Astreea Product Proposal: Fill-Level Sensors and 3D Product Models

This document assumes that Astreea will develop and market a line of dispenser products in which are embedded sensors which measure the level of soap/sanitizer in each dispenser, so as to track usage and flag when the device should to be refilled. The specific technology LTS would implement depends on the supplier of the sensors used, but we can describe a typical setup based on industry standards. E-Cube labs, for instance, deploy fill-level monitors (for both solids and liquids) as part of their "CleanCityNetwork" (**CCN**). A German organization, the Fraunhofer Institute for Integated Circuits, has similarly deployed its so-called "s-net" wireless sensor network in a "washroom information service" which, among other fixture sensors, tracks soap fill levels and also each occasion when a soap dispenser's "pump" is activated. The sensor data is packaged via s-net and transmitted from each building using this system to servers hosted by a company called CWS-Boco. In general, the sensor providers also maintain a central web service, such as **CCN**, which integrates data from all buildings/locations into one data access point. The software stack directly managed by Astreea, then, would need to interface with this central data source.

E-Cube's sensor network is a good example of a typical **IoT** data stack. All sensor data is aggregated onto E-Cube's **CCN** servers, which maintain web and mobile applications allowing E-Cube customers to visualize data for each sensor on the network. E-Cube also provides an **API** allowing customers to obtain raw data, rather than relying exclusively on E-Cube's own software. Using the **API** affords companies greater flexibility, because they can use the raw data however they see fit, but it requires companies to implement their own software. One benefit of using custom-built software, rather than the generic web service provided by **IoT** providers, is that companies can benefit from a self-contained application exclusively focused on managing sensor data, which can be fully customized for the company's needs.

It is quite common for **IoT** manufacturers to make sensor data available to customers only via web sites and mobile apps, both of which offer limited functionality. It is usually possible, however, to access the data in machine-readable form via **API**s or some other low-level mechanism, so that companies can design their own **IoT** applications. This can be a worthwhile investment first because of convenience: the software for managing sensor data can be installed directly on computers in the company's offices (or employees' homes), instead of residing solely on employee's personal devices (in contrast to smart-phone access modes). Likewise, the custom applications can be entirely organized around the tasks having to be performed with respect to sensor data: there is no need for employees to browse to a specific website to obtain data, nor could then browse the web in general while using the custom application. In addition to this basic convenience, customized desktop applications also offer superior User Experience: they are not constrained by small screen size and limited touch-screen interactions as with mobile apps, nor are they constrained by the limited screen layouts and interaction modes of web sites (where most user actions need to be implemented by clicking hyperlinks). Customized desktop applications, by contrast, can employ the full range of **GUI** features associated with native software, such as context menus, dialog boxes, multi-window displays, and dedicated windows for **2D** and **3D** data visualization. Moreover, by acquiring raw sensor data, companies can perform their own

market analytics customer metrics, rather than relying exclusively on the analytic information shared by sensor manufacturers. In the context of sanitizer dispensers, Astreea could track data to indicate which dispensers are being used the most/least; which customers are delayed in refilling empty dispensers; which customers are refilling with Astreea's own sanitizer or with some other product; and so forth.

Assuming Astreea were to use E-Cube sensors or some similar product, then, LTS could supply the third-party software which Astreea would need to access the E-Cube data via custom desktop applications. This would involve, first, constructing a "data profile" based on the distinct **CCN API** requests, which are grouped into 16 separate categories (the ClearCityNetworks **API** documentation is at https://doc.cleancitynetworks.com/partner-api-doc/). LTS would then implement all the logic for obtaining data from the **CCN API**, translating **CCN**'s **JSON** results into runtime objects instantiating data types defined in accordance with the **CCN** data profile. Next, LTS would translate these data types into **GUI** components designed and specified by Astreea — tables, charts, network diagrams, scatter plots, or any other textual or graphical designs which Astreea chooses in order to render sensor data in a useful manner, considering how Astreea seeks to use this data. Finally, LTS would implement a database engine to store all sensor data for long-term tracking and analysis. We would recommend for this engine a customized version of our own database technology, **ConceptsDB**, which is a hypergraph database specifically designed for custom desktop applications. In this context, the principal distinguishing feature of **ConceptsDB** is that all **GUI** information is stored directly in the database — that is, all information related to **GUI** design, layout, user actions, user personalization, and so forth — which makes it easy to integrate **GUI** front ends with database back ends.

LTS could implement the software as described above in stages, giving Astreea an opportunity to review and fine-tune each stage before committing to further development. The distinct stages might be organized as follows: (1) **API** access; (2) **GUI** components; (3) database persistence; (4) long-term tracking and market analytics based on database records.

3D Graphics

The use of **3D** graphics could be a complement to the above software in that each dispenser model supplying data to the sensor network would have a **3D** model that could be viewed as part of the sensor data management application. In addition, these **3D** models could be used to enhance the brochures and presentations available for potential customers.

An interactive Astreea brochure could be developed in one of two ways, depending on whether potential customers want to download additional software. The most sophisticated option is to offer a customized PDF viewer, which customers could use to browse the current brochure as well as specialized, interactive brochures which Astreea might consider developing in the future. These brochures would be specifically designed to help customers design special versions of the Astreea products by decorating them with their logos and designs, or customizing the products' shapes and dimensions.

We can also assume that **3D** models of the Astreea products exist or could be developed, because data on design customization has to be transmitted to the factory where nonstandard versions of the products are manufactured. We assume, then, that **3D** models of the Astreea product line can be deployed in formats like **STL**, **OBJ**, **PLY**, or **X3S**, and that **3D** modeling is used in the Astreea offices to create representations of non-standard versions of Astreea offerings. Presumably the graphics files would be similar to those visible here: https://www.turbosquid.com/3d-models/sanitizer-aibolit-2000-model-1561963 — to give an example of

a floor-standing dispensers roughly similar to Astreea's (notice particularly the fifth image on the scroll, which does a good job of showing the mesh geometry). For sake of discussion, assume that Astreea salespeople can use a version of MeshLab (an open-source 3D graphics engine) to design custom versions of the products. Assume, moreover, that some customers use a special PDF viewer that we can provide which includes the ability to view 3D graphics in a separate window – whereas the "static" brochure just shows photos of sanitizer dispensers, say, an interactive version could let potential customers visualize the products in 3D.

The next step, then, is to hook up the customer's PDF viewer with the salespersons' MeshLab application. Then, if the salesperson is on the phone with a customer describing desired customizations, the salesperson could modify the 3D graphics accordingly – for instance, they could apply the customer's logo onto the dispenser surface as a texture. The two applications could be extended with plugins so that the salesperson's modifications would be automatically downloaded to the customer's brochure, so that the customer could preview the appearance of the non-standard product right after the salesperson makes the changes, effectively in real time. The steps involved in applying images to a MeshLab model are described in tutorials such as: https://wikis.utexas.edu/display/specify6/Texture+overlay+in+MeshLab. We could implement a plugin to automate some of these steps so that a salesperson could quickly construct a personalized graphic to show a potential customer.

A variation on this arrangement is to allow the customer to obtain a version of MeshLab with the same plugin as the salesperson uses. This would be more complex from the customer's point of view because MeshLab is a more complex program than a PDF viewer, but some customers might enjoy using these sophisticated tools to design a personalized product. The PDF viewer could then be embedded as a component within MeshLab. This could be an appropriate solution when interfacing with a large company with a professional graphic-design team, who would be confortable with the concepts and software usage associated with 3D artistry. We could also discuss developing plugins for dedicated graphics design software, such as GIMP or Inkscape.

On the other hand, some customers might not want to download any software at all. In this case the salesperson could still use the MeshLab plugin on their side, providing customers with a web page where their private version of the Astreea graphics – with customer-specific/personalized designs – could be viewed through an ordinary web browser (via WebGL). The only real difference in this case is that the 3D graphics would reside on a web page wholly separated from the PDF brochure, whereas in the interactive-PDF case, with a special PDF viewer, the 3D graphics would be visible through a window integrated into the PDF application.

Some conventional PDF viewers, it should be added, e.g. some Acrobat versions, *do* support 3D graphics to a limited extent. These require a special kind of 3D file which can be derived from MeshLab via extension code that we could provide. The 3D graphics in this case are inline with the PDF text, not in a separate window, so they're less user-friendly. Also customers would still need to use a dedicated web page to view real-time changes while they're designing personalized dispencers with a salesperson. But as an initial interactive brochure, Astreea could offer a PDF with interactive 3D graphics for those users who already have these kinds of PDF viewers and are comfortable with them.