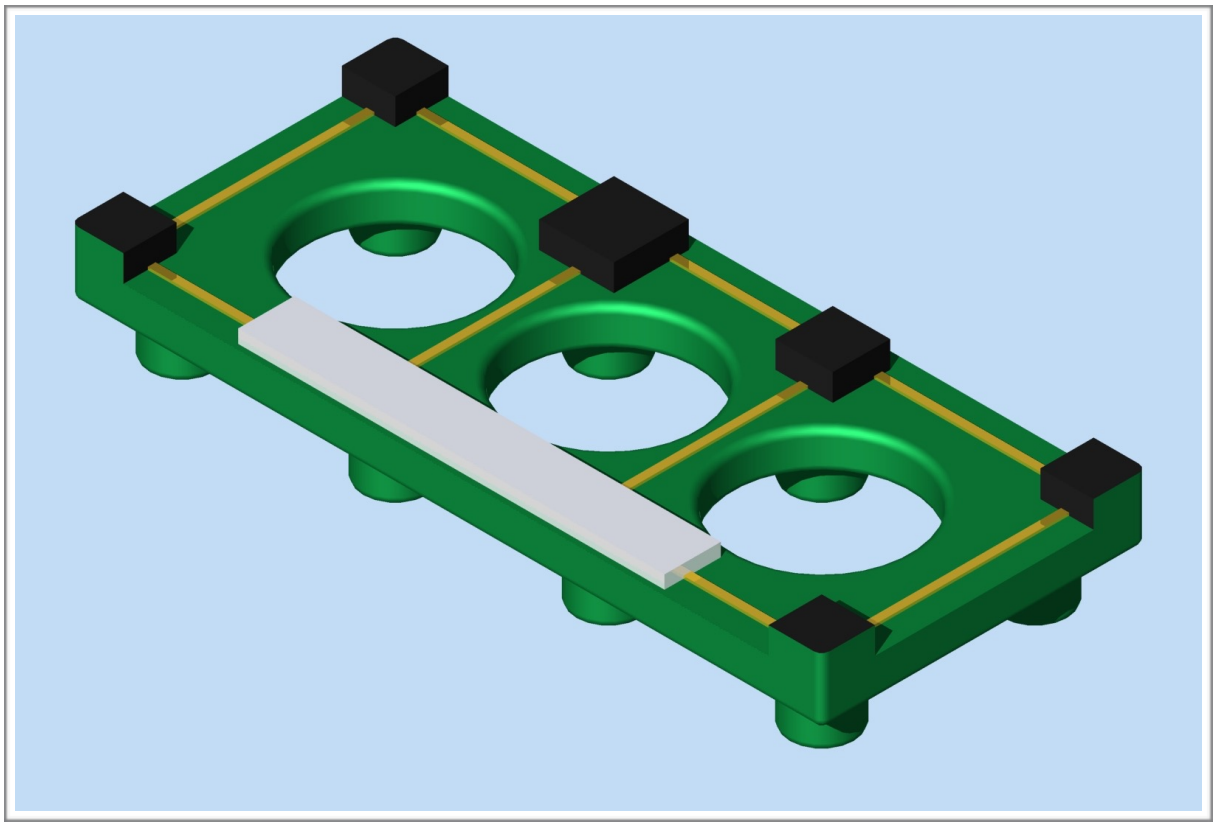
Human Computer Interaction

Smart Bricks



Michael Inglis

s1232123

October 2015

Introduction

The “Internet of Things” and “Interaction Design” are two different topics that share an almost symbiotic relationship in their implementation. Interaction Design defines the composition and actions of interactive systems. The designers of those systems must create relationships between the devices and interfaces and the people that use them.

My understanding of Interaction Design can be quantified by dividing it up into 5 principles -outlined by Adobe’s Vice President of Experience Design- all of which interrelated in implementation and are primarily concerned with what is experienced when a user interacts with the system:

- Consistency
- Visibility
- Learnability
- Predictability
- Feedback

In terms of **consistency**, we must have an experience which is uniform across the whole interaction. Users are sensitive to change, so having altering layout or interaction elements within the interface causes a distraction which unnecessarily draws attention away from the intended content.

The interaction must be intuitive. Elements which are interacted with must be **visible** to the user. The time the user takes to work out what an element does is important to their experience, so any possible interaction must be instinctive. We absolutely cannot have users searching for, or accidentally discovering possible functionality within the system. **Learning** the system should have the lowest possible learning curve. Most of the interaction should be deduced by the user, but things that can’t be carried out as easily, must be easy to learn and easy to remember.

Predictable interactions must too be paramount when designing a system. The behaviour of the user should show whether they are able to predict what the system will do as they use it. Users should at least have some expectation about the consequences of their actions which will result in them accomplishing their goal. The **Feedback** the user receives about their actions should not complicate their experience, but should provide a positive, understandable reaction. Failing to give feedback on interactions could result in the user unnecessarily repeating actions and becoming frustrated in the process.

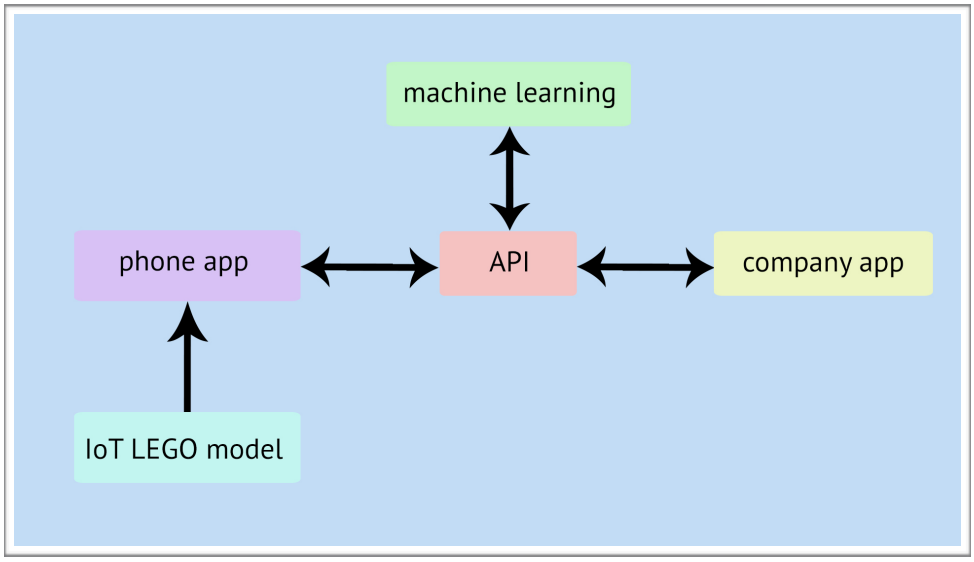


Figure 1.1

My understanding of the Internet of Things is that of a system which includes a physical object with embedded which electronics and software which allows it to connect with other devices, sharing information collected by sensors on the "IoT LEGO Model" as seen in Figure 1.1.. This data may then be processed and displayed in a format which is interpretable to who ever is looking at it. This is where the human computer interaction comes into play. The data which is displayed and the method it is interacted with must adhere to the five principles listed above. The hardware itself must also consider some of these principles. The objects use must be less of a tool and more of an extension of whatever it is monitoring, collecting data seamlessly and not obstructing whatever measured activity is vital for continuous data which can be acted on and analysed.

Design

Voted the "Greatest Toy of All Time" in 2015 by 100 of Britain's leading toy experts, LEGO bricks, manufactured by The LEGO Group, are one of the most complex and yet easily accessible lines of expressing one's creativity. Capitalising on many popular franchises in the last two decades, The LEGO Group has gone from a small Danish toy company, founded in 1934 to a worldwide sensation sporting theme parks, films and video games. Part of the universal appeal is the flexibility of what can be accomplished with the products. A set of interconnected bricks isn't limited to what can be built on the front of the box, but with what the builder's imagination can muster.

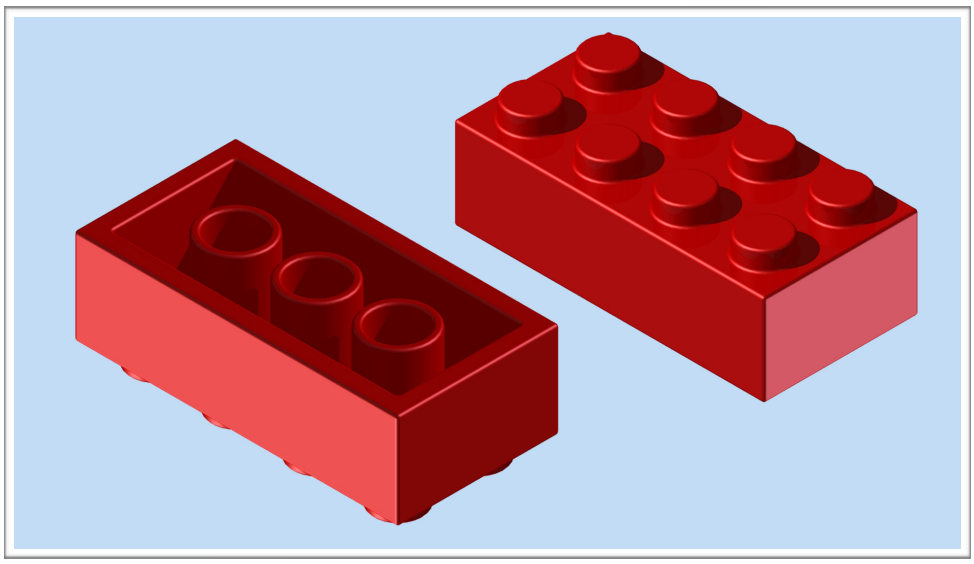


Figure 2.1

This creates a particularly interesting problem for how to best produce sets in order to comply with what their users want to create. The LEGO Group must figure out what people are doing and building with their LEGO bricks to fully cater to their audience and beyond. We have to work out whether people are building what is on the box of the set or designing something completely different with the bricks. This firstly enables us to consider how best to create the set; if people are keeping it as the model it was advertised as, then clearly the builder wants it as a display model, something that is static. This lowers the incentive of The LEGO Group to produce sets with a high number of extra pieces. The builder is buying it for the model and the model alone, they have little or no interest in how many spare parts are there or how many different permutations of bricks can be achieved. If however we can find that people are building new and interesting things with the set, they can accommodate for this in sets, producing them with a flexible amount of bricks, over and above what is included in the advertised model, with parts which allow for unique creations, as well as including more varied model instructions as to what can be produced.

Hardware

My proposal is a device which fits into the bricks produced by The LEGO Group and can be included in sets to allow the company to see what models people are creating. The device is unique to every different type of brick, prompting a unique signature to be emitted over bluetooth such that different bricks can be identified. For the sake of consistency, I will be showing a device which would be used within a 4x2 brick. The device senses a couple of things; whether or not the brick is attached to another brick from the bottom and what position the brick is in relation to the other bricks. This allows us to, in essence, create a virtual 3D model of the bricks the user puts together and will allow us to better understand what they create and how they create it.

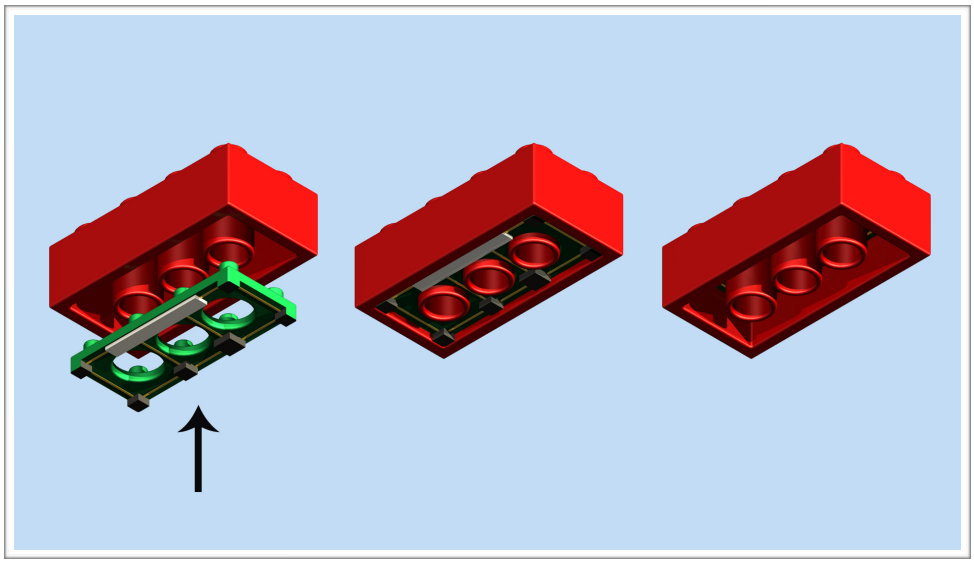


figure 2.2

If we look at figure 2.2, we can see where the device could be accommodated. The device, as mentioned previously, must be unique to every different style brick, simply to fit into each brick type and allow us to pick up what way something is attached to it in any possible permutation. This is done by 4 sensors, on the inside corner at contact with each of the inner side, which sense the amount of stress put on the sides of the brick, allowing us to accurately determine how the brick below is inserted. This also will enable the company to gauge whether or not people are merging their new Smart Bricks with older bricks due a brick having a "blind attachment", something we may see is attached, but has no feedback and thus having gaps appear in the virtual model.

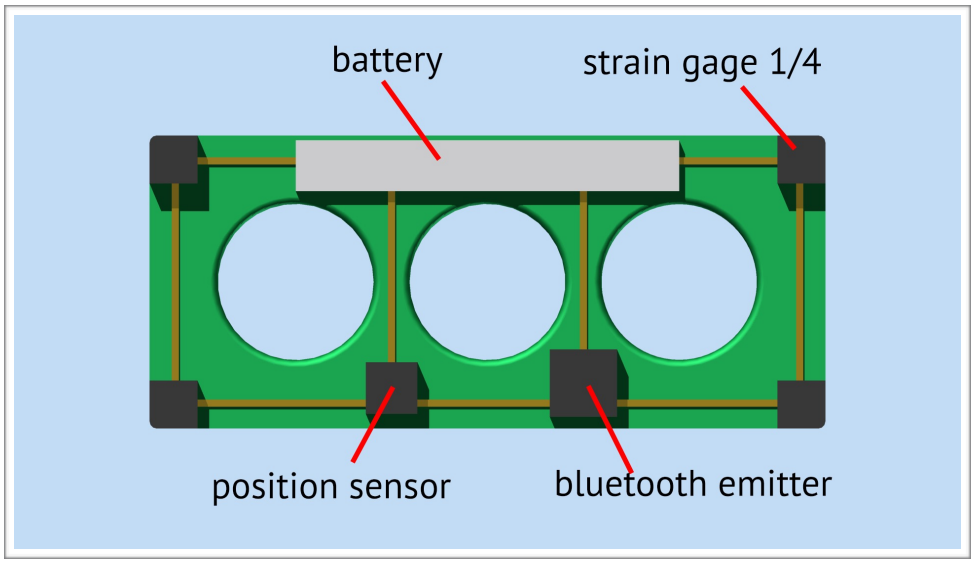


figure 2.3

LEGO bricks are tactile. It is designed to be put together with one's hands. Putting 3D models together on a 2D screen is no substitute for physically searching for a brick and adding it to the model. A delay is added to one looking for the brick, which will likely impact how the builder considers the design, often producing a different result to having access to an infinite amount of instantly retrievable 3D on-screen bricks. We are not hiding the fact this device is within the brick to the builder due to the applications which may be utilised on the user's end, what we do have however is a device which isn't intrusive to the building process the physical bricks were designed do.

User Application

To accompany our device, there will be a User Application. This interacts with the sensors via bluetooth on the builder's phone. The core functionality of the app allows users to generate virtual 3D versions of their models based on the types and positions of the bricks. This allows users to carry out functionalities such as instruction generation or sending models to their friends. This functionality is simply a front to incentivise the user to use the app. In the background however, the app can be sending the user's models back to a server using an API.

A mock-up screen can be seen in figure 2.4 and 2.5. The physical bricks with the inserted devices are displayed in a 3D view in which the user can manipulate, using their fingers to rotate by swiping, zoom by pinching and inspect individual pieces by tapping, an action seen in Figure 2.5.

Figure 2.4

Figure 2.5

API & Machine Learning

The API will be explored more in depth in the Implementation section, but in terms of basic design, it will function as a link between the user of the phone application and the application used by the company. The data received by the API can be preprocessed to produce data which is suitable for viewing within the Company Application. This API will also include forms of machine learning and pattern recognition on the builder's data to classify what people are doing with their LEGO bricks. Machine Learning can be extended in the API to determine the level of expertise the builder of the model is, drawn from what bricks are used and in what permutations.

Company Application

The Company Application is simply a front end for viewing the data collected and processed by the API. It allows the LEGO Group to easily view and draw conclusions from the large quantity of data drawn from the Users and their Smart Bricks. This data is presented in a visual form which is presentable to non technical users. It will have statistics, displayed in graphs on how people are using the bricks, whether they remain static or dynamic models and can provide examples of user's models displayed using the same 3D tools as we see on the User Application.

Implementation

Hardware

The bulk of the material on the device is taken up by the PCB board. It is shaped to insert into the bottom of a brick. Each brick requires a different cut board and a different positioning of the strain gauges to suit the brick. Some bricks will be undoubtedly too small to fit this device into, but more often than not, these pieces can be discarded as they are mostly irrelevant to the model's main design.

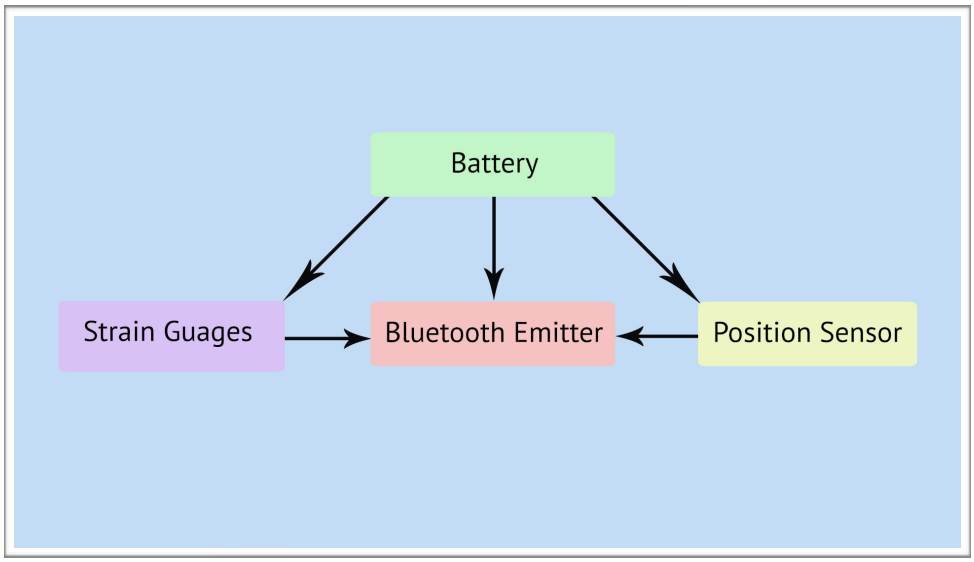


Figure 3.1

The battery is arguably the most fantastical part of the device. Due to the size of the device, I opted for it to be non-removable and built into the PCB. In terms of charging too, I opted out of this in favour of single use devices simply because a standardised port on the PCB would be far too large. We can reduce the energy consumption of the device by taking measures such as switching off the bluetooth and position sensors if the strain gauges remain idle for a certain period of time, occasionally pinging the User Application too in case it is at the base of the model, only attached from the above. The device in the end however has to settle on a short lifespan, with a depleted battery rendering it useless. While this diminishes it's appeal as a viable product, functionally nevertheless, it remains the same.

In figure 3.2, we see just some of the possible permutations of two bricks. This is the main reasoning behind the quantity strain gauges. We would of course require more than one to to counteract any false readings from environmental changes such as temperature. With one on each corner, we would be able to accurately determine, based on each sensor's value, how many, and where the studs were attached to the base of the brick. These values, one for each sensor, are recorded as floats and sent to the bluetooth emitter in an array.

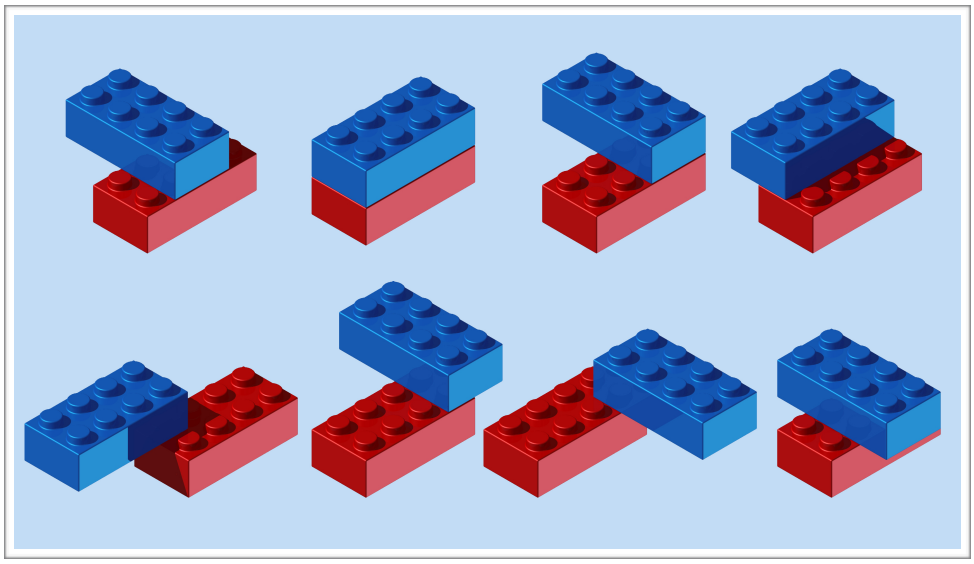


Figure 3.2

The position sensor takes note of the bricks position in 3D space, sending an array of 3 floats to the bluetooth emitter. The sensors measure their position in relation to a randomly selected "master brick" piece to ensure each one's position is relative to the others within the global position. The user may flick between which brick is the "master brick" within the User Application if they are experiencing odd results due to low batteries or outliers.

All the data from the sensors is passed onto the bluetooth transmitter which is forwarded onto the user's phone using the File Transfer Profile (FTP). Each brick transmits a packet of data of the tuple (brickType::int, xyzCoordinates::[float], strainValues::[float]) which can be interpreted by the User Application on the builder's phone.

User Application

The User Application is an app developed for iOS or Android. It acts simply as a front end for the API, displaying visually how the hardware is assembled. The data the application receives through bluetooth simply includes the device's brick type, a three dimension position vector, and values determined by the stress sensors which determine if and how a brick is attached to the base.

The app iterates through each brick picked up by bluethooth, determining how it's position relates to the it's global position and it's relative position to other bricks, based on the base attachment values and the type of brick it is. The bricks can be rendered to the screen using a library of 3D parts contained within the app. In order to make the bricks appear more realistic, Phong Shading can be applied to each vertex. We may also implement

transformations, simply by multiplying each vertex by a transformation matrix. This however is unlikely to require implementation at such a low level as 3D libraries such as Cocos3D exist and have access to basic 3D model viewers which only require settings to be tweaked to make them specific to the 3D models we want to render.

Further features include a friends list that the user can send their saved models to. This information is pulled down from the API and displayed as a friend list, primarily alphabetised, but the most regularly interacted with, "best friends" take priority at the top of the list as the user uses it, easing their search time. Users can simply send models to their friends as it is what our API already handles. The current model is uploaded every time the user opens up the app, allowing the API to check how it has changed or if it has changed at all.

The user also has the ability to generate a set of instructions for their model. This is done simply by taking a list of 2D isometric renders of the model, adding one brick each render by working its way up from the lowest to the highest y coordinate of each brick. This creates a simplistic, intuitive list of images which can be displayed on a page before being exported and saved as a PDF, a format which is understood universally across all devices. These features are simply to give the user a reason to open the app so the models can be uploaded to the API in the background.

API

The API will be designed in the lightweight Python framework Flask, using the Restful extension, an abstraction for building APIs. The bulk of the processing is carried out on the API, receiving raw data, and sending the findings to the Company Application. The only processing the data receives on both Applications is the visual elements which enhance the user experience such as the 3D model view.

The User Application sends HTTP requests with the brick type, position and sensor values in the JSON format with the user's username and password in the header. This is interpreted by the API which associates the model with the user's account and if it is to be be sent to another user or simply adds it to the "pool" of all the other models.

For functionality on the user's end, we require a basic login system which stores attributes such as "friends", other users, and "models", a timestamped collection of brick positions. This can be carried out easily using a technology such as "Stormpath" which offers scalable, secure, login system, so that we don't have to deal with the ethics of storing a user's information. This service interfaces with our server using their API, sending and storing the user information in JSON.

Once there is enough data collected, we may begin to use unsupervised learning methods such as hierarchical clustering algorithms in an effort to understand the data. This can include any such attributes as part use, design choices as well as how the builder uses the model. Once unsupervised learning has been carried out and the system architect better understands the trends within the test data, it is then possible to carry out some supervised learning, classifying attributes such as building ability as well as determining if the model remains static as seen on the box it came in. These attributes can be packed into feature vectors which can help us understand overall trends within the data and how attributes may correlate with others. This can be done using a variety of learning methods, such as Gaussian Models or Deep Layer Neural Networks, the best of which we can determine by testing the efficiency of the algorithm using cross fold validation on a manually classified set of data.

Given that this is the data, we are designing the system to collect, I cannot comment with absolute certainty on the trends which will be found within it or which learning methods would be best suited to get our maximum information gain, I can merely propose how I would consider approaching the problem.

Looking how the API interfaces with the Company Application, it would again be similar to the User Application in terms of sending specific models across JSON. The main functionality of this end however, would be pushing the results of the machine learning to the Company Application. This would include a substantial quantity of statistics on what users are doing with their models as well as specific models the system deemed significant.

Company Application

Like the User Application, the Company Application is simply an iOS or android application. It's main purpose is to allow the user on the company's end to easy visualise and interact with the data drawn from the API. The user would have an administrator account contained within Stormpath. The login which is again included in the HTTP request sent to the API, grants them the privileges to request the relevant data.

The 3D viewer described in the User Application will function exactly the same here, the only differing component being the models which are displayed and the fact they are taken from the API, not the sensors in the builder's bricks. The app will be dealing with the visualisation of statistics, so they must be easily be interpreted by the user. This can be done for example using "Core Plot" in iOS, a framework which allows us to display the data in many different 2D representations. With this we may add animation and colour to our charts, enhancing the user experience along with the well known gestures we would expect from a touch screen application.

Conclusion

The data that is created by these devices on a large scale allow for a research and development department to tailor the products the company produces to what their consumers are doing with it. What people want from LEGO bricks is too subjective and cannot be quantified easily, so having what is in essence a giant product testing focus group, where people aren't aware the information about what they're building is being collected gives us a decent insight into what people want to make with their sets in their own home.

One can gauge a decent understanding as to what people are creating with their bricks by looking and engaging with builders in online communities such as Flickr. This however is only a small subset of the whole demographic, with a skill level likely far higher than average. These people are also extremely active users; it is them who are posting their photos online, displaying to others, often only the best of what they are building. What we want to find out is what the other users, who are not not posting images of their models are creating. This can be captured passively using this device, simply by incentivising them through means of friend sharing and instruction generation to use the user application.

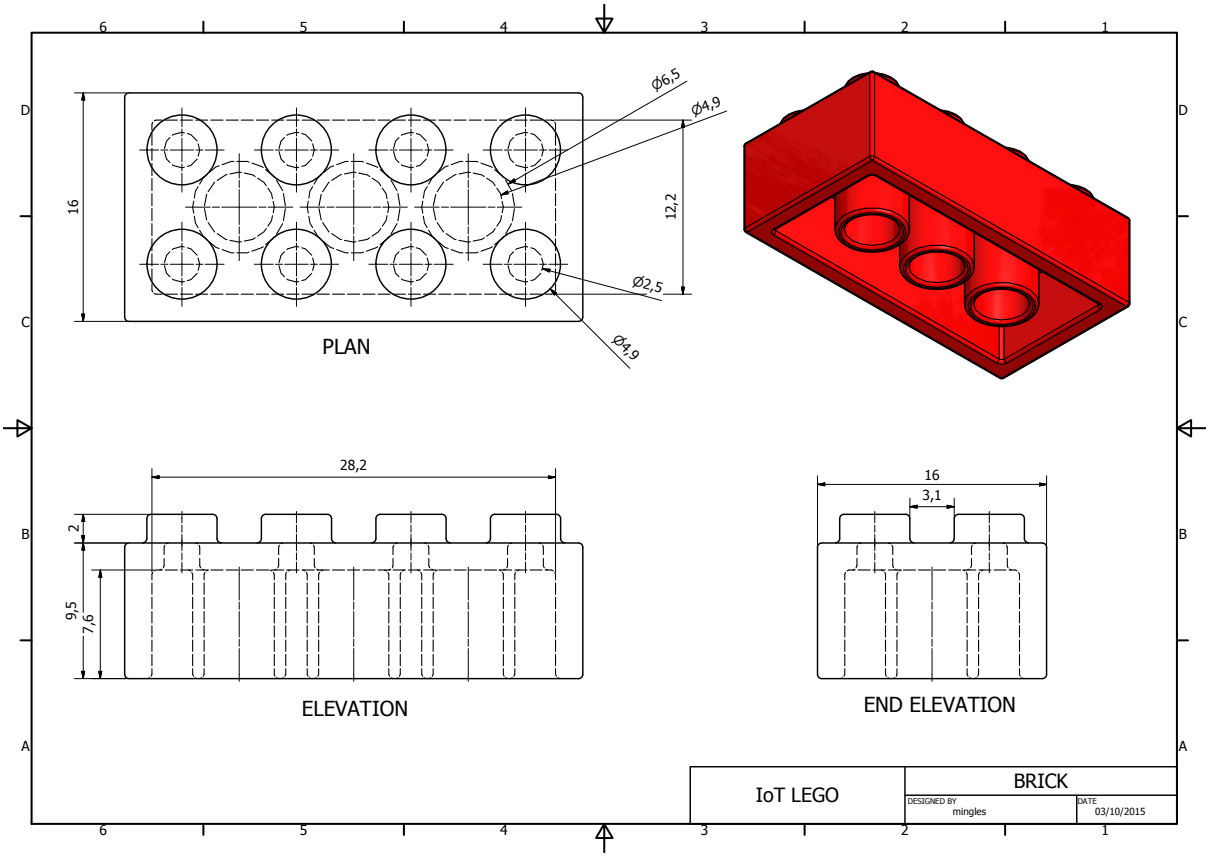
The LEGO group does indeed have a service, "LEGO Ideas", which allows users to crowd source ideas, but has people voting primarily for the high level concept rather than the brick design due to the fact The LEGO Group simply recreate the model on their own terms if the idea is approved. Our product can be used very well in conjunction with this, given that the company itself redesigns the model. This kind of service brings in a lot of users which aren't part of the usual demographic due to a lot of the submitted "Ideas" being parts of existing franchises or other hobbyist niches which haven't previously been made into sets. Using the Smart Brick device, we may properly understand the building patterns of people who don't regularly buy LEGO in an effort to gain better insight on how create sets which cater to them specifically, thus expanding the range of product, not necessarily trying to find the lowest denominator product.

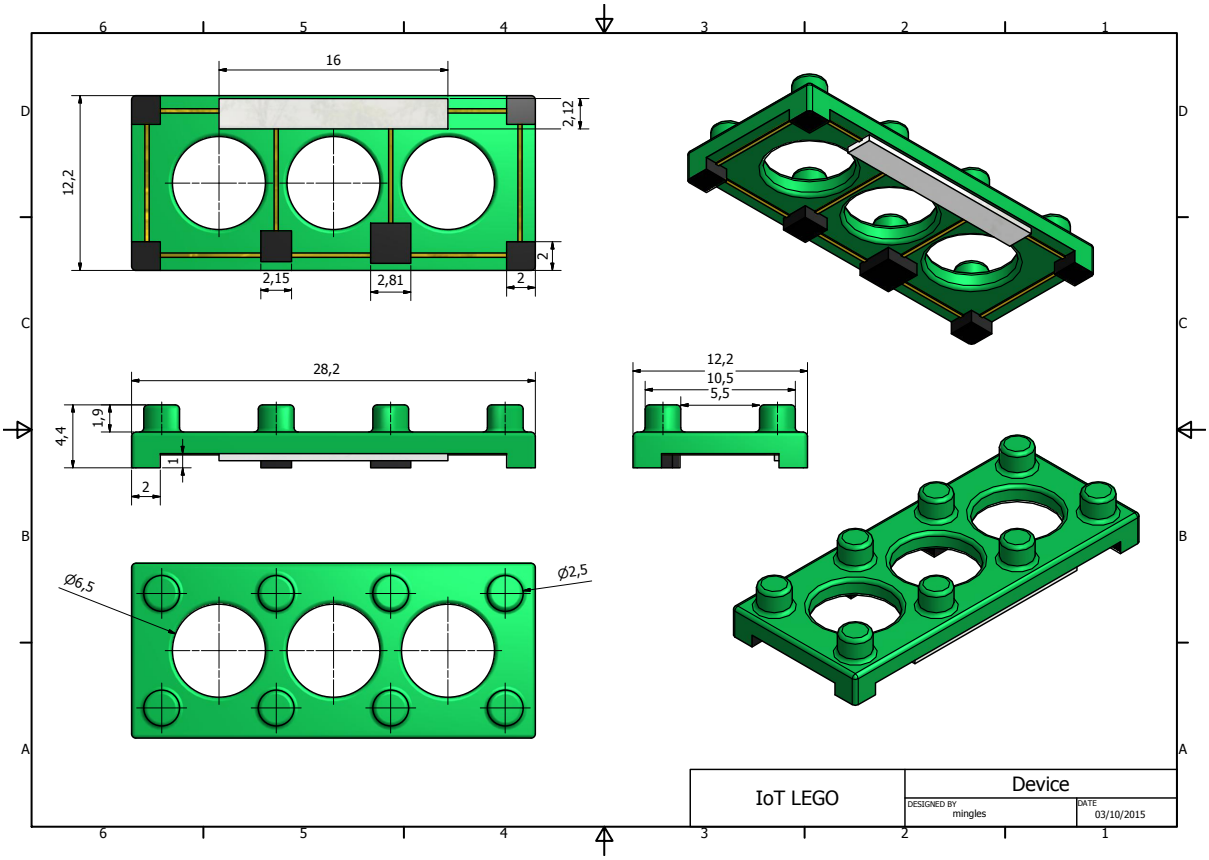
While the hardware limitations of the battery clearly limit the potential of the device over the long term, it will still serve it's purpose in the short term. It simply only needs to work several times with the application for us to get the general idea of what the people are doing with their bricks. Technologies such wireless charging could be perhaps possible to implement as the limiting factor here is simply the size of a standardised charging outlet. Another option is a "charging brick", a brick with metal plates on the edges of each stud and an outlet which connects to power. When a normal brick is inserted onto the charging brick, a line connected to the battery comes into contact with the metal planes and the battery can charge, allowing extended use of the device.

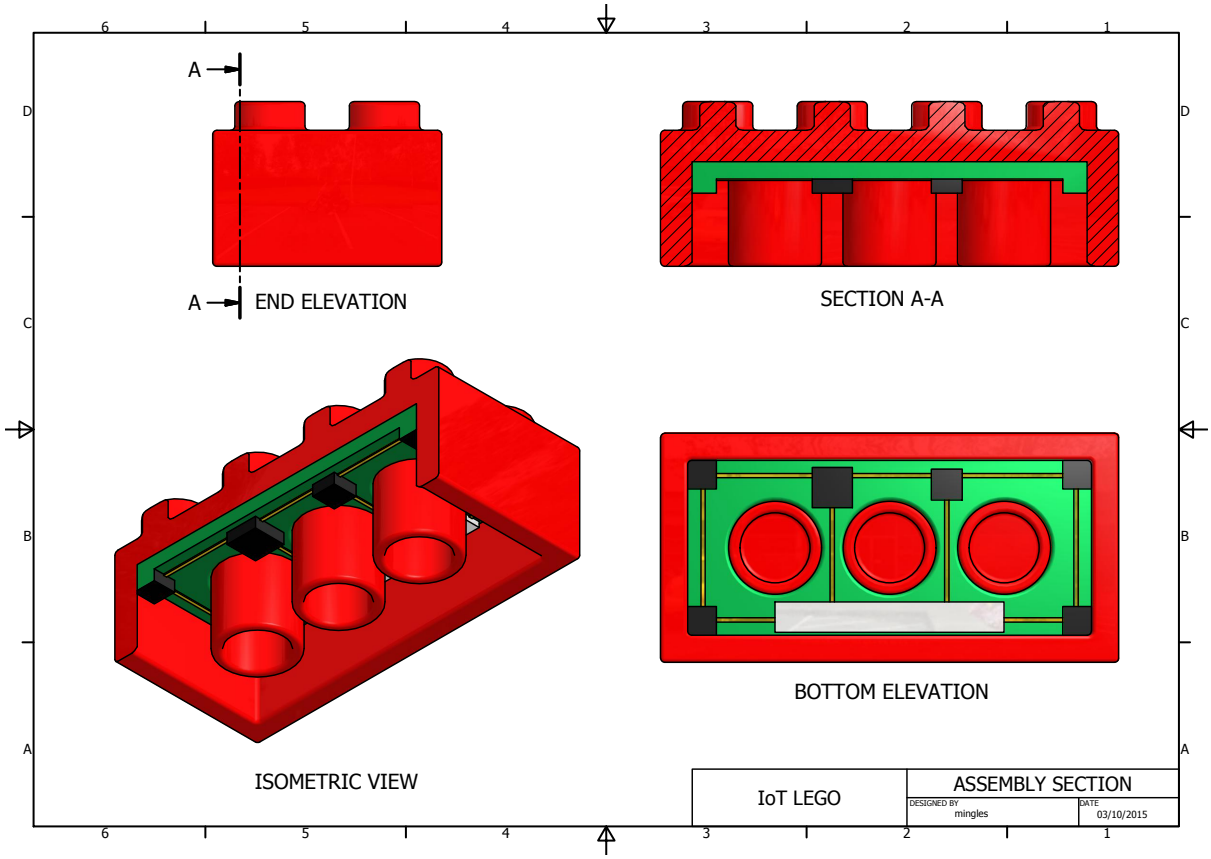
Given that the devices are bespoke to each kind of brick, this requires a significant amount of design for each PCB in terms of scale, shape and quantity of strain gauges. This is not only expensive to design, but extremely expensive to manufacture, so given that they are currently disposable when they run out of battery, this is the first thing that would need to be addressed. Some kind of recycling program would have to be introduced, especially if the company was simply giving them out for free with sets. It would be a very expensive, labour intensive project to collect information.

The device will likely also be too small for certain bricks such as the single stud bricks. These however are merely 'decoration bricks' and will be largely irrelevant to the model as a whole, certainly for looking at how/what people are building. The only issue that will likely arise from this is builder frustration in sending models to their friends. This could be resolved with the ability to manually add virtual bricks to the 3D model within the User Application before sending them to the API.

Appendix


ϕ6,5
ϕ4,9
16
12,2
ϕ2,5
ϕ4,9
PLAN
28,2
2
9,5
7,6
ELEVATION
16
3,1
END ELEVATION
IoT LEGO
BRICK
DESIGNED BY
mingles
DATE
03/10/2015


16
2,12
12,2
2
2,15
2,81
2
28,2
1,9
4,4
1
2
12,2
10,5
5,5
Ø6,5
ϕ2,5
IoT LEGO
Device
DESIGNED BY
mingles
DATE
03/10/2015


A
A
END ELEVATION
SECTION A-A
ISOMETRIC VIEW
BOTTOM ELEVATION
IoT LEGO
ASSEMBLY SECTION
DESIGNED BY
mingles
DATE
03/10/2015

References

https://en.wikipedia.org/wiki/Internet_of_Things

https://en.wikipedia.org/wiki/Interaction_design

http://www.dailymail.co.uk/news/article-2938297/Lego-voted-greatest-toy-time-Monopoly-Action-Man-Rubik-s-Cube-50-crowd-funded-newcomer-Cards-Against-Humanity.html

https://ideas.lego.com/

http://tv.adobe.com/watch/classroom-five-essential-principles-of-interaction-design/part-1-five-essential-principles-of-interaction-design/