Final Year Project

The Internet of Things Era and its Applications

Author Vikas Bhatia

C1366554

bhatiav1@cardiff.ac.uk

Supervisor Prof. Omer Rana

ranaof@cardiff.ac.uk

Moderator Dr. Kirill Sidorov

K.Sidorov@cs.cardiff.ac.uk

May 2016

Cardiff University School of Computer Science and Informatics

Abstract

The Internet of Things (IoT) is, conceptually, not new. For the past two decades – ever since the development of microprocessors and network-based appliances – companies in the major industries such as pharmaceuticals, manufacturing, gas, oil and mining have been eagerly exploring how to use sensors to make their processes more consistent, effective and safe. The practice of connecting "things" to each other with a network has only been given a buzzword that is becoming mainstream, but has been around for some time. The distinction here is the expansion to the consumer market. While industrial applications may have reached a certain level of capability/maturity, the trickling down of this technology into our everyday products reveals that IoT is still a relatively nascent field with challenges in this context, the benefits of which the average consumer is only just beginning to understand, experience and explore. In this project, I aim to demonstrate the benefits of the IoT era, which is bringing previously-disconnected devices into the IoT domain. I intend to analyse the impact of devices having "IoT capabilities", and show the problem-solving potential of network-connected devices from a business (social/economic) and technology perspective. Where applicable, I will explore the potential of converting a target device into a network-connected one; this will be done by building a physical hardware module deliverable that sends communications over Wi-Fi. I will use this an example of a first-hand investigation into the costs, knowledge, effort, and equipment required to personally build a scalable solution for deploying network capabilities on devices.

Acknowledgements

I would like to thank my supervisor, Prof. Omer Rana, for his inputs during the course of the project, complemented by his own exploration of the developing Internet of Things field. I am very grateful to him for helping me bring focus on the requirements of the project, and lending his expertise on the best practices to follow. I thank my family for their unmatched support throughout my degree, to whom I owe all my success. Their confidence and belief in me has been inspirational. I thank Bird Electronics Pvt. Ltd. and Cloudnix Software Labs for their guidance and assistance in setting up various parts of the tools and infrastructure investigated in this project, including support in areas that were previously outside the scope of my knowledge. I would like to thank Mr. Lawrence Semmens, technician at Cardiff University's School of Computer Science and Informatics who went out of his way to help me with lab equipment and putting together my hardware. I would also like to extend my appreciation to my friends Shazaib and Nathan, who have given me great advice, friendship and moral support ever since I have known them.

Table of Contents

1 Introduction

1.1 Outline

The Internet of Things (IoT) has become a vibrant part of the IT landscape. IoT means many things to different people, from smart homes to wearables to connected cars, with different developer ecosystems built up around each IoT pillar. As a megatrend in 2016, it is going well beyond a hobbyist developer's next fun project, to becoming a space where technology vendors and 4.5 million IoT developers are gearing up to invest their time [1].

The Internet of Things revolves around increased device communication; its foundation is built on cloud computing and (networks of) data-gathering sensors. It is mobile, virtual and instantaneous, with the promise of making our everyday devices "smart". However, it is also a relatively nascent field with challenges.

The devices that can benefit from such a conversion have some common problems. They are often complex and expensive. There are competing standards and approaches for adding smart capabilities to them, which may be expensive themselves. They are placed in off-site/remote locations, and maintaining them may be a challenge. If the device becomes defective, service engineers may take a long time to diagnose issues because they lack the context of the machine's recent operation, and what spares may be required. These devices may even perform poorly since there is no established way to collect data from them or centrally monitor them.

This project shall focus on enhancing an existing device with Internet/network capabilities in a way that mitigates the issues mentioned above by prioritising simplicity and low cost above others, logging data from it, and showing that the real value that the Internet of Things creates is at the intersection of gathering data and leveraging it to solve these problems. Potential use cases for this will be explored, with the main use case being bringing IoT capabilities to a vending machine. Through small modifications to the electronic control system of such a device at a very low cost, implementing an IoT solution will allow the collection of data such as sales patterns, total revenue generated, usage statistics, and more. This data may then be transmitted to a central server. With this information, the potential benefits are (including but not limited to) remote diagnosis and monitoring, reduced turnaround time to repair the device, increase in revenue and uptime, and a reduction in wastage of resources.

1.2 Summary of Aims

The purpose of this project is to develop a proof-of-concept low-cost solution to bring network capabilities to a theoretical “target device” (appliance), in a common use-case scenario where being able to collect data from the previously-disconnected machine would be useful. The solution should be in the form of a Wi-Fi module which would be able to receive inputs from a host machine specified by the user, each pertaining to a parameter which is to be collected. The module must then be able to send these inputs to a specified location such as a central server in a standardised format and mode of communication, where they may then be logged, and where a software developer would be able to create applications to leverage this into actionable information. If possible, exploration into creating a sample dashboard where the data points are shown will be conducted.

1.3 Project Scope

The solution that I will be presenting will adopt standard technologies and communication methods previously researched and established in the consumer market, and apply these in both the hardware and software side of the implementation. Of the known methods of wireless communication and communication standards implemented in devices (such as Bluetooth/Wi-Fi/Z-Wave/ZigBee), Wi-Fi will be used. Deploying on an actual real-world target device is unnecessary given time/resource constraints; a sample representative piece of hardware that has a microcontroller will be acquired and programmed as a proof-of-concept. The solution will be presented as a Wi-Fi module which is able to connect to an access point and send data to an external server. The environment and programming will be in C, using the KEIL µVision IDE. The language and environment will be evaluated and justified. A number of existing code examples provided by the microcontroller vendor, Freescale, will need to be integrated into the application to ensure the solution is met in the specified time.

1.4 Project Outcomes

The solution will ideally be simple, economical, and use-industry standard interfaces which will adapt the module to internet-enable a target device. This will be coupled with a central server which is set to receive data from the module. I will evaluate a commercially available Wi-Fi module for this use, which will be the basis of the implementation. A contrast will be made with alternate modules/methods of connecting to the network, including a comparison of the benefits of using an independent Wi-Fi module vs. an out-of-the-box IoT board such as the Raspberry Pi. The events to be communicated (such as button presses on the device) will be evaluated, and sent to the server, where their usefulness may be interpreted in the context of the particular use case. Sample use cases will be discussed. Due to time, resource, and skill constraints, the actual setting up of the server infrastructure will be implemented with help from external vendors. Coding a dashboard to analyse data is outside the scope of this project. This project will have been considered achieved if the data is deposited in a specified location in a usable, standard format (such as for a software developer creating an interface to monitor this data).

1.5 Project Structure

The initial sections of this report are centred around a number of different areas regarding the emerging technologies in the Internet of Things. Research is presented on IoT interests, through exploring previous work in the field, as well as the established and competing standards in the industry on implementing smart capabilities on devices. Latter sections in the report focus on the solution and how the researched issues can be addressed, adopting some of the findings presented from the earlier research and justifications for these choices.

Background research will focus on what the advent of the Internet of Things era means for the average consumer and wireless communication technologies that are being leveraged to fit this trend. The reasoning and benefits for connecting devices to a network will be discussed. A brief introduction into alternatives will also be explored, in order to allow a contrast between offline "dumb" devices and internet-connected "smart" devices. IoT services and their cloud infrastructure will be investigated in order to identify key components and the flow of information as a system diagram. Challenges currently faced in this burgeoning field and reported cases of controversy surrounding potential exploitation of sensitive data from IoT devices will also be visited, with an overview of surveys presented.

The approach in section three of this report largely focuses on the original project aims and the differences between what has been achieved compared to the initial plan. Parts of the system diagram discussed earlier will be re-visited with consideration of the hardware purchased, costs, and code written for it, given in the later sections of this report.

Implementation of the application will be discussed, outlining the hardware used, programming environment, coding language, and scalability. The code used will be discussed in this section, mentioning the sample code acquired from the hardware vendor. Code written specifically for my application will be explained in key areas of functions, each of which may demonstrate programming concepts pertaining to the embedded domain.

This will be followed up by an evaluation of the aims achieved and the limitations of the implementation. To conclude, there will be a reflection on personal development, lessons learned and refinements that can be made given certain resources to build on the proof of concept.

2 Background

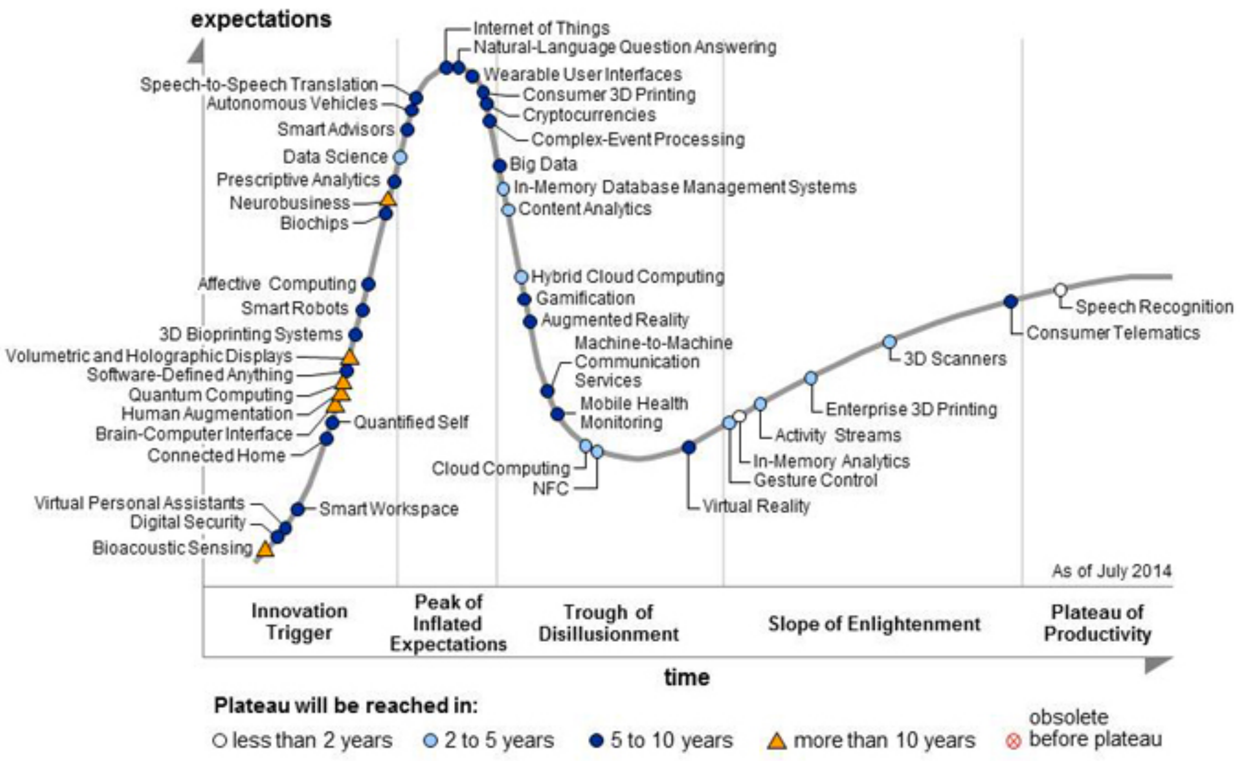
Our homes are increasingly the domain of smart refrigerators, autonomous robots such as Roombas and 3D printers. Recently, we have started looking forward to self-driving cars and personal consumer drones. The "Industrial Internet" may have existed before, but now, start-ups such as FitBit and Withings are taking familiar old devices, such as step counters and weighing scales, and making them intelligent by adding microcontrollers and internet connections to them. No longer are sensors and networks the domain of jet engines or locomotives – the idea has started permeating our everyday lives. Every interesting new product seems to have a network connection, whether it is Wi-Fi, Bluetooth, Z-Wave, ZigBee, or even a basic USB connection to a PC. Everything has a sensor, and devices as wide apart as a smartphone and an alarm clock are stuffed with them. The intersection of these trends – the conjunction of hardware, software, networking, data, and intelligence is much more significant than any recent development in the world of technology and computer science. This is The Internet of Things.

IoT has officially taken over Big Data as the new, most hyped technology.

.

The special report [2] by research firm Gartner provides an insightful outline of the "market promotion and perception of value for over 2,000 technologies, services and trends in 119 areas." In an analysis of the technologies, services, and areas of research and study that have changed most from 2013 to 2014 in their position on what they call the "hype cycle", time to plateau, rating, and adoption rates, Gartner has found that what they call "the four Nexus of Forces (social, mobile, cloud and information)," were highly correlated with the profiles that had changed most significantly on the peak portion of the Hype Cycle. Two trends Gartner called out specifically as having an influence at initial stages of the Hype Cycle were digital business and the Internet of Things.

The world has witnessed a boom in software development over the last few years from some truly revolutionary developments: pervasive computing can make the internet a central part of any app.



Business models have changed to providing services rather than products. New assumptions can be made for a consumer's ownership of a baseline hardware and connectivity – if the internet is available on a device, it immediately turns into a portal connected to a much larger world with a near-infinite set of possibilities. Application Programming Interfaces (APIs) make modules available to everyone to leverage easily. Open source projects can make code and knowledge available to anyone to pick up on. An explosion in cloud services right from Microsoft's Azure for Enterprise, down to Amazon Web Services can dramatically reduce the capital needed to start a new software venture. These same equivalent developments are now, in a sense, coming to the physical world. For instance, GitHub enables easy access to expertise from open source projects hosted and made available online in the software domain. This is now trickling into the physical domain with devices such as the Arduino. APIs allow developers to build software on pre-existing platforms, such as the Facebook API. In the hardware world, the service IFTTT ("If This, Then That") is extending that form of innovation and interaction to our physical devices.

Tom Tunguz, venture capitalist at redpoint perhaps put it best when he says "The real promise of the Internet of Things isn't simply linking millions of devices together, just like the real innovation of the web wasn't networking a bunch of computers. Instead, the true and still unrealized potential of IoT is to transform business models; it's enabling companies to sell products in entirely new and better ways that benefit both the company and the customer." [3]

2.1 Previous Work in IoT Technology; Emerging Nature of IoT

Software and hardware working together is an idea that has existed since at least a few decades. TVs and cars have had software-driven components since the early ‘80s. Even kitchen appliances such as ovens and cookers have timers and programs and cooking settings – all these need microcontrollers and software. This brings up an important question of what makes the Internet of Things so different and important. The hardware renaissance of the last few years entails more than just embedding processors into devices. It is the pervasive/ubiquitous computing which changes the game profoundly. Devices that have controllers in them are no longer alone. Your Nest thermostat connects to your smartphone, which connects to your speaker, which dims your lightbulb when playing soft music streaming from another server. It is this we have begun to recognise as the Internet of Things.

In the past, the focus has been on putting in as much functionality into a device before it shipped, since there was no concept of cloud computing in the manner we have today. There were no updates, no changes to an experience that was once received out of the box. Smartphones more or less retained the same software that they shipped with, and were hence made as powerful as the manufacturer deemed at the time, and “frozen”. The functionality in new hardware has been seeing a trend toward modest, “functional” local computing capabilities, backed up by powerful cloud computing. A smart watch such as the Apple Watch, for instance, can carry out some functionality, such as basic data parsing and fitness tracking on local hardware, but it depends on sophisticated machine-learning algorithms on an ever-improving cloud service to offer further insight as the data is fed to an iPhone with the Health app, which in turn may be read by other applications and devices that work in conjunction with the data logged to create a health plan for the user. While it is undeniable that devices such as the Apple Watch have their share of software and hardware limitations and issues that need to be refined and eliminated, they have been credited to make significant impacts already by saving lives on more than one occasion. [4] [5] The Internet of Things promises to make our lives easier, better and safer in more ways than we can picture right now.

Home automation is proving to be a hotbed of useful innovation. The most important devices to automate may just be the most boring ones. These are the devices that do something essential (such as climate control and HVAC), but a consumer never bothers to actively control. Will the Internet of Things give us more comfortable homes because our temperature sensors communicate with our thermostats to turn up the temperature before we come home? Yes.

Networking is hardly a new development either. We can trace back fully functional computer networks all the way back to the days of the ARPANET. Buildings have had connectivity in some form or the other for a few decades. Vehicles have had a standard since the ‘80s too: development of the CAN bus started in 1983 at Robert Bosch GmbH. The protocol was officially released in 1986 at the Society of Automotive Engineers (SAE) conference in Detroit, Michigan. The first CAN controller chips, produced by Intel and Philips, came on the market in 1987. The 1988 BMW 8 Series was the first production vehicle to feature a CAN-based multiplex wiring system. [6]

Yet again, the difference comes in pervasive computing. While remote access to safety features in a car has been there for years, what makes it different today compared to the IoT predecessors of a few decades ago is that no longer are our devices alone. We now comprehend how to build standards, build protocols upon which devices from different manufacturers can interoperate not just with their own technology but with technology from other manufacturers and other classes of devices at large.

Pervasive computing is now beginning to collide with the Big Data Era, which is manifesting itself as the Internet of Things Era. Since working with huge datasets has become significantly easier and inexpensive with our cloud infrastructure compared to the previous decade, everything that touches software has become instrumented and optimised. Education, manufacturing, consumer technology, medical, advertising, retail, e-commerce, logistics and shipping, and practically every other area of study and industry has sought to model itself and adapt to achieve greater efficiency in this space. We now have software that can accept data, process it, and send out actionable commands based on that to devices and users in real-time.

2.2 Shifts in Capabilities of Existing Processes

Software and hardware are merging into a single, flowing, unified discipline, with a single development routine that incorporates both software and hardware. While before, there were a few people who needed a deep understanding of processes in their respective discipline, now, as was my experience in the course of this project, many people will soon need some integrated understanding of both hardware and software. Software that exists on servers, operating above the level of a single smart device, can correspond with masses of physical devices, each fetching data from the server, interpreting that data in a global context, each processing signals such as button presses or events and exchanging them with each other in the grid. This is a fluid system of hardware and software.

As intelligence is permeating the physical world, we must also consider new manufacturing techniques. In the past few years, there have been shifts in existing business processes such as manufacturing: from the earliest prototyping stage through the largest mass-market production runs, CNC machine tools, CAD software, and new supply chain models are transforming the way that companies create and build hardware. The rise of additive manufacturing, otherwise known as “3D printing” is bringing the promise of manufacturing (rapid prototyping) to people’s homes. The IoT era has the potential to disrupt economies of scale given this unique fusion of skills.

2.3 Empowering Individuals: Disruptions in Economies of Scale

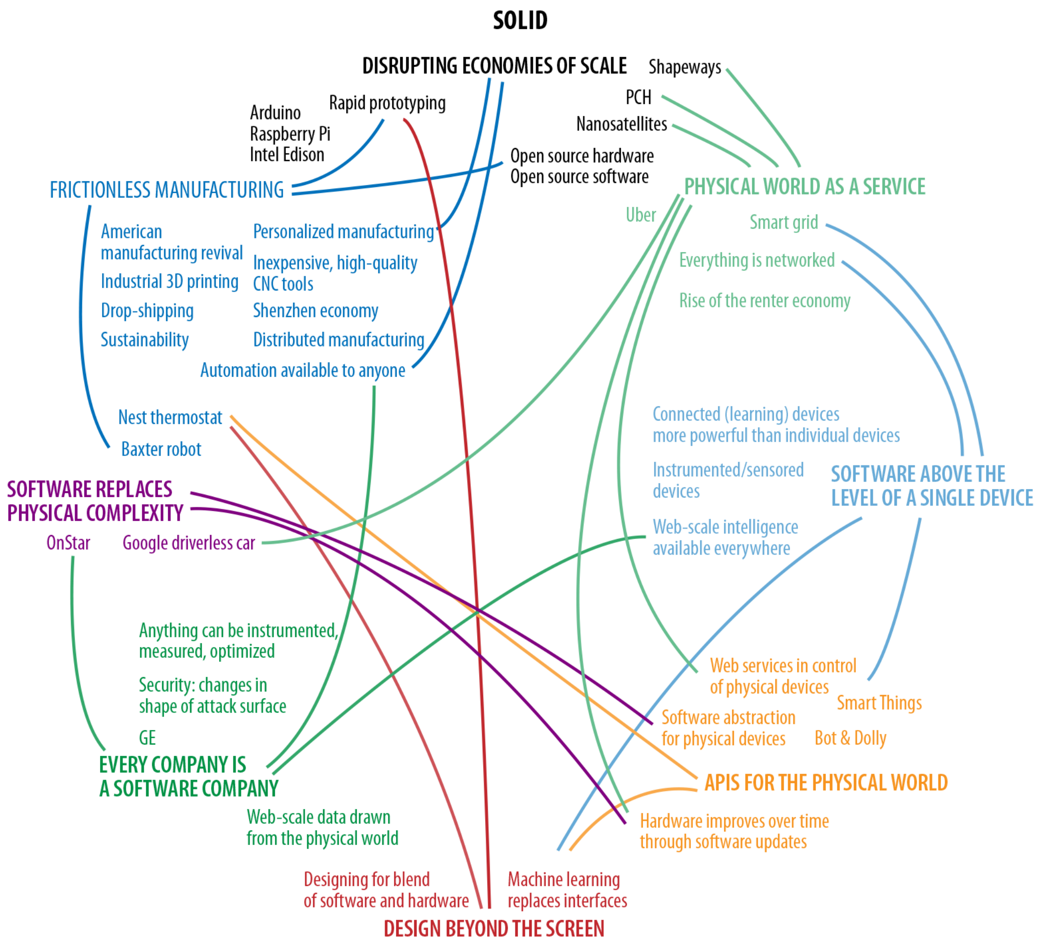

SOLID
DISRUPTING ECONOMIES OF SCALE
Shapeways
PCH
Nanosatellites
Arduino
Raspberry Pi
Intel Edison
Rapid prototyping
Open source hardware
Open source software
FRICTIONLESS MANUFACTURING
PHYSICAL WORLD AS A SERVICE
Uber
Smart grid
Everything is networked
Rise of the renter economy
American manufacturing revival
Industrial 3D printing
Drop-shipping
Sustainability
Personalized manufacturing
Inexpensive, high-quality CNC tools
Shenzhen economy
Distributed manufacturing
Automation available to anyone
Nest thermostat
Baxter robot
Connected (learning) devices more powerful than individual devices
Instrumented/sensored devices
Web-scale intelligence available everywhere
SOFTWARE ABOVE THE LEVEL OF A SINGLE DEVICE
SOFTWARE REPLACES PHYSICAL COMPLEXITY
OnStar
Google driverless car
Anything can be instrumented, measured, optimized
Security: changes in shape of attack surface
GE
EVERY COMPANY IS A SOFTWARE COMPANY
Web-scale data drawn from the physical world
Web services in control of physical devices
Smart Things
Software abstraction for physical devices
Bot &amp; Dolly
APIS FOR THE PHYSICAL WORLD
Hardware improves over time through software updates
Designing for blend of software and hardware
Machine learning replaces interfaces
DESIGN BEYOND THE SCREEN

Figure 1: Disrupting economies of scale map

Perhaps one of the biggest changes is that you no longer need to be a Microsoft or an Apple to bring a smart product to the market. Companies of all sizes, even individuals such as me are now being given the power to build and produce devices and bring them to market. While there obviously exists a skill and capability gap in what an individual is able to do in contrast to a company with teams and resources to dedicate to the job, the limitation is no longer with tools and knowledge, which is opening up the potential for individuals to acquire the missing components as they go along. Just as Amazon Web Services or Digital Ocean allow you to launch new software at very low up-front cost, the freely available open source knowledge in both hardware and software coupled with the coding and manufacturing capabilities enables entrepreneurs to prototype a product and bring it to other users without depending on an initial investment ranging into the hundreds of thousands of pounds. All you really need is an idea, a proof of concept, and a successful crowdfunding campaign, such as those on the websites Kickstarter or Indiegogo. There have been numerous success stories in this regard.

Just as the internet allows indie game developers to make and sell their own games easily, new manufacturing techniques allow innovators to create and market their own physical products. This is a qualitative change that reflects on the changes we have seen in the past decade when it comes to software. Now, with hardware, you can take risks, be creative, and experiment at a relatively nominal cost.

The Pebble smartwatch is one such success story. The one very interesting aspect to note here is how they were able to stand in the face of competition. Initially, Pebble had a $100,000 Kickstarter fundraising goal. Compared to the investment that typically goes into the R&D for such a device alone, this was a relatively small funding goal. A team of upstarts with absolutely no manufacturing capabilities managed to raise capital, strategize, create and release a full-fledged IoT product, and beat Apple, one of the world's largest, and the world's most valuable company, to market by more than a year. Pebble spent vastly less than Apple in the process. [7]

As with software, one-person teams are being given many of the same tools and access to knowledge as the incumbents. During the course of my project, I have had exposure and access to, and used many of the same microprocessors, tools and software packages that are used in the industry. In this manner, the economies of scale in hardware are well on their way to being abolished.

2.4 Burgeoning Recognition and Creating Value for Economy and Society

The Internet of Things is not limited to small start-ups or individuals. The tech giants of the world have also taken note and have shown active interest in the field. In conjunction with car manufacturers, companies such as Google and Apple are racing towards the Next Big Thing by creating app platforms for cars such as CarPlay and Android Auto, that will decouple the development cycles for entertainment, navigation, and connectivity from the much longer development cycles for cars, bringing with them network connectivity to a domain that did not enjoy it before, along with a sense of refinement and interoperability between cars and their devices that previously was either unsatisfactory, or did not exist at all.

This is an example of how value is perceived in the car market is shifting drastically. Where does the true value of a car lie? Is it in the doors? The material the body is made of? Is it concentrated in the frame, or the engine? Increasingly, the value is diffuse: the value is being delivered in the form of the network connectivity and smart capabilities that are tacked on to the top of a regular vehicle. Building an open API onto the car's data bus which taps into the sensors spread around the car firmly recognises that reality. A prime example of this is how Tesla Motors builds network functionality into their cars. As discussed above, products are no longer "frozen" in the state in which they are shipped due to their nature of belonging to the IoT domain. They are able to continue to deliver value in the field years after they are first created and purchased since they belong to a platform. As product lifecycles become ever shorter, especially in the consumer market (unlike enterprises who are slower to adopt the latest and greatest and accept change), automakers have a new opportunity to respond to market demand with much more flexibility. As Tesla Motors' software taps deep into integration with the drive train, Tesla was able to adjust the suspensions on every one of its sedans through an over-the-air software update in late 2013 [8], and a month later adjusted the cars' chargers the same way [9]. Not only did car owners not need to bring their cars into the dealership to be fixed, Tesla managed to save costs while simply pushing the update out over the internet to everyone affected at once.

As Alex Brisbourne of Kore notes, this may be the best example of the Internet of Things yet. Rather than having the tiresome task of an unplanned trip to the dealer put upon them, Tesla owners can continue with their lives while the car "fixes itself." [9]

This also creates sustainable value. As the article notes:

"The real challenge for the "consumer-facing" Internet of Things is that applications will always be fighting for a tightly squeezed share of disposable consumer income. The value proposition must provide tangible worth over time. For Tesla, the prospect of getting one's vehicle fixed without "taking it to the shop" is instantly meaningful for the would-be buyer – and the differentiator only becomes stronger over time as proud new Tesla owners laugh while their friends must continue heading to the dealer to iron out typical bug fixes for a new car. In other words, there is immediate monetary value and technology expands brand differentiation. As for Tesla dealers, they must be delighted to avoid having to make such needling repairs to irritated customers – they can merely enjoy the positive PR halo effect that a paradigm changing event like this creates for the brand – and therefore their businesses."

There are other ways in which IoT developments in the field are likely to change the way businesses and society function in the foreseeable future. Cars may pre-emptively sense when they need to be maintained and taken in for servicing. No longer will this be by virtue of the distance driven and the memory of the driver, potentially reducing the number of incidents on the road. The car will automatically sense that it needs maintenance, send a notification to the registered dealer or mechanic, cross-reference this with your calendar, set up an appointment, and automatically ask for your approval, at your convenience, all in one tap on your smartphone. For what may essentially be a $5 microprocessor and some intelligent software doing the job, this system delivers incredible value and return on investment once it has crossed the initial investment, design, and deployment stage, elegantly foreshadowing the changes to come.

The shift into a blend of machine and software working in tandem can and will change business models in the future. It simplifies inventory, since products typically ship with certain specification/accessories installed, which are then enabled or disabled by the seller. There will be no need to stock dozens of different accessory packages; a product can become whatever it is supposed to be when and after it has been purchased. The only difference is what is communicated over the internet and what is sent to a screen to the user, which is a change in the software setting.

Internet connectivity and over-the-air updates are not just for big products such as cars. Smart refrigerators can download health plans that pertain to a particular user's health tracking routine, informing the user to buy more meat if they need more protein in their diet, having detected the levels of stock inside the fridge. New machine-learning algorithms can be applied to old products to effectively teach an old dog new tricks. Nest's Protect fire alarm could initially be silenced with a wave of the hand. This feature made it possible to unintentionally disable the device, unbeknownst to the user. Rather than recalling the units, they were able to issue an automatic update that disabled this feature. Consumers did not need to actively participate in the process, which is as it should be. Every device in the field was updated, not just the ones whose owners were willing to take them down and ship them back to the company for the update.

2.5 The Role of Software in Innovation and Solutions to Problems

Hardware that didn't have any association with software is now beginning to meld with it. Consider a large, industrial company that makes vending machines. Their machines are shipped as tons of metal and plastic. These are machines that are commonplace and make up the basic infrastructure of our lives. Their physical design, their hardware, have been refined over many decades, and in many respects, they are superbly optimised. Many of these devices have reached a saturation level in their basic design and philosophy, and have not been modified since decades. The cost of further improvements to their materials and physical design can be not only prohibitive, but considered unnecessary as they "do the job". Now, with IoT, intelligence in code that is routine in software and on the internet is just starting to touch these machines. Consider that those innovations can be retroactively applied to existing machines on the market with smaller optimisations and inexpensive changes to integrate with existing microcontrollers on-board. This has been the focus of my investigation in this project, and it offers an entirely new set of improvements as well as a less expensive approach to existing features.

What does this mean for innovation? Considered as simple hunks of metal and LEDs/bulbs, it is difficult to visualise significant improvements to devices such as street lamps. However, consider that the streetlamp has been networked. Perhaps it could serve as an access point for the city's free Wi-Fi infrastructure, it could display notifications about traffic conditions, public transport delays, send weather updates, help users navigate with beacons or whatever you want. It could even call the police if it picks up sounds that match a car accident or a human cry for help, bolstered by machine learning algorithms. It is difficult to imagine garbage cans as anything that may be improved. But consider a networked garbage can: it could notify the city council if garbage has been missed during pickup. If it is empty, the collection truck may be notified, and need not take the time to check the bin. This demonstrates that basic hardware need not be changed, neither does the class of product need to be an entirely new hybrid device (such as consumer drones) for the benefits of IoT to reach everyday "dumb" devices.

These ideas require looking at a vending machine, or a garbage can, or a streetlamp as a node in a software system. In turn, the companies that make them need to think of themselves not as companies that shape steel and mould plastic, but companies that are involved in software. That organisational change is happening, and corporations that don't get it will be left behind – most notably, the change has been recognised by automaker Ford. [10] When asked about his remarks on a spinoff being known as a software company, Ford CEO Mark Fields said "As we go forward, I want us to be known as a manufacturing, a technology, and an information company. Because as our vehicles become a part of the Internet of Things, and as consumers choose to share their data with us, we want to be able to use that data to help make their lives better. And also, create some business models that will help us earn a return. That's where we're heading … Our approach is to first, disrupt ourselves. That's why you see us creating things like Ford Smart Mobility, the LLC, creating FordPass, really thinking differently about this from a consumer-end standpoint. Thinking about experiences, and then how does technology, hardware, and software deliver that, as opposed to the other way around. I think we're really disrupting ourselves. Who knows what Apple is going to do? Our working assumption should be, they're doing a car, it's going to have a great user interface, it's going to be connected seamlessly to the cloud. That motivates us even more as a company, to make sure that we push the company forward in a lot of these different areas."

Just as Apple understood and capitalised on the change that was coming to the phone and device industries, and was able to unseat the incumbents Nokia and BlackBerry, those that understand and go along with this change can be more creative, more novel, and more influential than many of the companies that, today, we conventionally call "tech companies". [11]

2.6 How Sensors, Data and Communication Intersect Problem Solving to Create Value

In a nutshell, The Internet of Things elevates software above the level of one device. In other words, one of the main principles of Web 2.0 is being brought to the material world. Software is working closer than ever to hardware, but it is not tied to a specific piece of hardware. It is not remarkable that a thermostat, a refrigerator, a car, a watch or a pair of shoes is "smart" (in the sense that it has a microcontroller in it that functions as some sort of real-time control and feedback mechanism). True intelligence and value comes from intersection between all the connected devices, and those to the different kinds of software built to understand the raw data. An electric car may start charging at night because software at the level of the grid signals that electricity prices are cheaper at night (a process that is itself kicked off by lots of sensors that are networked together, performing measurements across the city grid).

To take advantage of that kind of intelligence, there are two ways to interact with data. One, the way I have chosen in my project, is to use open standards, widely available programming languages and communication methods that are device and platform agnostic. Data is communicated in a standard format, available for any software developer to take advantage of. The other popular alternate exercised by hardware companies is building products with explicit APIs. Just as phone manufacturers built operating systems and opened up APIs to allow developers to find novel ways of using them and building ecosystems around their products, automatically giving you value and reason to own those devices, hardware makers are opening up APIs in smart devices too. Philips has published APIs that let developers find creative uses for their Hue lightbulbs and turn them into platforms. [12] The result is a kind of modular comparative advantage: the best appliance maker can build coffee makers, and software engineers can find ways to integrate them with the best software.

Google Maps provides an interesting example of when the same software runs on and takes advantage of multiple devices. Although Google Maps appears as though it is putting a simple map on your smartphone screen and plotting your location on the screen, the service achieves remarkably accurate real-time traffic data; these don't come from aerial and satellite reports – they come from myriad Android phones, each of which has location services reporting to Google's servers. If the phones are moving, traffic is flowing. If the phones on a particular route are moving particularly slowly, that is an indication of congestion on the road. Your phone isn't just displaying traffic conditions, it's also reporting traffic conditions, as part of a much larger system.

Whether the context is GPS and maps, home lighting, fitness trackers or others, they all share the common denominator of software systems that can potentially run across many devices. Smart light bulbs are an interesting case since they may exist by themselves, but more bulbs together become much more powerful in a system with other bulbs or devices. Individual light bulbs may be controlled by a program that also reads motion detectors scattered throughout the house, turning on and off lights as you move around the house.

Many more systems such as this have the possibility to exist. Cars, microwaves, air conditioners, and other appliances that consume a lot of power can communicate with a smart electrical grid and optimise their use of power. The preference to run only when electricity is at its cheapest can be programmed into them. Once you have this ability, it makes sense for the electrical company to give you discounted rates for moving your power consumption to off-peak hours. Why dry your clothes during the day when electrical use is at a premium? Controlling the operation of an appliance manually to make use of the off-peak hours is at best a pain, and probably impractical. However, sending commands over the internet to a smart dryer (or programming it to be independent entirely) can take care of that for you by participating in a network of devices that works above the level of any one node in that network.

3 Approach

3.1 Overview

As a basis for my deliverable IoT module and implementation of the solution, I evaluated the system in parts. It is tempting to consider the reduction in friction for building physical products as a technological revolution, but there is nothing in the manufacturing space that is really new on its own. What is different is that the tools are much more accessible. Few people could evaluate microcontrollers and create a design on old programming tools; they were highly specialized, and had difficult user interfaces that required lengthy training. Now that the hobbyist world has been given new life with the IoT space and the proliferation of microcontrollers into public thought, manufacturers have to make tools that are very capable, easy to use, and inexpensive. Anyone has the resources available to begin to understand the basics of how microcontrollers work with tools such as the commercially available Raspberry Pi, and many of the programming tools/IDEs that allow programming in the languages that support downloading code to a microcontroller are free. This provides an exposure point with a lowered barrier to entry into the field for the average consumer, hobbyist, or student. Tool vendors are learning that consumer grade is the new professional grade; professionals now want the same ease of use that consumers required.

That said, we may never eliminate friction entirely. The real world has constraints that you just do not find in software development. Still, the kind of software that can understand and deal with real-world constraints is becoming available to individual prototypers. Setting up coding environments that can map out libraries, identify conflicts, and find ways to work around them are now a matter of running an installer on your laptop.

While expertise is becoming available in modules, rather than in rigid take-it-or-leave-it packages (in itself an amazing change to the landscape of personal electronics and consumer devices), hobbyists and students will perhaps not start fabricating their own processors or complete products with expertise any time soon. True expertise is as important as ever, but the ability to pick and choose the depth to which you can explore one facet of a product (such as the code) while leaving the others to acquired knowledge, to be able to focus on the bigger picture is a liberating change.

3.2 Competing Standards for IoT Connectivity

A true IoT network not just some set of given devices that know how communicate with each other. During the infancy of the internet, where the concept of protocols that were both standard and open was first established, we gained that expertise back then so that anyone could construct hardware that could interoperate with everything else.

We take this for granted now, but the world wide web would be a completely different place if we needed one browser to use social media, another to check email, and another to watch movies on Netflix. Those browsers may not even have run on the same hardware. In the past few decades, our devices used to function in this manner – each had a specific purpose and that purpose only, and could not be used for anything else. Our TVs were only TVs, meant to receive, interpret and display one signal, and only that. Standards did exist, but the standards were so specific that they did not end up standardising much. One box could never do what another box did.

Standard protocols turn devices into platforms. As the networking protocols that eventually shaped the internet into the place we see today achieved dominance, we started to understand that there was no special need to have specialised devices, and that using one browser that could understand many protocols and do many tasks was going to be beneficial to both content creators and consumers.

IoT is in a state of flux in this regard right now, where devices are growing faster than the standards being implemented for them.

If you have a smartphone that detects and sends proximity to your home to a thermostat, would you want it to turn up the heating in your home just as you arrive? Do you want lights that adjust themselves to the music that is playing? Do you have a vending machine that you would like to send you a report of sales made that day as you approach it? A device could sense contextual information and transmit it back to an application on a cloud server. However, intriguing as the opportunities may be, none of this can happen meaningfully without standards. We have many devices in our homes that we would like to see brought into the IoT domain, but we would not like to be locked into a single vendor and buy all our devices from them just so that they may interoperate. We would want to take advantage of other deals, or get the best of all worlds from each manufacturer that is the best at creating a certain class of device.

In a world where every manufacturer rolls their own communications and proprietary protocols and hopes that the winner will take all, there will inevitably be a standards war and a mess in the space: incompatible devices from different vendors that do not work together. There is no "win" here; everyone loses.

Currently, in the market, the popular communication standards for smart devices are Wi-Fi, ZigBee, and Z-Wave.

The ZigBee website explains [13] ZigBee as a "wireless language that everyday devices use to connect to one another" and that "ZigBee standardizes everything from basic communication to how a product operates. Products with the ZigBee logo work together seamlessly, even if they're from different companies." Z-Wave's website [14] also defines the standard as "a wireless technology that lets smart devices talk to one another. Household products, like lights, door locks and thermostats are made "smart" when Z-Wave connectivity is added inside the product's design, giving them the capability to communicate and perform the desired function."

While both standards claim to offer advantages and a wide portfolio of products that are compatible with them, for the purpose of my project I have decided to use just Wi-Fi as the standard. This is because not only do the other standards have rigid rules on use, they also require licensing and an initial investment to get a kit ranging in the hundreds of pounds to be able to start using the technology. Z-wave's FAQ states "With a range of affordable options, Z-Wave can help you create a smart home a budget. The cost really depends on how many products you choose to start with. For example, a basic starter kit with a hub and a few devices can cost a few hundred dollars. Many people start off small with a limited number of products and then add on as they become familiar with their smart home and want more functionality."

For the purpose of my deliverable, Wi-Fi is the most suitable since:

- Wi-Fi is completely open.
- It does not require joining an alliance or paying a licensing fee.
- It is widely available.
- Wi-Fi is already built-in in nearly all consumer devices we see today, regardless of manufacturer.
- It is a standard that is shared by various classes of devices in the IoT space already.

- It is a reasonable assumption that any location or home will already contain a Wi-Fi infrastructure to use, as opposed to the hit-and-miss existence of a ZigBee or Z-Wave product.
- Simplicity is key and Wi-Fi simplifies connection methods and tools used to build the project.
- I do not need to gain an understanding of a completely new and unfamiliar protocol in the time frame I have.

3.3 Hardware and Software Used, Justifications for Use

Several microcontrollers are available for students and hobbyists to begin experimenting with creating IoT devices. Of those most popular are the Raspberry Pi, Arduino, and Intel Edison. In the right hands, and with the right consideration of the product limitations, these are excellent products. However, in recognition that an IoT device needs to work reliably, in real-time, I have chosen instead not to opt for these commercial microcontroller boards and get the Freescale Freedom KL25Z evaluation board from the vendor Freescale. The Freedom KL25Z is an ultra-low-cost development platform built on an ARM® Cortex®-M0+ processor. [15] The choice of this board is for reasons including but not limited to:

1. There is no guarantee that the Raspberry Pi you purchase at a later date will be the same as the one you bought six months ago. The Raspberry Pi foundation can and do make some alterations to their designs and specifications with little to no notice. At least one of which has been visible to user software (the GPIO – general purpose input output pins reassignment on revision 2 of an earlier model) and at least one of which has required new.
2. According to research conducted and opinions of companies in the industry, such as Bird Electronics Pvt. Ltd., the Pi and its equivalents are not designed to be and perhaps never will be an embedded control system. Designing them into an industrial system is considered at best negligent, and at worst, dangerous.
3. The form factor is not under our control.
4. Real time performance (which is important for IoT applications) is not guaranteed here. Even a microcontroller suited for embedded applications that costs 50p can do great real time control, with a guaranteed response time of 10 ms and better.
5. While a board such as the Raspberry Pi costs £30+ for a capable board minus the accessories, a sample development board from a microcontroller vendor such as Freescale costs only as much as £10-12 depending on where it is purchased, which has cost implications on the end-user.

6. Power consumption on the standalone ARM processor boards is very low – they draw only 10-15 mA of current. An IoT device may be operated using batteries.
7. If the Pi (or other board) becomes obsolete, or undergoes design revisions that break our application, we have no control.
8. Our selected chips from Freescale have an assurance of support for the same design for up to 10 years.

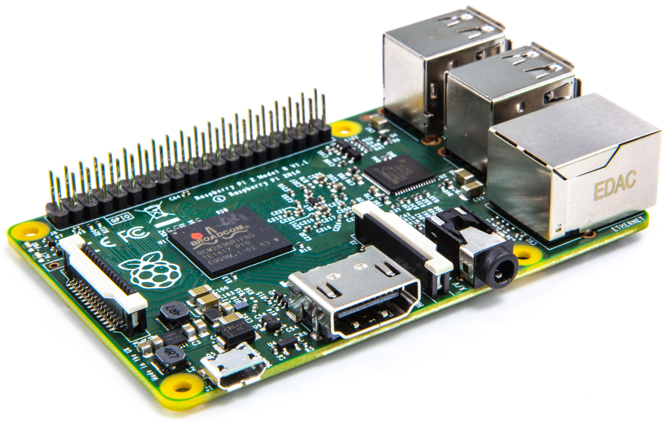


Figure 2: Raspberry Pi

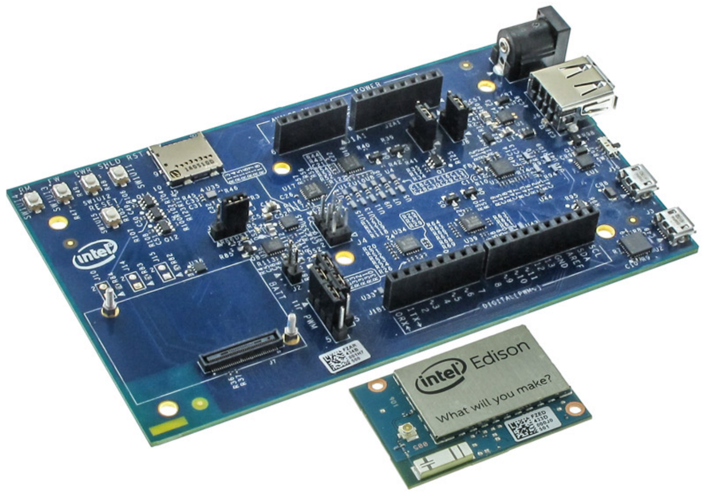


Figure 3: Intel Edison

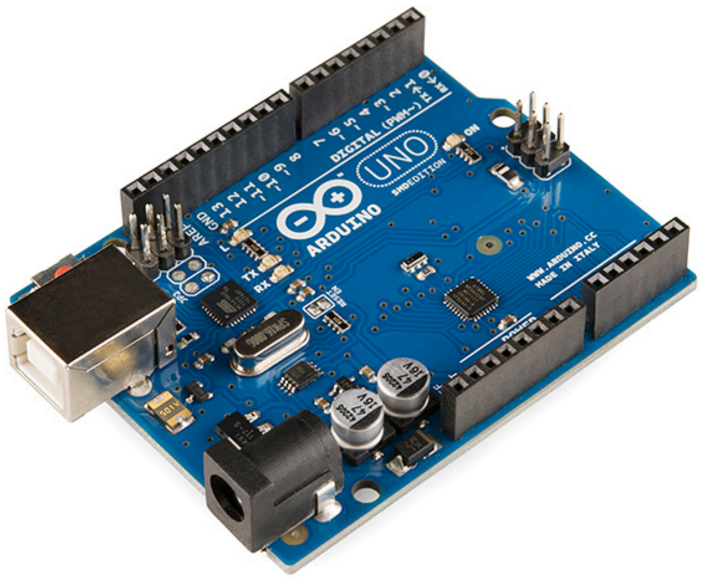


Figure 4: Arduino Uno

For the software, I will be using the KEIL µVision IDE and programming in C. This is a very capable software package that is used in the industry for professional applications.

1. KEIL is an ARM company, belonging to the same parent organisation that holds patents on the designs of the microcontrollers being used.
2. C code is rock-solid and the programming concepts can be adapted to.
3. The device or the language does not need updates to perform its function.
4. Real-time response time is guaranteed.
5. The hardware resources required are very small 1K of RAM and less than 4 KB of flash memory can be used to produce a meaningful system.
6. The size of the microcontrollers being programmed can be as small as 5 mm x 5 mm x 1 mm.

For Wi-Fi connectivity, I have evaluated and chosen to use the Espressif ESP8266-01 Wi-Fi [16] module. This was paired with a USB-Serial converter, a device that allowed the Wi-Fi module to be connected to a PC so that it could be configured to connect to an access point.

Not only is this module very popular with hobbyists and commercial product designers alike, it has been used in commercial applications before, and is an inexpensive £4 module that supports serial communications, such as from a terminal on a PC. It can be instructed using commands from a terminal, or from a microcontroller sending the same commands over UART. A UART (Universal Asynchronous Receiver/Transmitter) is the microcontroller with programming that controls a computer's interface to its attached serial devices.

Other justifications:

Why use a standalone microcontroller board with UART? Given that the aim is to convert a target device to a network-connected one, can you expect a UART connection available to instruct the Wi-Fi module?

Every microcontroller of this kind has UART available. I am working on the assumption that it is a reasonable expectation to create a module that interfaces with a target device using UART. Microcontrollers that perform this function are cheap and may even include multiple UART connections. For the equivalent of £2, you can get a chip in the market that has multiple UARTs. If more connections need to be made, you can upgrade to the next processor with minimal effort because many processors from the same vendor such as Freescale are pin-compatible. Even if the target device isn’t based on/doesn’t have a microcontroller, one may be implemented by buying one for a reasonable price.

Why use Wi-Fi and not Bluetooth, an equally prolific standard? How is the Wi-Fi module instructed?

Bluetooth adds a layer of complexity to the project that is unnecessary. Bluetooth radios are run essentially on the same frequency as Wi-Fi. While implementation is different, it is at the end of the day a form of RF, and most of our wireless devices already run at 2.4 GHz. Essentially, communicating data out using Bluetooth will require a middleman to be able to send this data out to the internet, which amounts to adding a layer of abstraction to the end use. Connecting to a Wi-Fi access point instantly gives us access to not only a network connection, but should the access point have an internet connection, it allows the device to directly connect to the internet and send data in this matter. The ESP8266 Wi-Fi module is instructed using the AT commandset, the reference for which is standardised, used by several modules on the market, and is easily available online. This is a set of standardised commands that allow us to instruct the Wi-Fi module by sending it command-line commands (in the format AT+_____) to perform functions such as connecting to an access point or sending data to a specified server at an IP address. [17]

This will form the basis of sending communications to the central server.

4. Implementation

4.1 Target Scenario and Specifications

Having established the meaning of the Internet of Things Era, the implications for consumers and the changes it will bring to businesses, societies, individuals and economies, I aim to demonstrate the benefits of providing network capabilities to a device in a simple, hands-on scenario. In my example, I will explore the use-case of a vending machine, where the Freedom board will serve as a stand-in for the microcontroller on the device. These machines have worked on the principle of coin slots, button presses on a keyboard and physical mechanisms for dispensing for a long time. There is perhaps no need for these to change for basic setups where simple items such as water or food need to be dispensed. However, the benefits begin to emerge when the same machines are deployed in different contexts. Drinking water issues are significant in developing countries:

- Water filtration equipment is complex and expensive.
- It is often required in remote places.
- Maintenance is difficult in remote places.
- Manual operators are sometimes required to sell water at the site and collect money from the buyers, instead of fully-automated vending/dispensing machines.
- There is no firm accounting of how much water was sold, and how much revenue was generated.
- Pilferage of funds collected is an issue.

- If the equipment goes defective, service engineers take a long time to service a machine because they don't know what spares are required.
- These machines exist but perform poorly because central monitoring is not possible.

A solution to these problems identified above is giving the device network capabilities, making very minor modifications to the electronic control system at a low cost to implement network connectivity on the target device. Following are the specifications for the proof-of concept device:

- Set up a central server with a user interface to receive data from the Wi-Fi module.
- Use the AT command set to instruct the ESP8266 Wi-Fi module from a PC to connect to a certain access point.
- Program the ARM microcontroller board to interpret an event (pressing a button) and connect the ESP8266 to the microcontroller board with the headers.
- When the event occurs, send a command to the Wi-Fi module to log a corresponding value for a parameter to the central server.
- Demonstrate a visualisation of the logged parameters and their values and how this can help solve these problems and provide insight to the operation of the machine.

4.2 System Diagram and Description

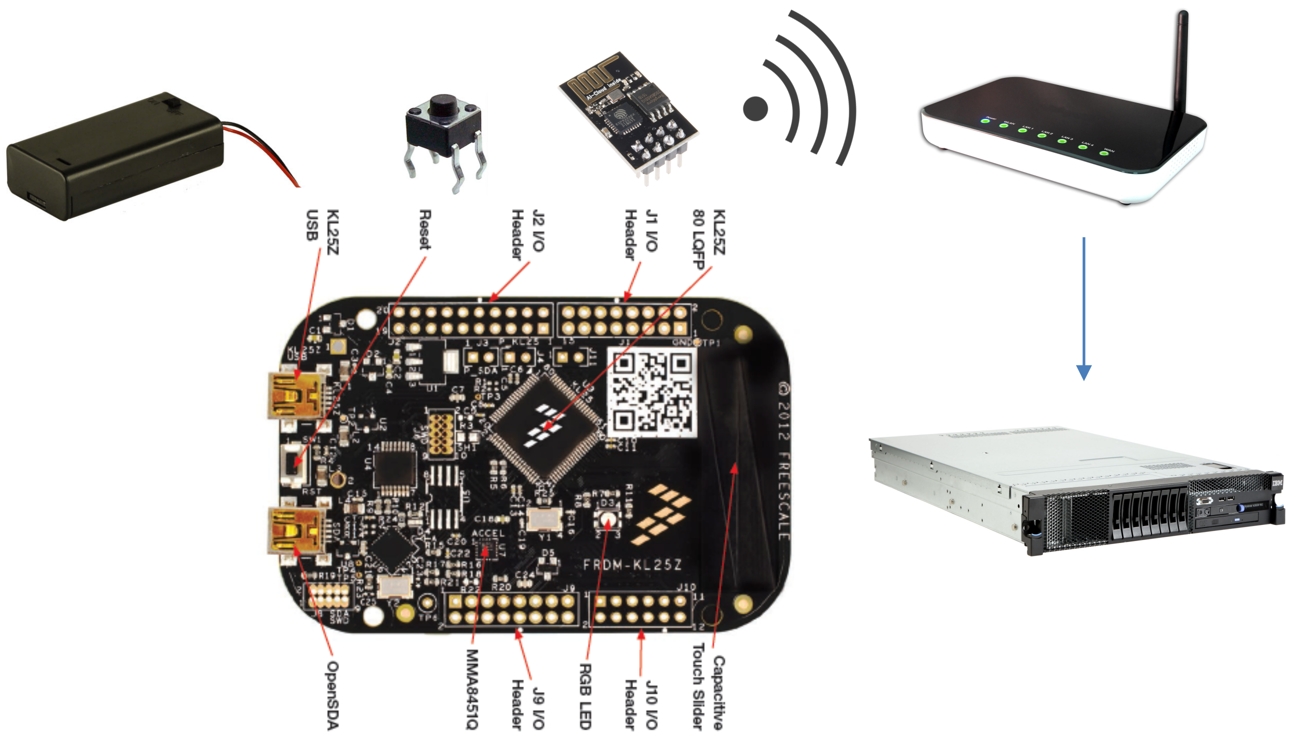


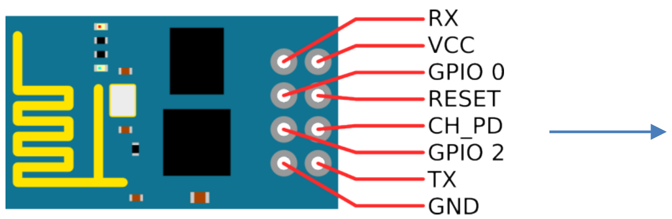
Figure 5: System overview

A small AA battery pack/power supply is used to power this system. The Freedom KL25Z board serves as a stand-in for the target device, and has a switch connected to one of the port lines of the processor, Port A4 (the switch is connected between the port line and ground). This is one of the pins on the microcontroller. When the switch is pressed, it produces an “interrupt” to the processor. This indicates to the processor that it must halt and execute a specific piece of code, which will be seen in the code section.

The ESP8266 Wi-Fi module is connected to the Freedom board via its headers. The Wi-Fi module has been pre-programmed to connect to a specific access point when the system is on. The wireless access point acts as the gateway to the internet where it may reach the central server, which is ready to receive the data.

4.3 Implementation Steps

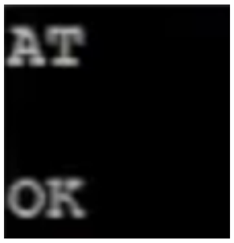
At first, the ESP8266 is connected to the USB-Serial converter, where GND is connected to GND, and the transmit (Tx) and receive (Rx) pins between the Wi-Fi module and the converter are crossed (Tx connected to Rx and vice versa). A 2xAA battery pack is used to give power to the Wi-Fi module on the supply pin, VCC, and the negative connected to GND. This establishes a connection with the module so that it may be plugged into a PC and interacted with using a terminal (command line).



When the module is plugged into the USB port on a PC, a serial (COM) port is allotted to the device. The number of the COM port allotted is verified in Device Manager.

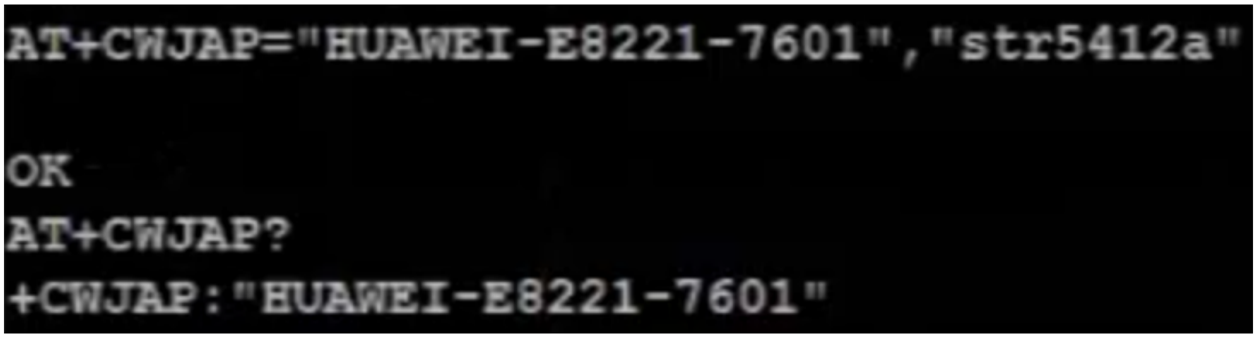
Once verified, a terminal window can be opened to interact with the Wi-Fi module. In my scenario, I used PuTTY, [18] a free terminal application for Windows to open a serial connection on the specified COM port.

The Wi-Fi module begins a session with "AT" and OK" indicating that it is ready to receive AT commands.



```
AT

OK
```

To connect the Wi-Fi module to a nearby access point, the AT command used is "AT+CWJAP=<ap name>,<ap password>", for "join access point". The name and password of the access point are specified. Once connected, the module responds with an "OK". To verify that the module has been connected, a question mark may be included in the same command. A question mark indicates that the command is a query. The module sends a response confirming the name of the access point it is currently connected to.



```
AT+CWJAP="HUAWEI-E8221-7601","str5412a"

OK
AT+CWJAP?
+CWJAP:"HUAWEI-E8221-7601"
```

The module is now configured. Any data to be sent over the internet may be done using the access point it is connected to.

The next step was to set up a Linux webserver. I used a VPS server from Amazon Web Services for this. AWS is one of the most popular, cost-effective and reliable providers of hosting services.

To log into the server, I used the server's public IP address and the password for the "root" user's account. An SSH session with the server could be established using the command:

```
ssh root@<SERVER_IP_ADDRESS>
```

Where the server IP address is 188.166.247.116. With the first time logging in, the root password was subsequently changed.

The next step was to set up an alternative user account with a reduced scope of influence for day-to-day work. Once you are logged in as root, the new user account to be used henceforth was created. A user called “demo” was created using the terminal command “adduser demo”.

The next step was to use the command “gpasswd -a demo sudo” to set up what is known as a "super user" or root privileges for our normal account. This allowed run commands with administrative privileges by putting the word sudo before each command. By default, on Ubuntu 14.04, users who belong to the "sudo" group are allowed to use the sudo command.

The next step was to install LAMP on the server.

LAMP stack is a group of open source software used to get web servers up and running. The acronym stands for Linux, Apache, MySQL, and PHP. Since the virtual private server is already running Ubuntu, the Linux part is taken care of.

Step 1: Install Apache. Apache is a free open source software which runs over 50% of the world’s web servers. To install apache, the following terminal commands were used:

```
sudo apt-get update

sudo apt-get install apache2
```

Step 2: Install MySQL. MySQL is a powerful database management system used for organizing and retrieving data. To install MySQL, the following commands were used:

```
sudo apt-get install mysql-server libapache2-mod-auth-mysql php5-mysql
```

Once MySQL was installed, it was activated with the command:

```
sudo mysql_install_db
```

Finished up by running the MySQL set up script:

```
sudo /usr/bin/mysql_secure_installation
```

Step 3: Install PHP. PHP is an open source web scripting language that is widely use to build dynamic web pages.

To install PHP, the following command was used:

sudo apt-get install php5 libapache2-mod-php5 php5-mcrypt

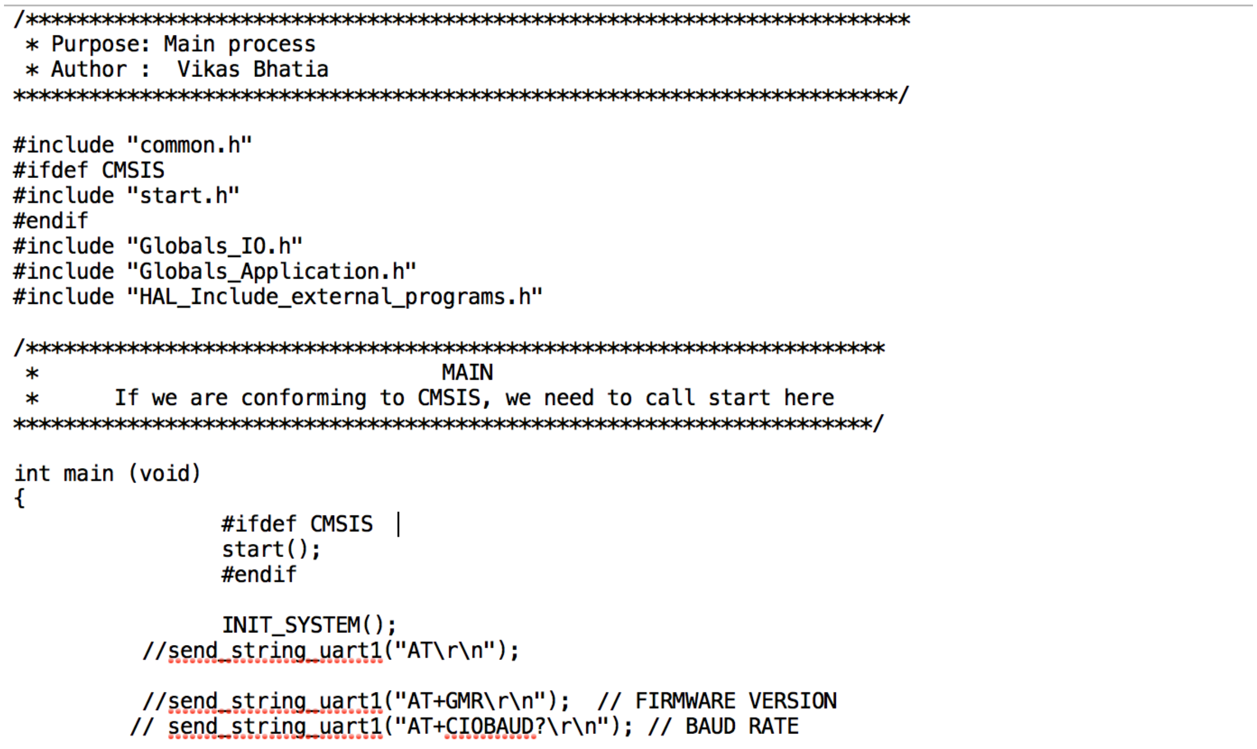
This gave me a functioning web server with LAMP installed. Cloudnix Software Labs assisted during the set up and created a dashboard to receive the data from the Wi-Fi module and graph it; the coding of the dashboard was beyond the scope of my skills and resources.

4.4 Application Code

Bird Electronics Pvt. Ltd. assisted and advised me on writing the application code for this project. Sample code was obtained from Freescale, the vendor of the microcontroller, edited to an extent and passed on by Bird Electronics Pvt. Ltd. to me. Programming a microcontroller in C can be incredibly complex, which is why I found it useful to have a starting template from which to be able to understand and use commands to instruct the Wi-Fi module. This section will outline significant areas of code and explain what the purpose of such code is within the application. Please note that not all files are described within: the microcontroller's sample code includes several other project files including common files, drivers for the CPU etc. that are required to establish communication with the compiler/PC and the board. All files are provided in the archive.

4.4.3 Main File

The main file is executed when the processor is powered on. Here, it initialises the system by calling INIT_SYSTEM.



```
/*****************************************************************************
 * Purpose: Main process
 * Author :  Vikas Bhatia
*****************************************************************************/

#include "common.h"
#ifdef CMSIS
#include "start.h"
#endif
#include "Globals_IO.h"
#include "Globals_Application.h"
#include "HAL_Include_external_programs.h"

/***************************************************************************
 *                              MAIN
 *      If we are conforming to CMSIS, we need to call start here
***************************************************************************/

int main (void)
{
                #ifdef CMSIS |
                start();
                #endif

                INIT_SYSTEM();
          //send_string_uart1("AT\r\n");

          //send_string_uart1("AT+GMR\r\n");  // FIRMWARE VERSION
         // send_string_uart1("AT+CIOBAUD?\r\n"); // BAUD RATE
```

```
             //send_string_uart1("AT+CWMODE?\r\n"); //1 = client, 2= access point, 3 = dual
            // send_string_uart1("AT+CWLAP\r\n");  //List all surrounding access points
            //send_string_uart1("AT+CWJAP=\"AndroidAP\",\"abcd2605\"\r\n"); // join access point
            //Delay(10000);
             //send_string_uart1("AT+CWJAP?\r\n");
            //send_string_uart1("AT+CIPMUX=1\r\n"); // setup to multiple connections

            //send_string_uart1("AT+CIPSERVER=1,333\r\n"); // setup to multiple connections
            //Delay(5000);
            //send_string_uart1("AT+CIFSR\r\n"); // get ip address


                 while(1)
                 {

                         loop_count++;

/*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*/
                         curTicks = msTicks;
      while(abs(msTicks-curTicks) < 2 )
                         {
                                 // Do nothing
                         }
/*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*/
                         curTicks = msTicks;

                         Calculate_ASM_Status();

                         while(abs(msTicks-curTicks) < 2 )
                         {

                                 // Do nothing
                         }

/*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*/

                 }
}
/*****************************************************************************
                                        END OF FILE
*****************************************************************************/
```

The main file was used for debugging while I tested sending different AT commands to the Wi-Fi module. Sending characters over UART to the Wi-Fi module behaves as if the module had received the command from a terminal. The module simply receives characters from the processor instead of characters typed into a PC. While these commands are commented-out (disabled), in other situations, the main file only initialises the system and calls the function "Calculate_ASM_Status".

4.4.4 Calculate_ASM_Status

This file established two states the system can exist in, and different actions it must perform depending on the state in which it exists. These are currently unused

```
**********************************************************************/
#include "common.h"
#include "Globals_IO_External.h"
#include "Globals_Application_External.h"
#include "HAL_Include_external_programs.h"

/****************************************************************************
 *                   ASM_Status = Default None of above
****************************************************************************/

void ASM_Default(void)
{

                 Char_UART_Read =       0x00;
}

/****************************************************************************
 *                            THE STATE MACHINE
 *            Switch to program for respective ASM_Status
****************************************************************************/

void Calculate_ASM_Status(void)
{

        switch (ASM_Status)

        {

                 case 0 :                        //ASM_Status = INIT = 0
                 {
```

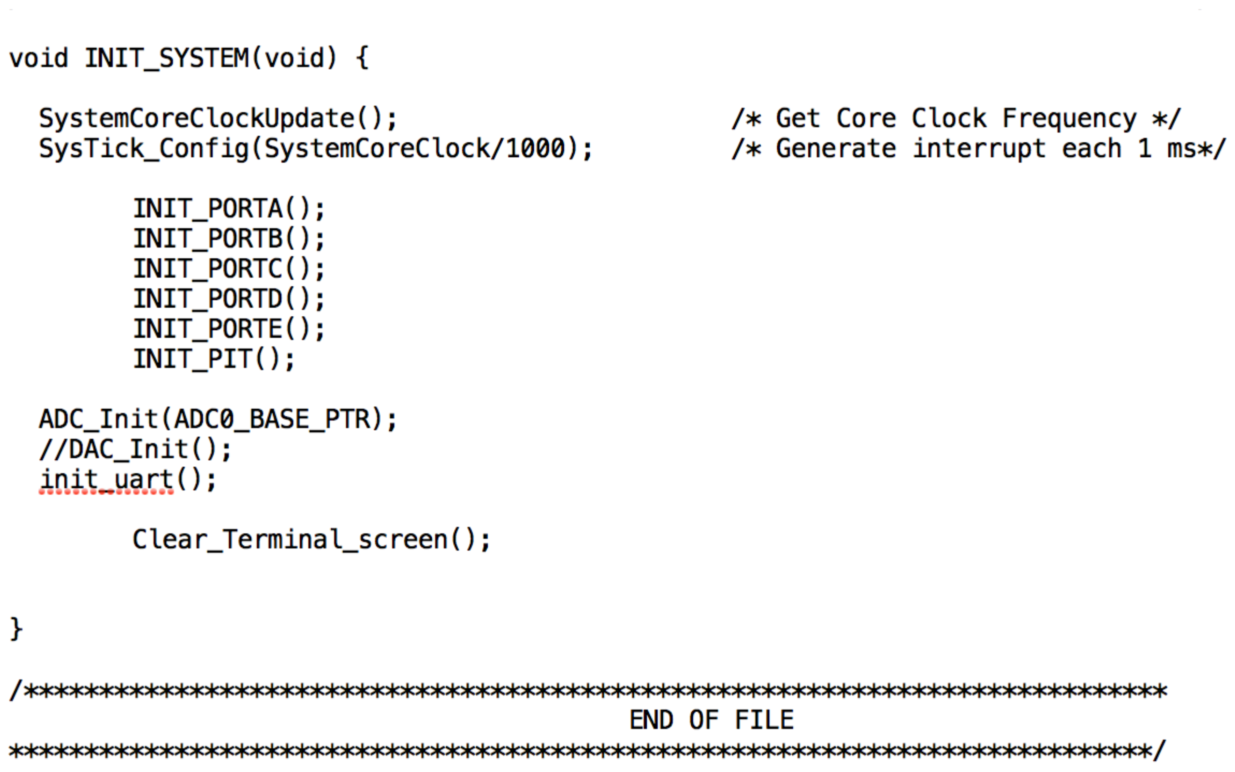
```
        switch (ASM_Status)

        {

                  case 0 :                        //ASM_Status = INIT = 0
                  {

                                  ASM_0();
                                  break;

                  }

                  case 200 :                      //ASM_Status = STANDBY= 200
                  {

                                 ASM_200();
                                 break;

                  }

                  default :
                  {
                                 ASM_Default();
                                 break;
                  }


        }

}

/*****************************************************************************
                                          END OF FILE
*****************************************************************************/
```

4.4.5 HAL_Init_system

This initialises all the port lines (pins) on the processor, making them available for use, initialises the UART to be able to send commands to the Wi-Fi module.

```
/**************************************************************************
 * File:         HAL_Init_System.c
 * Purpose:      Initialize PortA, Initialize PortB, Initialize PortC,
 *          Initialize PortD,Initialize PortE,Initialize ADC,
 *          Initialize DAC,Initialize UART, Initialize PIT,
 *          SysTick_Handler ISR, Clear Terminal Screen
 *          Delay Program
 **************************************************************************/

#include "common.h"
#include "Globals_IO_External.h"
#include "Globals_Application_External.h"

#include "HAL_Include_external_programs.h"

/***************************************************************************/
//                                INIT_SYSTEM
/***************************************************************************/

/***************************************************************************/
/*                               SysTick_Handler                          */
/***************************************************************************/

void SysTick_Handler(void) {
  msTicks++;    /* increment counter necessary in Delay() */


        if(msTicks%10 == 0)
        {
          TenmsTicks++;

  }

        if(msTicks%400 == 0)
        {
          Counter_500ms= 1;
  }
```

```
        if(TenmsTicks >99)
        {
         TenmsTicks = 0;
  }


  /* These two commands to be given to reset watchdog timer in this sequence */

        SIM_SRVCOP = 0x55;
        SIM_SRVCOP = 0xAA;

}


/*******************************************************************************/
/*        delays number of tick Systicks (happens every 1 ms)                  */
/*******************************************************************************/

void Delay (uint32_t dlyTicks) {
  uint32_t curTicks;

  curTicks = msTicks;
    while ((msTicks - curTicks) < dlyTicks);
}



/*******************************************************************************/
//                                INIT SYSTEM
/*******************************************************************************/
```



```
void INIT_SYSTEM(void) {

  SystemCoreClockUpdate();                      /* Get Core Clock Frequency */
  SysTick_Config(SystemCoreClock/1000);         /* Generate interrupt each 1 ms*/

        INIT_PORTA();
        INIT_PORTB();
        INIT_PORTC();
        INIT_PORTD();
        INIT_PORTE();
        INIT_PIT();

  ADC_Init(ADC0_BASE_PTR);
  //DAC_Init();
  init_uart();

        Clear_Terminal_screen();


}

/****************************************************************************
                                         END OF FILE
****************************************************************************/
```

4.4.6 Execute_ASM_0

When the main file is called, the processor starts in state 0, where it allows for a 95 millisecond delay for the system to initialise. When the delay is finished, it moves to state 200, where it waits in standby for an “interrupt” (a button press).

```
#include "common.h"
#include "Globals_IO_External.h"
#include "Globals_Application_External.h"
#include "HAL_Include_external_programs.h"

/*****************************************************************************
*                          ASM_Status = INIT = 0
*  NV is preset if : The key does not match from location 28 and 29
*  Or operator presses and holds Dispenser2, Dispenser3 and Dispenser4
*  then switches on power
*****************************************************************************/

void ASM_0(void)
{
                if (msTicks > 95)
                {
                                /*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                          +                    Go to standby at power on
                          ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*/

                                ASM_Status = 200;

                                msTicks = 0;

                }
}

/*****************************************************************************
                                        END OF FILE
*****************************************************************************/
```

4.4.7 Execute_ASM_200

The processor is in standby and does nothing here.

```
#include "common.h"
#include "Globals_IO_External.h"
#include "Globals_Application_External.h"
#include "HAL_Include_external_programs.h"


/*****************************************************************************
                             ASM_Status = STANDBY = 200
*****************************************************************************/
void ASM_200(void)
{



}
/*****************************************************************************
                                        END OF FILE
*****************************************************************************/
```

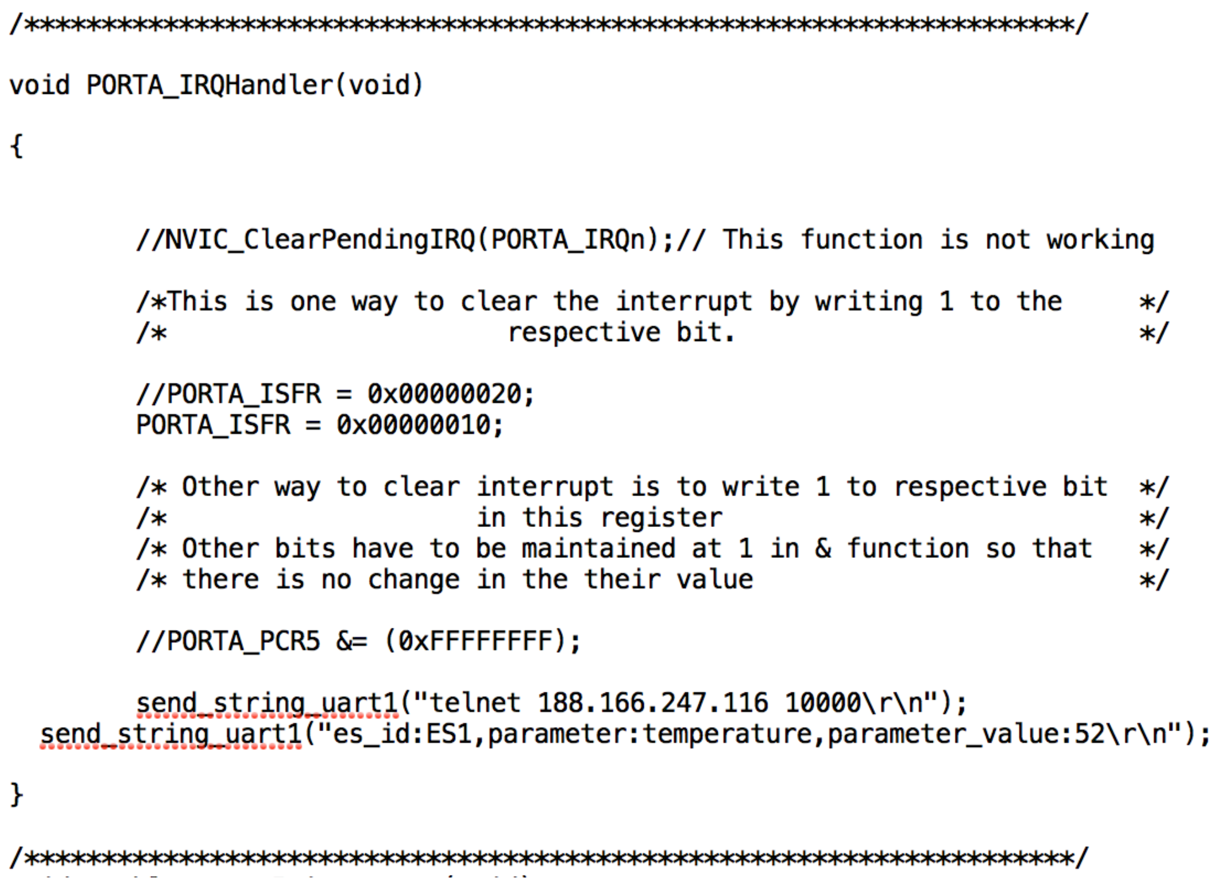
4.4.8 HAL_PortA

This file contains the code most relevant to the project. The relevant function here is the interrupt handler for port A. When a button is pressed on the board, it generates an interrupt, which instructs the processor that the code specified in the interrupt handler for that port must execute. For our purpose, this sends a string of characters to the UART where the Wi-Fi module is connected. The commands instruct the Wi-Fi module to open a telnet connection to our server's IP address, and then send a command in the following format:

"es_id:ES<X>,parameter:<parameter name>,parameter_value:<number>\r\n"

Where

- <X> is an assigned ID of the Wi-Fi module (a number to differentiate itself from other Wi-Fi modules that may be sending data to the same server)
- <parameter name> is the parameter to log (for instance, if this was to be deployed in a vending machine, it would correspond to the press of a button for a certain product, or if it were on another device, a parameter such as a temperature reading)
- <number> is the value to be logged for that parameter.



```
/**************************************************************************/

void PORTA_IRQHandler(void)

{


        //NVIC_ClearPendingIRQ(PORTA_IRQn);// This function is not working

        /*This is one way to clear the interrupt by writing 1 to the     */
        /*                      respective bit.                          */

        //PORTA_ISFR = 0x00000020;
        PORTA_ISFR = 0x00000010;

        /* Other way to clear interrupt is to write 1 to respective bit  */
        /*                    in this register                           */
        /* Other bits have to be maintained at 1 in & function so that   */
        /* there is no change in the their value                         */

        //PORTA_PCR5 &= (0xFFFFFFFF);

        send_string_uart1("telnet 188.166.247.116 10000\r\n");
   send_string_uart1("es_id:ES1,parameter:temperature,parameter_value:52\r\n");

}

/**************************************************************************/
```

In our example, this has been hard-coded to identify itself as ES1, where the parameter is temperature, and the value is 52.

4.5 Data Visualisation on the Server Dashboard

Once the webserver receives the data, it parses it and logs the activity in real-time. Below are screenshots of data sent to the server over the course of testing the system by both pressing the button on the board and issuing it directly from a terminal window. The system worked as intended, meeting the requirements. The server demonstrates a potential dashboard that a software developer may use to visualise data collected from various Wi-Fi connected IoT devices in the field.

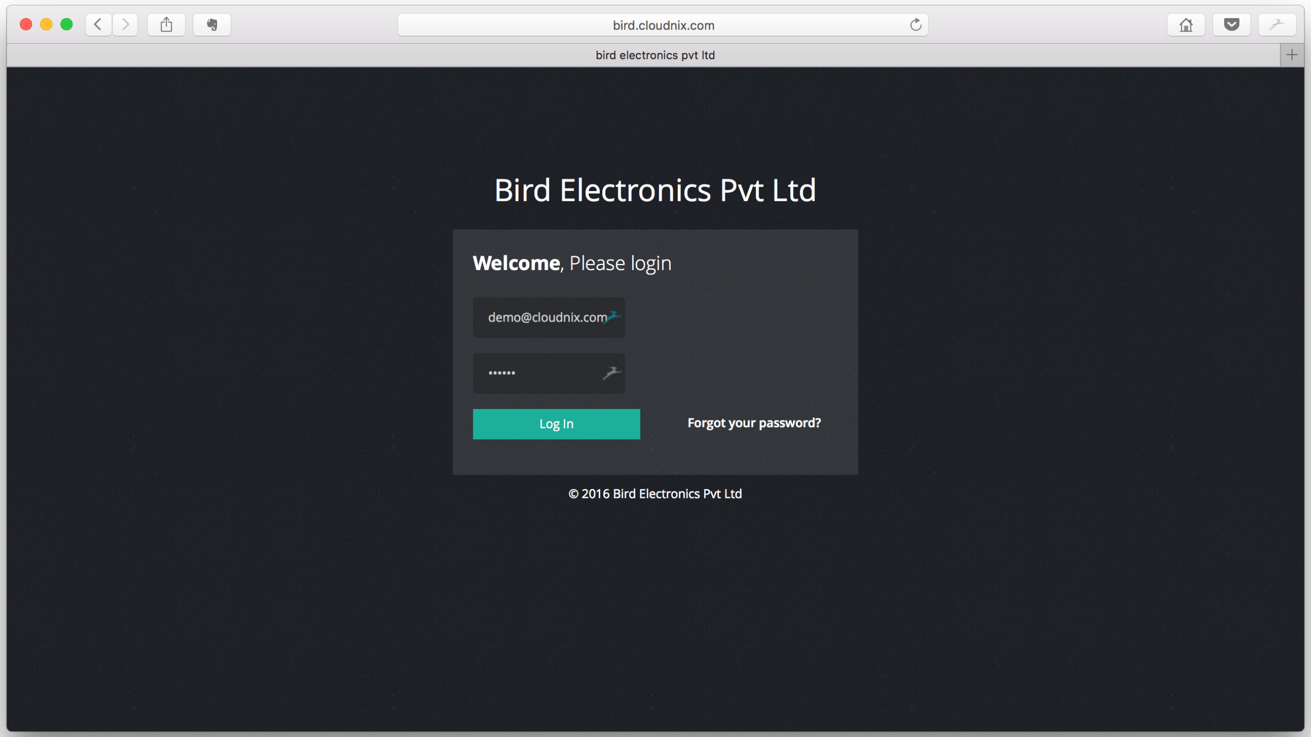
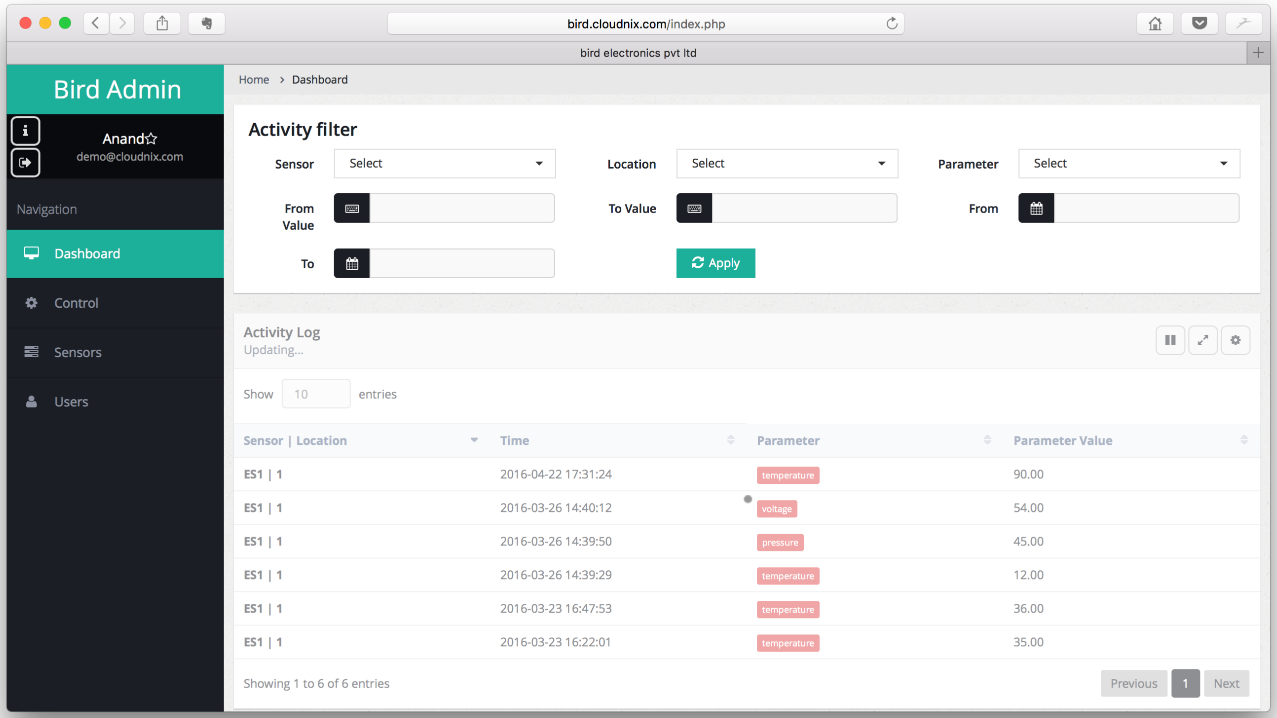


Figure 6: login screen



Activity filter

Sensor	Select	Location	Select	Parameter	Select
From Value		To Value		From	
To		Apply			

Activity Log
Updating...

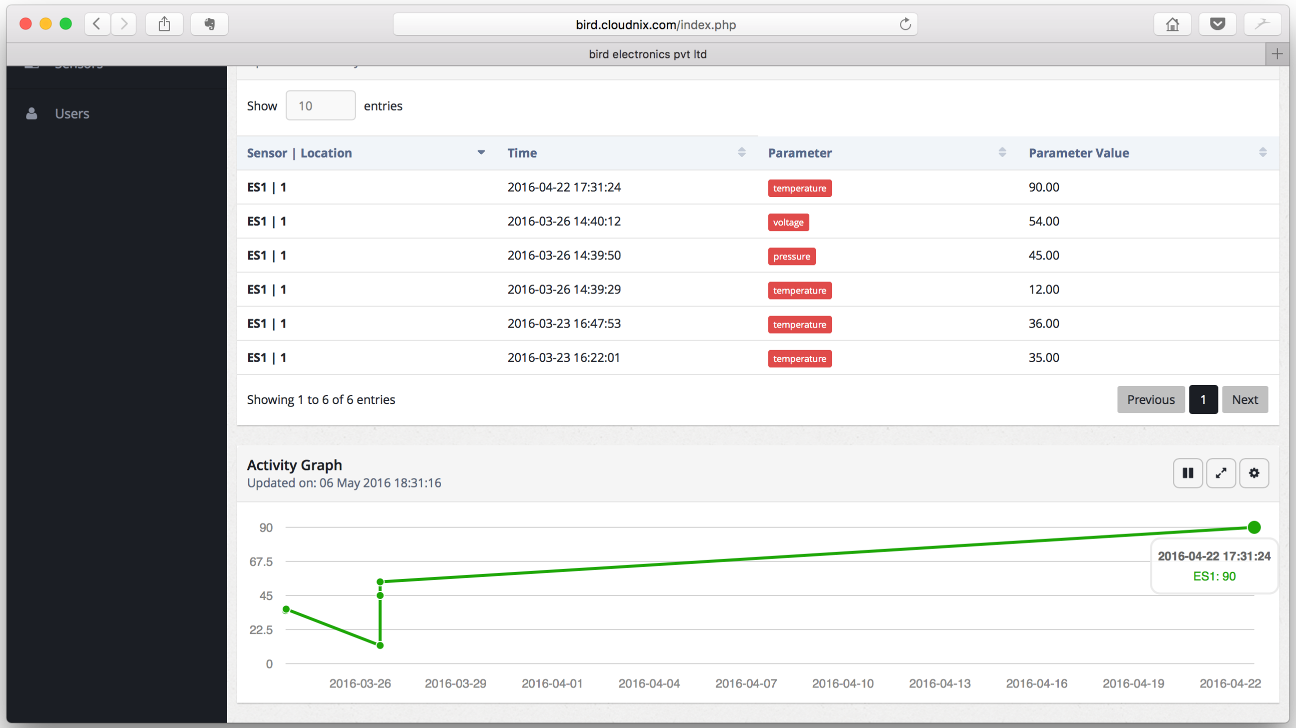
Show 10 entries

Sensor | Location	Time	Parameter	Parameter Value
ES1 | 1	2016-04-22 17:31:24	temperature	90.00
ES1 | 1	2016-03-26 14:40:12	voltage	54.00
ES1 | 1	2016-03-26 14:39:50	pressure	45.00
ES1 | 1	2016-03-26 14:39:29	temperature	12.00
ES1 | 1	2016-03-23 16:47:53	temperature	36.00
ES1 | 1	2016-03-23 16:22:01	temperature	35.00

Showing 1 to 6 of 6 entries

Previous 1 Next

Figure 7: Main dashboard activity view



Show 10 entries

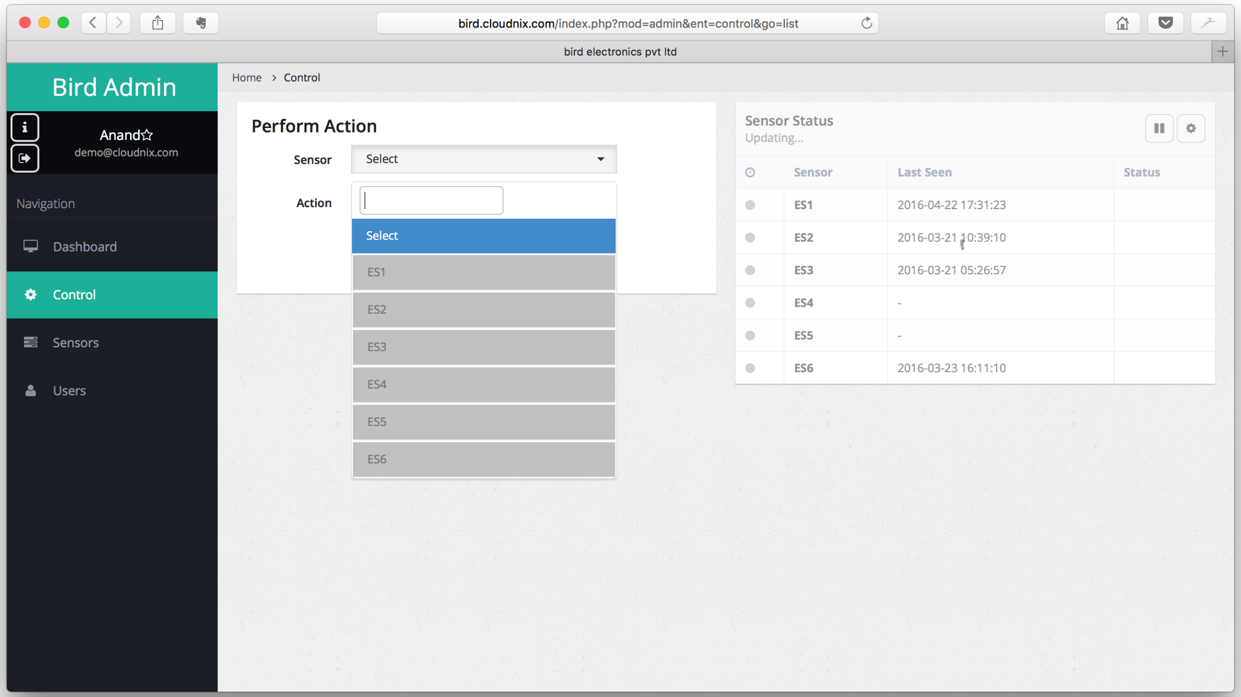
Sensor | Location	Time	Parameter	Parameter Value
ES1 | 1	2016-04-22 17:31:24	temperature	90.00
ES1 | 1	2016-03-26 14:40:12	voltage	54.00
ES1 | 1	2016-03-26 14:39:50	pressure	45.00
ES1 | 1	2016-03-26 14:39:29	temperature	12.00
ES1 | 1	2016-03-23 16:47:53	temperature	36.00
ES1 | 1	2016-03-23 16:22:01	temperature	35.00

Showing 1 to 6 of 6 entries

Previous 1 Next

Activity Graph
Updated on: 06 May 2016 18:31:16

Figure 8: Activity graph for logged data



bird.cloudnix.com/index.php?mod=admin&ent=control&go=list

bird electronics pvt ltd

Bird Admin

Anand☆
demo@cloudnix.com

Navigation
- Dashboard
- Control
- Sensors
- Users

Home > Control

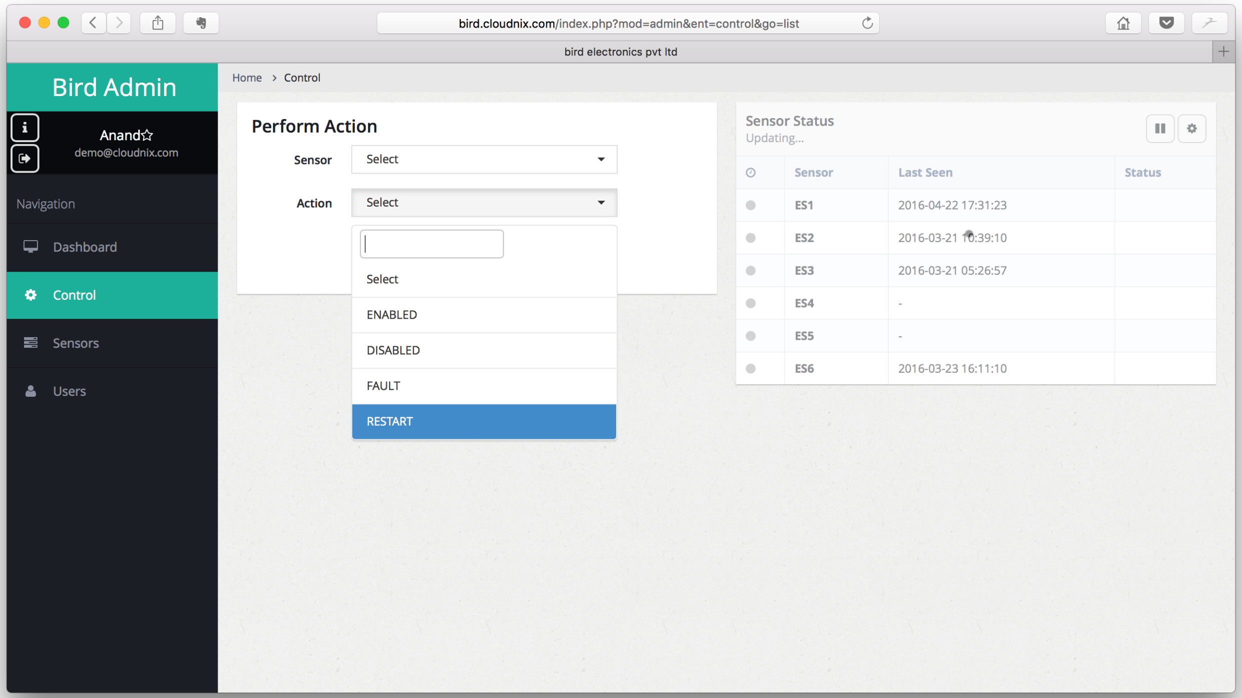
Perform Action

Sensor: Select

Action: Select, ES1, ES2, ES3, ES4, ES5, ES6

Sensor Status
Updating...

	Sensor	Last Seen	Status
●	ES1	2016-04-22 17:31:23	
●	ES2	2016-03-21 10:39:10	
●	ES3	2016-03-21 05:26:57	
●	ES4	-	
●	ES5	-	
●	ES6	2016-03-23 16:11:10	



bird.cloudnix.com/index.php?mod=admin&ent=control&go=list

bird electronics pvt ltd

Bird Admin

Anand☆
demo@cloudnix.com

Navigation
- Dashboard
- Control
- Sensors
- Users

Home > Control

Perform Action

Sensor: Select

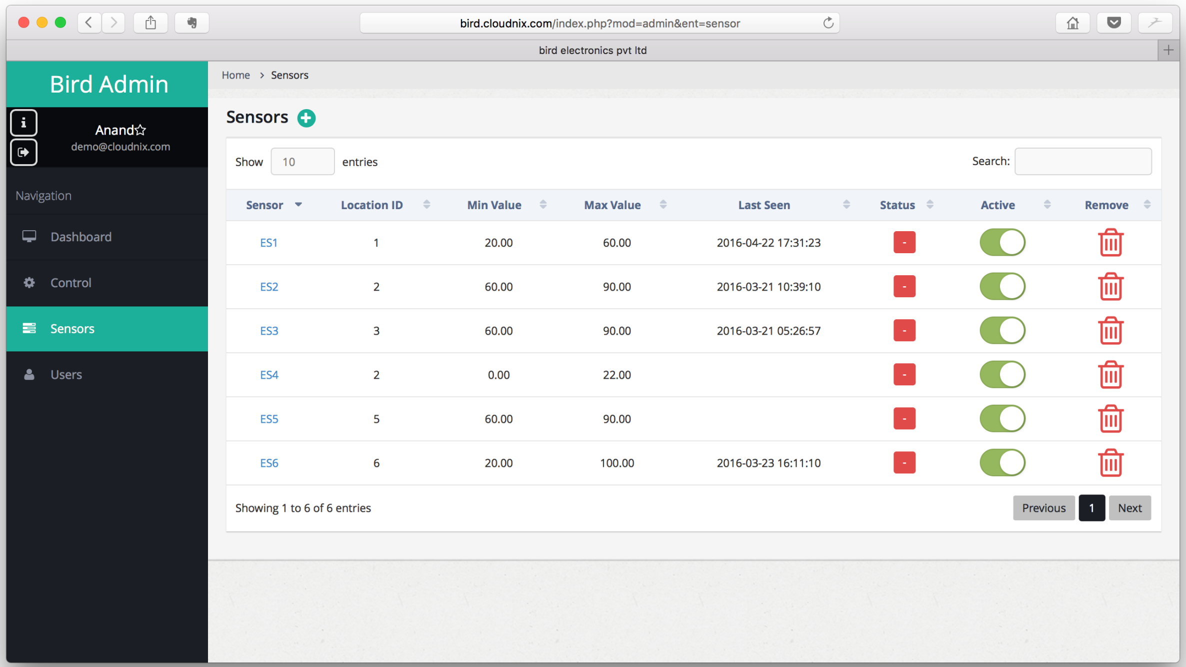
Action: Select

Select, ENABLED, DISABLED, FAULT, RESTART

Sensor Status
Updating...

	Sensor	Last Seen	Status
●	ES1	2016-04-22 17:31:23	
●	ES2	2016-03-21 10:39:10	
●	ES3	2016-03-21 05:26:57	
●	ES4	-	
●	ES5	-	
●	ES6	2016-03-23 16:11:10	

Figure 9: Control screen, as a placeholder demo where sensors may be issued commands



Sensors

Show 10 entries

Search:

Sensor	Location ID	Min Value	Max Value	Last Seen	Status	Active	Remove
ES1	1	20.00	60.00	2016-04-22 17:31:23	-	●	🗑
ES2	2	60.00	90.00	2016-03-21 10:39:10	-	●	🗑
ES3	3	60.00	90.00	2016-03-21 05:26:57	-	●	🗑
ES4	2	0.00	22.00		-	●	🗑
ES5	5	60.00	90.00		-	●	🗑
ES6	6	20.00	100.00	2016-03-23 16:11:10	-	●	🗑

Showing 1 to 6 of 6 entries

Previous 1 Next

Figure 10: List of all sensors

4.5.1 Problem-Solving Potential of the Data Collected

When data is logged in this manner, it becomes an incredibly powerful remote monitoring system where sensors on the target device begin communicating the parameters they are programmed to detect with the server. Patterns and trends may begin to emerge. The vending machine now becomes a capable device that is able to give us data from which we can infer sales patterns; for instance, if the logged parameter corresponds to the sale of a particular product (a button press against that product on the vending machine), then the graphed data may be used to deduce total revenue generated, the time at which the revenue is generated and so on. Some benefits include:

1. Remote diagnosis of machine operations and failures. For instance, if a machine has been reporting high temperatures consistently for some time.
2. Service engineers for the machine can carry parts and reduce turnaround time to repair the equipment by 70-80% by knowing beforehand the spares they may carry by observing the machine's activity.
3. Thus, the availability of the machine and revenue increase.
4. Users get an uninterrupted supply of drinking water and products from the machine.

5. It reduces wastage of resources (where a service engineer or user has to make physical trips to get information from the machine – now, it is done remotely).
6. The solution is cost-effective in the long run as the parts I used cost approximately £16.

5 Evaluation and Reflection

With this implementation finished, there is much to reflect on, in terms of what the hardware was able to achieve, areas that could be improved upon, highlighting the strengths and weaknesses of the presented solution and limitations surrounding the device.

One of the first realisations was an appreciation for how much work goes into doing a real, live project in the field. Though the hardware cost was low, the effort in making the system and all its disparate parts was inversely proportional. It was an advantage that the system was able to operate without any overhead, however, when there is no operating system involved or a simpler interface provided by a board such as the Arduino and its associated programming IDE, the effort required to do the project was multiplied. It is up to us as the end users to decide whether the initial effort in setting everything up is worth the trade when the advantage is that the hardware capability required is probably 1/1000 or 1/10,000 of what is normally required where there is an OS present (to support running the OS on the board).

The advantage for a user like me implementing network capabilities on a device using the ESP8266 is that the microcontrollers required to drive them, by themselves, cost very little. In small quantities, the standalone ARM Cortex processors are extremely powerful and capable, including being able to control the ESP8266/implement a full data acquisition and control system/become an internet node while costing only £0.3 is small quantities. There is a lot of room to buy and experiment with several devices due to their low cost. Implementations will also cost very little as a result.

Additionally, these chips are extremely reliable in terms of not hanging, and not needing software updates. It is virtually impossible to hack into them once they have been programmed and sealed.

Furthermore, they are known to be extremely power efficient, known to consume power low enough (milli-watts) to be fed by simple lithium-ion cells (for instance, CR2032-type) for years together. This may be very advantageous if, for instance, you need to make a sealed non-rechargeable system. A potential use case is if you are monitoring the temperature of a room. You do not need to sample data every second because the temperature is not changing at a rate fast enough to warrant that. If you are sampling every hour, then you can deploy a sealed system running off a battery that will run independently for a very long time before needing attention, which is paramount for the IoT space, unlike a device that runs Android, which requires more RAM, a faster processor and requires recharging.

That said, the cons of a system without an OS such as this one are that, for one, good compilers are not free. They cost in the range of USD $10,000 and more. I was able to use the KEIL compiler here since they offer it for free for projects up to 32K in size, which I soon realised after beginning to use it, was a limitation. To be able to expand this in the future, a huge investment must be made, which may not be suitable for all audiences. Other tools that one may require to do embedded programming and debugging, such as a JTAG debugger, cost anywhere from $700-$2000. This can be prohibitive for a user 3D printing a simple enclosure and dabbling in the IoT space.

Furthermore, programming embedded microcontrollers in C is an involved task. I leave this project with a deeper appreciation for what teams of dozens of computer scientists and hardware engineers do when they work on a single project to bring a product to market, and despite the developments in recent years that make engaging in programming and manufacturing IoT devices/dabbling in the space much easier, why that kind of manpower is required.

Bibliography

[1] VisionMobile, “IoT Megatrends 2016 - VisionMobile,” December 2015. [Online]. Available: http://www.visionmobile.com/product/iot-megatrends-2016/.

[2] Gartner, “Gartner Hype Cycles,” [Online]. Available: http://www.gartner.com/technology/research/hype-cycles/.

[3] T. Tunguz. [Online]. Available: http://tomtunguz.com/iot-business-model-innovation/.

[4] L. Science, “What Heart Conditions Can the Apple Watch Detect?,” [Online]. Available: http://www.livescience.com/52313-apple-watch-detect-health-problems.html.

[5] C. News, “Apple Watch Helps Save Teen Athlete's Life,” [Online]. Available: http://www.cbsnews.com/news/apple-watch-helps-save-teen-athletes-life/.

[6] Wikipedia. [Online]. Available: https://en.wikipedia.org/wiki/CAN_bus.

[7] H. Lamarque, “When the Crowd Met Pebble: a Story of Kickstarter Success,” [Online]. Available: http://www.huffingtonpost.com/hannah-lamarque/when-the-crowd-met-pebble_b_7759940.html.

[8] K. Fehrenbacher, “In the wake of fires, Tesla releases suspension software, while watchdog starts probe,” [Online]. Available: https://gigaom.com/2013/11/19/in-the-wake-of-fires-tesla-releases-suspension-software-calls-for-investigation/.

[9] A. Brisbourne, “Tesla’s Over-the-Air Fix: Best Example Yet of the Internet of Things?,” [Online]. Available: http://www.wired.com/insights/2014/02/teslas-air-fix-best-example-yet-internet-things/.

[10] C. Z. a. N. Patel, “The Verge - For CEO Mark Fields Interview,” [Online]. Available: http://www.theverge.com/2016/4/7/11333288/ford-ceo-mark-fields-interview-electric-self-driving-car-software.

[11] J. Bruner, “What's a Tech Company, Anyway?,” [Online]. Available: http://radar.oreilly.com/2014/05/whats-a-tech-company-anyway.html.

[12] “Philips Hue Developers,” [Online]. Available: http://www.developers.meethue.com.

[13] ZigBee, “What is ZigBee,” [Online]. Available: http://www.zigbee.org/what-is-zigbee/.

[14] Z-Wave, “Z-Wave FAQ,” [Online]. Available: http://z-wave.com/faq.

[15] NXP, “Freedom Development Platform for Kinetis MCUs,” [Online]. Available: http://www.nxp.com/products/software-and-tools/hardware-development-tools/freedom-development-boards/freedom-development-platform-for-kinetis-kl14-kl15-kl24-kl25-mcus:FRDM-KL25Z). .

[16] S. Electronics, “ESP8266 Wi-Fi Module,” [Online]. Available: https://www.sparkfun.com/products/13678.

[17] SparkFun, “AT Commands Datasheet,” [Online]. Available: https://cdn.sparkfun.com/assets/learn_tutorials/4/0/3/4A-ESP8266__AT_Instruction_Set__EN_v0.30.pdf)..

[18] PuTTY, “PuTTY,” [Online]. Available: http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html.

[19] M. Shacklett, “10 Ways to make the Internet of Things Pay Off,” [Online]. Available: http://www.techrepublic.com/blog/10-things/10-ways-to-make-the-internet-of-things-pay-off/.

[20] M. L. a. J. Bruner, What is the Internet of Things?, O'Reilly, 2015.