration, a master service can also be connected to multiple other devices Run our DeliverED service hence, we have observed the advantages of the particle cloud as opposed to using general methods to connect to the hardware.

### 2.4.3. Notification and Alert System

The mailbox will send out a message to application after a successful action. This message consists of an unique ID and action code. The unique ID is used to identify a particular mailbox. The notifications will always indicate the name of mailbox according to ID. Every action code have one and only one corresponding hardware functionality. There is an internal reference document in application. The document stores action code, the corresponding functionality and functionality description. This document have been agreed by hardware engineers. After receiving the responds from mailbox, the notification system will split the message to ID and action code. A template function will be called. It will search the reference document and find the description by the given action code. Then the description will be filled into the content part of notification. Template function will also visit Multi-Device system, which is the subsystem of Device Control system, to get the name by the given unique ID. The name of function will appear at the right-bottom corner of the notification. This progression is shown in Figure 3.

The notification system depends on the connection between mailbox and application. Since the connection might be unstable, we decide to use a timer to ensure the any request can get a respond from mailbox. After device control sending request, it will call the timer of notification function. The notification system will start to wait a responds for a period. If there is no responds from hardware, notification system will tell device control system to send the same request again but labelled 'confirmation'. The mailbox should have an ability to distinguish an confirmation request from a general request. The confirmation request can only be started by notification system. The mailbox will check whether an command has been conducted after receiving an confirmation request.

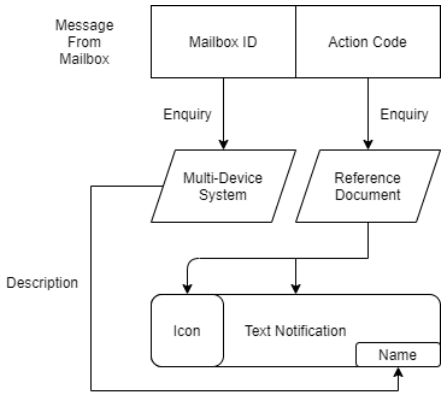


*Figure 2.* Notification System

## 3. Evaluation

### 3.1. Hardware Testing

With respect to the the electromagnetic lock, we are able to operate it from our microcontroller, as has been discussed in the hardware section. We are also able to read sensor values to check if our door latch is pulled in (=1) or pushed out (=0), or if our solenoid off (=0) or on (=1) and if it is pulled in (=1) or pushed out (=0). We plan on using these later for testing noise values and representing the system state in the firmware.

### 3.2. Software Testing

#### 3.2.1. Testing Strategy

We decided a 4-stage testing strategy: developer testing, integration testing, system testing and acceptance testing. At the current stage, developer testing is the main method. Developers test basic functions by checking whether an expected output is generated. After the development of all basic functions, we will start an integration testing. This testing concentrates on the combinations of segments of application. Concurrent testing, as a part of integration testing, is the most important. Every mailbox will be regarded as a module. We will emulate several modules sending request and response at the same time. We are expecting the application can work with several devices simultaneously. The notification system and device control system are main parts of a device module. They will be tested in sequence. Request will be generated by device control system and sent out to many devices. The notification system is expected to handle responses from different devices. The last 2 testing stages are combined together. We will test the application as a product. Non-functional testing like stress testing will also be conducted in this stage. The robustness of application is the most important. Along with system testing, we will also invite potential users to try our product and collect possible advice to improve our product.

#### 3.2.2. Developer Testing

Developer testing is based on the finished functionalities. The development of basic functionalities is expected to finished in Demo 3. Therefore, we can only provide part of unit testing. An typical success of a function in navigation bar will turn to the correct page. The expected output of alert page are tabs showing the given message. Part of developer testing result has be shown in Table 1.

| Class | Function | Success | Errors |
|---|---|---|---|
| Welcome Page | Sign In | 5 | 0 |
| Navigation Bar | Device | 5 | 0 |
| | Alert | 5 | 0 |
| | Account | 5 | 0 |
| Alert Page | 'Full' | 5 | 0 |
| | 'Received' | 5 | 0 |
| | 'Low Battery' | 5 | 0 |
| | 'Move' | 5 | 0 |
| Account Page | Log Out | 5 | 0 |

*Table 1.* Developer Testing Result

| Items | ♯ | Cost(£) |
|---|---|---|
| Metal Work [Lock] | 1 | 10.00 |
| Standard Deadbolt Lock | 1 | 20.00 |
| Solenoid | 1 | 25.00 |
| Electronics And Wiring [Lock] | - | 5.00 |
| Motors [Fork Lift] | - | 2.50 |
| Lego [Fork Lift] | - | 15.00 |
| Miscellaneous [Fork Lift] | - | 10.00 |
| Basic Sensors | - | 10.00 |
| Advanced Sensors | - | 30.00 |
| Motor And Encoder Boards | 1 | 10.00 |
| Steel Plates | 6 | 45.00 |
| UV Sterilizer LED | 2 | 18.30 |
| Door Spindle | 1 | 8.40 |
| Magnetic Lock | 1 | 7.00 |
| 6mm Springs | 4 | 12.00 |
| Particle Photon | 1 | 14.34 |

*Table 2.* Budget

## 4. Budget

The budget is explained in Table 2. The estimated cost maybe subject to change as project proceeds. The prices are acquired from the market or from hardware experts. We have avoided providing a total for our budget spending as not all of it has yet been actually spent. A lot of our product is still being built and hence their is some ambiguity as to the exact costs that will be incurred. We expect that the actual spending will be recorded by Demo 3, when we have most of our complete parts delivered.

## 5. Video

Watch our Demo video here (SharePoint).