



The
University
Of
Sheffield.

Department of Electronic and Electrical Engineering

**Automated low cost device for the remote monitoring of
glucose for type-1 diabetes using mobile phones.**

Electronic and Electrical Engineering, MEng

Student: Ben Trevett

Registration No.: 110231253

Word Count: 6322

Date: 30/04/2015

Supervisor: Dr. Mohammed Beniassa

Second Marker: Dr. Charith Abhayaratne

Abstract

Diabetes mellitus is a disease that effects hundreds of millions of people worldwide. This number is only expected to increase in the future. The healthcare costs due to complications caused by diabetes are high, and are also expected to increase. Diabetes is being treated using telemedicine, an emerging field that aims to increase medical feedback and reduce healthcare costs. For diabetics, telemedicine is being used for self-management of blood glucose levels, mostly in the form of smartphone applications and smart glucometers.

In this project, a system is created to aid the self-management of blood glucose levels by using both hardware and software. It offers an alternative to the current methods of self-management by providing a way to add the advantages of smart glucometers to the patients existing glucometer.

The hardware side is made up of an Arduino Uno with a USB host shield and Bluetooth capability. The user connects their glucometer to the USB host shield and the Arduino takes the readings from the glucometer and transmits them over Bluetooth. Three methods of Bluetooth communication were used: Classic Bluetooth, Bluetooth Low Energy without using a profile and Bluetooth Low Energy using a blood glucose profile.

The software side consists of two Android applications, one for Bluetooth Classic and one for Bluetooth Low Energy without blood glucose profiles. These applications allow the user to receive the readings being transmitted via the Arduino and display them in a scrollable list. No application was created for Bluetooth using blood glucose profiles as sufficient compatible ones already exist.

The project was created successfully, meeting specification. However, there are many possible improvements that can be made to the current system as well as numerous additional features that can be added.

Contents

Abstract.....	i
1 – Introduction	1
1.1 – Diabetes	1
1.2 – Blood Glucose Monitoring	1
2 – Aims and Objectives.....	2
3 – Literature Review	3
3.1 – Telemedicine	3
3.2 – Current Research.....	3
3.3 – Smartphone Usage.....	4
3.3.1 – Smartphone Applications.....	4
3.3.2 – Smart Glucometers	5
4 – Hardware.....	6
4.1 – Hardware Selection	6
4.2 – Arduino.....	7
4.2 – Communication.....	8
4.2.1 – Serial.....	8
4.2.3 – Bluetooth	9
5 –Software	12
5.1 – Android.....	12
5.2 – Application Development	13
6 – Conclusion	15
7 – Recommendations	16
8 - References	17
9 - Appendix	22
9.1 – Receiving Readings Walkthrough.....	22
9.1 - Program Code	23
9.2 – Arduino Schematic	24
9.4 – Interim Report.....	25

1 – Introduction

1.1 – Diabetes

Diabetes mellitus is a disease of the pancreas in which an insufficient amount of the hormone insulin is produced. Insulin's main function is to break down glucose (blood sugar) in the body so it can be turned into usable energy. An increased amount of glucose in the blood can lead to organ damage. There are two types of diabetes; type-1 and type-2. Type-1 is when the body produces no insulin, and as a result insulin supplements will need to be taken for the patient's lifetime [1]. Type-2 is when the body does not produce enough insulin. Consequently, insulin supplements may not be needed and glucose levels can be controlled through diet and exercise [2].

Health complications caused by diabetes can be severe and include: strokes, heart attacks, high blood pressure, kidney failure, loss of sight and nerve damage [3]. Diabetics also have a reduced life expectancy and complications from diabetes causes 200,000 deaths in the UK every year [4].

There are 3.2 million people with diabetes in the UK [5], accounting for 6% of the population. It is estimated that half a million people with diabetes remain undiagnosed [6]. The NHS currently spend £9.8 billion on diabetes annually, a figure expected to rise to £16.9 billion in 2035 (17% of the NHS annual budget) [7]. Furthermore, 382 million people worldwide suffer from diabetes, reflected in the \$376 billion spent on treatment (11.6% of the world healthcare spending) [8]. This total is expected to rise to \$592 billion by 2035 [9].

1.2 – Blood Glucose Monitoring

The most common method for diabetics to measure their blood glucose levels is by using a glucometer.



Figure 1 – Contour Next USB Glucometer

The monitoring process is carried out by the patient first pricking their finger with a 'lancet' to produce a drop of blood. This blood is applied to a 'test strip' which is inserted into the glucometer which displays the blood glucose level after a short amount of time. Glucose readings are then recorded by hand in a 'glucose diary' [10]. The patient visits their medical practitioner every 2-4 months with their readings to receive feedback which may involve recommended changes to diet and exercise and changing of insulin dosages. This current method, although acceptable, has its downsides. For example, the patient may not take their own measurements frequently enough, they may not be receiving the correct insulin dosages and the patient does not receive feedback from their medical practitioner often enough [11].

2 – Aims and Objectives

The aim of this project is to develop a low cost system for the remote monitoring of glucose levels in diabetics in which the patient's glucose levels can be remotely monitored by their medical supervisor. The system needs to be user friendly and provide a clear benefit over using the 'pen and paper' method of the glucose diary. The system consists of two parts: hardware and software.

The hardware side of the system is made up of a device that has three main functions: receiving, processing and transmitting. The receiving functionality is done when the patient connects their glucometer into the device and the readings on it are transferred from the glucometer to the device's memory. Processing is carried out by the device and formats the received data. The transmitting is where the formatted data is sent from the device to the software. The hardware must be portable, low cost and low power.

The software side uses a mobile phone application to carry out the same three main functions as the hardware side: receiving, processing and transmitting. The application receives the data transmitted by the hardware and then processes the data in order to display it to the patient. Transmitting of the data is done by uploading it to a webserver where the patient's medical supervisor can view this data. The application must display the data to the user in a clear and concise way.

The scope of this project only covers the system from the glucometer to the application. The webserver, although part of the overall system of glucose monitoring, is not covered in this project.

Below is the Gantt chart which shows the project schedule. The horizontal axis is the week number and the vertical axis displays the activity.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
A																				
B																				
C																				
D																				
E																				
F																				
G																				
H																				
I																				
J																				
K																				

Table 1 – Gantt Chart of Project Schedule

KEY

- A. Initial Subject Research
- B. Hardware Research
- C. Programming Research
- D. Writing Program
- E. Debugging and Testing Program
- F. Application Research
- G. Application Design
- H. Writing Application
- I. Debugging and Testing Application
- J. Presentation Preparation
- K. Final Report Preparation

3 – Literature Review

3.1 – Telemedicine

Telemedicine is the process of providing medical support to patients in different geographical locations [12]. It can provide services such as: remote monitoring of patients, referral services and providing health information or education. It can take the form of synchronous communication, such as a phone call, or asynchronous communication, in the form of 'eHealth' websites [13]. Patients who live in rural areas or in developing countries with poor infrastructure benefit most from telemedicine due to them being able to communicate with their medical supervisor without travelling long distances [14]. It also reduces healthcare costs due to better management of healthcare practitioner's time and by reducing unnecessary hospital visits for patients [15].

Due to progress made in smartphone technology and increased numbers of smartphone usage, telemedicine has become a topic of interest of the last decade [16]. Studies have shown that continual self-monitoring of glucose levels compared to non-self-monitoring had a significant improvement on controlling blood glucose levels [17]. The use of self-monitored blood glucose levels stored in an application have also been shown to increase blood glucose control and reduce hospital visits [18] and research has found that the use of smartphones with regards to blood glucose management has opened up a wide variety of options to patients [19].

Although telemedicine has many potential benefits, it has not achieved maximum adoption due to a number of factors. Most telemedicine projects have difficulty continuing to run after initially beginning, due to losing funding [20]. This is due to telemedicine not having proved its financial viability for large scale medical companies to move into the market [21]. Telemedicine relies on advanced technologies that are carried out by small teams and require correspondence with health care professionals. Some studies have shown that in some cases these correspondents have a reluctance to provide the effort needed to push these systems [22]. There are also problems with patients' perspective of telemedicine, due to them either not being accepting of new technology or possessing the technical knowledge to use it [23].

3.2 – Current Research

There has been research into replacing the standard glucometer with 'interstitial' glucometers, such as the FreeStyle Libre [24]. These require the patient to have a small patch on their arm that has a glucose sensor underneath it which goes into the patients' skin. These patches can contain a near field communication (NFC) chip and therefore the patient simply places the paired 'reader' in close proximity to the patch to take their current glucose readings.

Interstitial glucometers possess many benefits to the patient, such as convenience, the glucometer continually measures and stores readings without any input by the patient; less invasive, users can scan their reading in less than a second and through items of clothing and are usually water resistant and user friendly, from being activated by proximity via NFC without the complications of pairing associated with Bluetooth.

The downside of interstitial readings is that they are measured in the interstitial fluid, not within the blood. Changes in glucose levels in the blood through food consumed can take up to 45 minutes to convey that same change in the interstitial fluid. This delay can also cause an inaccuracy of up to 15% in readings [25]. There has also been technical issues such as skin irritation, missing readings and false alarms, as well as psychological issues such as too much information being displayed to the patient and causing anxiety due to thinking about their condition more often [26].

3.3 – Smartphone Usage

3.3.1 – Smartphone Applications

There currently exists two main types of products that offer blood glucose management using smartphones: smartphone applications and smart glucometers.



Figure 2 – Blood Glucose Management Application, Glooko

Smartphone applications act as a replacement the glucose diary, as well as offering many other improvements, such as:

- As they are on the patient's mobile phone, they are much easier to transport.
- New readings can be added quickly in a few button presses within the application
- The amount of data can be held is only limited by the phones internal memory
- Reminders can be set to remind to patient to take a reading
- Data can be displayed in a more convenient format, such as displaying average readings, displaying in a graphical form, etc.

Applications are, however, not without their disadvantages. For example:

- The application is linked to the phone, meaning it has all the downsides of a phone such as battery life, applications crashing and losing data, etc.
- There are some situations where pen and paper is more convenient
- Patients may not own their own smartphone, especially true in developing countries or in younger patients
- Patients may not be familiar with the technology and be unwilling to use over pen and paper

3.3.2 – Smart Glucometers

There are also smart glucometers, such as iHealth Labs' BG5 [27] and the OneTouch VerioSync [28]. These are glucometers with Bluetooth built-in and are able to transfer data directly from the glucometer directly to the smartphone, removing the need to manually enter data into the application.



Figure 3 – Smart Glucometers; iHealth Labs' BG5 (left) and OneTouch VerioSync (right)

Although smart glucometers provide great convenience, they have several drawbacks. Using the two smart glucometers pictured in figure 3 as examples, these drawbacks are:

- Requiring the patient to use the application unique to the smart glucometer - This forces patients who use an application to measure their glucose levels to switch and potentially lose all their previous data.
- Compatibility with patient's smartphone - Not all applications are available for all platforms, for example the OneTouch VerioSync's application is only available for iOS and the BG5's application is only available for iOS and Android. Neither glucometer has an application for Windows Phones.
- Suboptimal power usage - Both the example smart glucometers use Classic Bluetooth, not Bluetooth Low Energy (BLE). Classic Bluetooth uses more power than BLE [29] and this can only be solved by changing the Bluetooth chip within the device.
- Limited availability - Both the BG5 and the OneTouch VerioSync are only available in America.
- Higher cost - The smart glucometers are only compatible with unique testing strips. These are more expensive than standard strips. Standard test strips costing approximately \$0.5 each [30] whilst test strips for the OneTouch VerioSync cost approximately \$1.6 each [31].

The system developed for this project aims to minimise as many of the drawbacks of using a smart glucometer as possible. Instead of designing a competing smart glucometer, a device will be created that will aim to be compatible with as many of the existing glucometers as possible and will implement functionality similar to that of the smart glucometer. This will allow patients to continue to use their current glucometers and low cost testing strips but allow them to take advantage of using an application to replace their glucose diary. The hardware must take advantage of BLE to ensure minimal power usage.

4 – Hardware

4.1 – Hardware Selection

To fit the specification for the hardware, the device selected must be low power, low cost and portable. The three potential devices were: Intel Edison, Arduino Uno R3 and the Raspberry Pi B+.



Figure 4 – Intel Edison with Mini Breakout Board, Arduino Uno R3 and the Raspberry Pi B+

The Intel Edison has BLE built into the device, however it does not possess any USB host ports [32]. The Arduino Uno and the Raspberry Pi do not have Bluetooth functionality [33] [34], and the Arduino Uno doesn't possess a USB host port. Therefore, for a fair comparison between the devices the Intel Edison must use the 'Mini Breakout Board' [35], the Arduino Uno must use a BLE shield [36] and a USB host shield [37] and the Raspberry Pi must use a BLE module [38]. This will allow all devices to connect to the glucometer and transmit data to the application, as well as fitting the condition of being low power.

The approximate cost of each devices is:

- Intel Edison with Mini Breakout Board - £60 [39]
- Arduino Uno with BLE shield - £40 [40] [41]
- Raspberry Pi with BLE module - £40 [42] [43]

The Raspberry Pi B+ and Arduino Uno are a comparable price, with the Intel Edison costing 50% more. Comparing sizes of the device:

- Intel Edison with Mini Breakout Board – 4.5 * 3 cm – 13.5 cm² [32]
- Arduino Uno – 7 * 5 cm – 35cm² [33]
- Raspberry Pi – 8.5 * 5 cm – 42.5cm² [34]

The Intel Edison is the smallest board, followed by the Arduino Uno and then the Raspberry Pi.

The Raspberry Pi was then discarded as a choice of hardware as it is the same price as the Arduino Uno, but offers no benefit in terms of size. The Edison is much higher spec than the Arduino, for example the Edison has 4GB of memory and 1GB of RAM [32], whilst the Arduino has no internal storage and only 2kB of RAM [33]. The Edison is considered to be more of a small computer running a Linux operating system, the Arduino Uno is an embedded microcontroller.

It was decided that the Arduino Uno would be used for this project. Although the Intel Edison provided many benefits over the Arduino, the main specification was cost and size. The Arduino is much cheaper than the Edison, and although the Edison is smaller than the Arduino, the Arduino is small enough to still be considered portable and thus fit specification.

4.2 – Arduino

The Arduino family is a series of open source microcontrollers [44]. There are an approximate 700,000 official Arduino boards in circulation [45] and as such, they have a large homebrew following. The use of ‘shields’, headers that connect on top of the Arduino, allows for extra functionality to be added to the Arduino board, such as: adding an LCD screen, an Ethernet port, Wi-Fi connectivity, ability to play sounds, etc. [46]. The two shields used in this project were a USB host shield and a BLE shield.

The Arduino is programmed using Arduino’s integrated development environment (IDE) and coded in C/C++. Code written using the IDE are called ‘sketches’. Through the IDE, sketches can be compiled and then uploaded to the Arduino [47].



Figure 5 – Example ‘Blink’ Sketch on Arduino IDE

The IDE also makes use of ‘libraries’. Libraries, similar to shields, add extra functionality to the IDE, such as allowing the IDE to communicate with shields, for example reading data from a temperature sensor shield, or by allowing the IDE to perform advanced data processing [48].

4.2 – Communication

4.2.1 – Serial

The serial communication is carried out between the glucometer and the Arduino and uses the USB host shield.

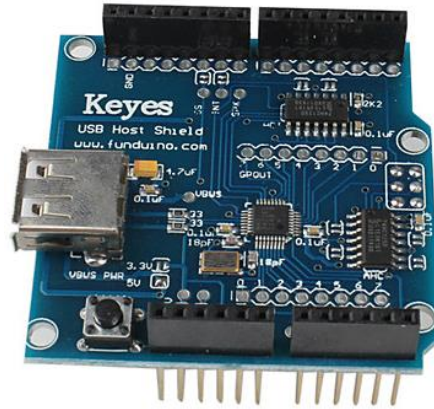


Figure 7 – USB Host Shield

Although the Arduino already has a USB port, this is used for powering the device and compiling to the board only. The USB host shield adds the capability for the Arduino board to use: human interface devices (HIDs) such as mice and keyboard, USB storage devices, USB Bluetooth dongles, etc. [49].

The GitHub page for the USB host shield contains many Arduino libraries for different devices, such as using a PS3 controller [50]. As the code is open source, it is freely available to be downloaded and edited. Using a library for a MIDI controller as a basis [51], the code was edited to work with a glucometer.

First, the glucometer is connected to the Arduino through the USB host shield and a 'USB description' sketch (included in the USB host library) is ran. This returns a detailed description of the USB device connected, including: product ID (PID), vendor ID (VID) and endpoint (EP) descriptions.

Both the product ID and the vendor ID are 16-bit numbers that are used by a computer to identify the USB device connected [52]. These can be used either to tell if the correct device is connected (or respond with an error if it is not) or to allow the sketch to perform different actions depending on which device is connected, for example setting the endpoint address for the connected glucometer.

Endpoints will have an address, a defined direction, either IN (from USB device to computer) or OUT (from computer to USB device) and one of four types: control, isochronous, interrupt or bulk [53]. The glucometer used in this project is the Contour Next USB glucometer, which uses bulk transfers. This type of transfer is good for large amounts of data, however the speed of the transfer varies [54]. This was deemed an effective method of transfer as readings are relatively small file size and can be transferred quickly even at low speeds.

See section 9.1 on a walkthrough for serial transmission.

4.2.3 – Bluetooth

Bluetooth is a method of wireless communication, similar to Wi-Fi and Zigbee, which is specialised for short distances with low power consumption [55]. Bluetooth is maintained by the Bluetooth Special Interest Group (SIG) that maintains the ‘Bluetooth Core Specification’ [56] a set of documents which details the definitions and standards that companies using Bluetooth must work toward. Bluetooth Core Specification 4.0 (also known as Bluetooth Smart) was released in 2010 and contained new protocols for Bluetooth Low Energy (BLE), an alternative to ‘Classic Bluetooth’, which is designed to use less power [57]. Bluetooth data can also be formatted for ‘profiles’, which are designed for devices so they are able to understand how to interact with the data it is receiving. There are Bluetooth profiles for printers, headsets, proximity sensors, etc.

Three different types of Bluetooth communication were used in this project: Classic Bluetooth, BLE without using the Blood Glucose Profile and BLE using the Blood Glucose Profile. Although low power was selected as one of the specifications for this project and Classic Bluetooth is not as low power as BLE, it was still used. This is because to make use of BLE, a smartphone must have a BLE chip within it, and BLE chips come in two forms: single mode and dual mode. Dual mode supports both BLE and Classic Bluetooth, while single mode only supports BLE [58]. Also, the BLE protocol for Android applications is only supported on Android version 4.3 or higher – see section 5.2. Therefore, it was decided that to be compatible with as many phones as possible an option of using Classic Bluetooth was used. The method of Bluetooth alters the cost of the overall system, with the Classic Bluetooth module costing £2 [59], the BLE shield costing £20 [41], and the CSR chip for Bluetooth using profiles costing £70 [60].

For Classic Bluetooth, a HC-05 wireless serial Bluetooth transceiver was used.

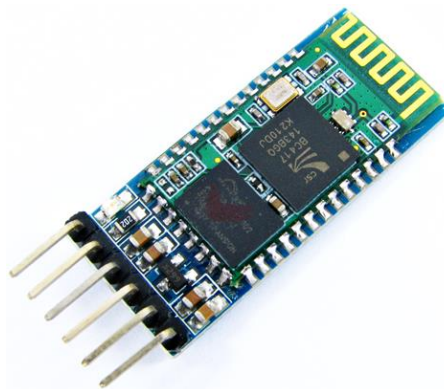


Figure 8 – HC-05 Wireless Serial Bluetooth Transceiver

The HC-05 module is a serial port protocol (SPP) module that can act in master or slave mode [61]. It acts to replace the serial port of the microcontroller. The HC-05 has connections labelled ‘VCC’, ‘GND’ (ground), ‘TX’ (transmit) and ‘RX’ (receive). The ‘VCC’ and ‘GND’ must be connected to the 5V and the ground pins of the Arduino respectively, whilst the ‘TX’ of the HC-05 must be connected to the ‘RX’ of the Arduino, whilst the ‘RX’ of the HC-05 must be connected to the ‘TX’ of the Arduino. When connected as such, any information written to the serial port from the Arduino at the same baud rate as the HC-05 module will be sent wirelessly via Bluetooth. For example to write “Hello World!” over Bluetooth, the code needed is simply:

```
Serial.print(“Hello World!”);
```

A complimentary application must be created to read the data over Bluetooth.

For BLE without Blood Glucose Profile, RedBearLab's BLE Shield was used.



Figure 9 – RedBearLab's BLE Shield

The BLE shield uses Nordic Semiconductor's nRF8001 Bluetooth chip [62] an extremely low power chip that lasts months off of a coin cell [63]. As the nRF8001 only supports single mode BLE, it is incompatible with Classic Bluetooth devices. The shield is simply connected on top of the Arduino. For the Arduino to make use of the shield, the libraries for the nRF8001 and the BLE shield must be imported [64]. This allows the IDE to make use of functions within those libraries. Instead of simply writing to the serial, the 'ble_write' function is used to write characters to an outgoing buffer, then the 'ble_read' function must be written to the serial port. For example:

```
ble_write('Hello World!');
```

```
Serial.write(ble_read());
```

When both the USB host shield and the BLE shield are connected to the Arduino, a problem with interference arose. The Arduino board communicates with the shields using serial peripheral interface (SPI), where the Arduino acts as the master and the shield acts as a slave [65]. SPI uses 4 pins on the Arduino: MISO (Master In Slave Out), sends data from the slave to the master; MOSI (Master Out Slave In), sends data from master to slave; SCK (Serial Clock), used to synchronise data transmission and SS (Slave Select), used to select which slave device the master is talking to [66]. The SS pin must be unique for each shield or else the interference between MISO and MOSI pins between the two shields will cause them to work incorrectly or damage the shields. The desired shield is selected by holding its SS pin low. By default both shields use pin 10 for SS.

Two alterations must be made to change the SS pin, hardware and software. The USB host shield was selected as the slave device to change to SS pin on as it requires less software modification. On the shield itself the leg of pin 10 originally connecting to the BLE shield is bent outwards so it does not connect the two shields, and a connecting wire is soldered between the SS pad and pin 8. Within the library for the USB host shield, the SS pin was changed from 10 to 8, and at the beginning of the Arduino sketch, in the setup() function, the pins need to be set correctly, as shown:

```
pinMode(8, OUTPUT);
```

```
pinMode(8, LOW);
```

```
pinMode(10, OUTPUT);
```

```
pinMode(10, HIGH);
```

This will keep pin 10 high, so the BLE shield will not interfere with the USB host shield until the Bluetooth communication is needed, at which point pin 8 will be set high and pin 10 will be set low. The shield has an application created by RedBearLab that reads the data transmitted via Bluetooth. For BLE with the Blood Glucose Profile, CSR's μ Energy Starter Development Kit was used.

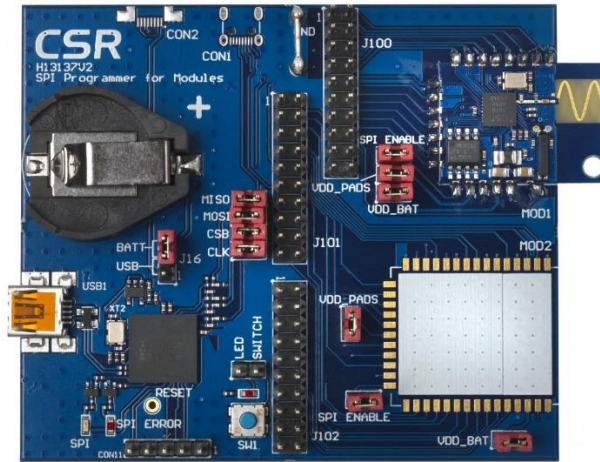


Figure 10 - CSR's μ Energy Starter Development Kit

The μ Energy Starter Development Kit contains the CSR1010 Bluetooth chip [67]. Although only the chip is needed, the board allows for ease of development. Similar to the HC-05 module the 'VCC', 'GND', 'TX' and 'RX' pins on the development kit must be connected to the 5V, ground, 'RX' and 'TX' pins of the Arduino, respectively. The data read from the glucometer is written to the serial port.

The CSR chip must be programmed using the CSR μ Energy Software Development Kit (SDK) [68]. The CSR chip is specialised for Bluetooth profiles and the SDK contains an example for using a Blood Glucose profile [69]. However, when using the Blood Glucose profiles, the time and date must be changed into Unix time. Unix time (or Epoch time) is the number of seconds that has passed since the 1st of January 1970 at 00:00 at UTC (disregarding leap seconds), known as the Epoch [70]. For example the date 17/05/15 at 13:45:00 is represented in Unix time by 1429278300. It is used as it does not change depending on geographical location, thus is independent of time zones [71]. Therefore, the Time library [72] must be imported and the Arduino code must be edited.

After the data is transferred to the CSR chip it will transfer it via Bluetooth. The data can be read by any application that can read Blood Glucose profiles and can be for either Android, iOS or Windows phones, for example CSR's own 'BLE Apps' application [73].

5 –Software

5.1 – Android

It was decided that the application for the project was to be written for Android phones. Android was the chosen platform as it is: open source [74], the most popular OS (greater than Windows, iOS and Mac OSX combined) [75] as well as having a wide variety of free development tools and documentation available, including: IDE, sample code, emulators, debuggers, documentation, tutorials, etc. [76].

Android applications are created using Android Studio, an IDE designed for all Android devices, including: tablets, phones and smartwatches. It contains many features useful for application development, such as: templates, importable examples, drag-and-drop layout, built in Android emulator, etc. [77]. Android applications are made up of two main parts .java files and .xml files. The java elements declare the functionality of the application, such as telling the application to display a message to the user once a button is pressed. The xml controls the UI elements on the screen, for example the shape, size and colour of buttons and where they are positioned.

Each screen in an application is called an activity [78]. Activities can be switched between using 'intent'. Intent can also be used to pass data when switching between activities or other applications, such as opening a URL in a web browser.

User input is controlled by 'onClickListen' functions [79]. UI elements are referenced to by these functions and when they are pressed on, they perform the code inside the function, such as displaying a pop-up message to the user (known as a 'toast') [80] or to pass intent.

Data can be stored within the application itself, increasing its size as more data is stored, or on external storage, such as an SD card [81].

UI elements can also be controlled by the java code, such as changing the text inside a text box when a button is pressed. The XML allows for multiple layouts [82], such as RelativeLayout, where items are displayed relative to one another; LinearLayout, where items are displayed in a horizontal or vertical list or in a scrollable list using ListView.

The application's manifest is also an XML file. The manifest sets the applications name, icon, theme, minimum compatible version of Android and which activity will appear when the application is launched [83]. It also manages the permissions, for example the applications for this project need permission from the user to use Bluetooth.

5.2 – Application Development

The application for the BLE shield was taken by modifying an open source version of RedBearLab's Android application [84]. The application originally began by searching for nearby BLE devices to connect to and once connected it allowed the user to send text from the Bluetooth device to the Android phone and for the phone to send text to the Bluetooth device. To increase user friendliness and allow for data storage and presentation the application was heavily modified.

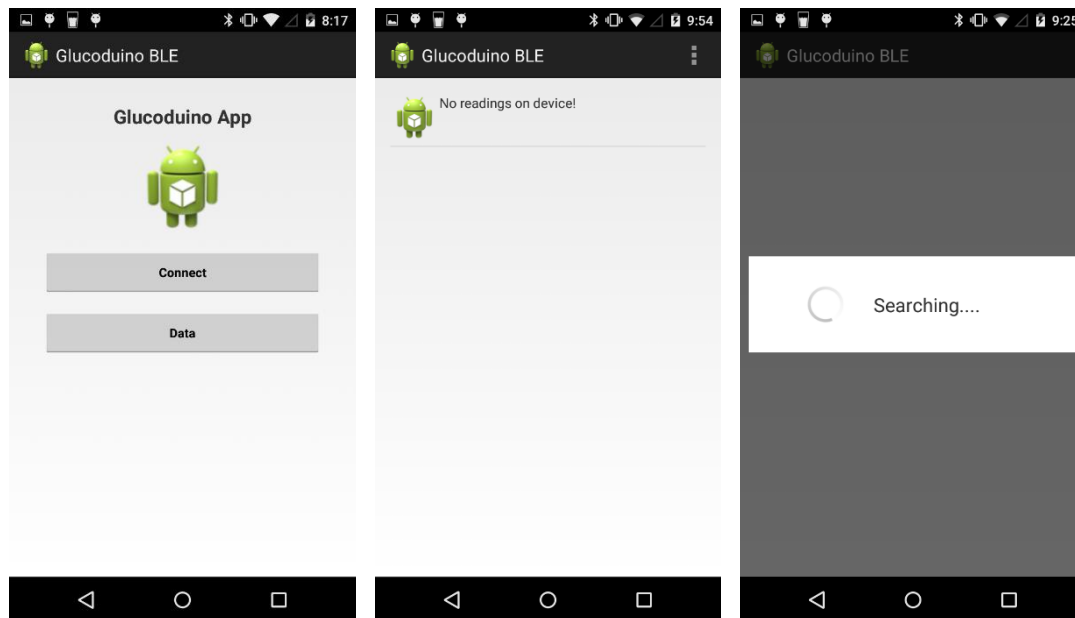


Figure 11 – Application Main Activity (left), Empty Data View (centre) and Searching for Bluetooth (right)

A new main activity was added to give the user the option to connect to a device and update the data on the application or to view the data currently on the device. This was designed to be the simplest activity as to not overwhelm the user with information on their first use of the application. The xml of this activity consists of 4 elements: the title of the application, an image and two buttons. When either button is pressed, it takes the user to a new activity. If the 'Data' button is pressed before any data is received, the page simply displays a message telling the user there is no data stored on the application.

When the 'Connect' button is pressed it takes the user to a new activity where the application begins to automatically scan for BLE devices and displays a list of each one found with their name. If no devices are found it alerts the user and they are given the option of scanning again. If devices are found the user can press on a device name to connect to it. If connection is unsuccessful the user is alerted and are brought back to the option to scan for devices again. If connection is successful they are taken to a new activity that displays a list of all data from the device as it is received. Each reading received is converted into a string and stored in an array of strings.

Once all the data has been read the application automatically switches activity to the 'Data' activity and passes the array of readings. In the 'Data' activity, the readings array is manipulated, taking the type of reading, value and date and passing each one into a java object. A 'DataAdapter' is used to display the java objects into a scrollable list [85]. The data is saved to the internal file storage of the phone [86]. From this point, if the application is closed or the user moves off the 'Data' activity, the readings are stored and will remain until the application is uninstalled.

The application for the Bluetooth communication using the HC-05 module follows the same activity flow. However, instead using RedBearLab's Android application, Android's own Bluetooth Chat example was modified [87]. It was necessary to create a second application because Android versions below 4.3 do not support the BLE protocols [88] even if the smartphone itself contains a BLE chip. In both of the applications, the ability to send text from the phone to the Bluetooth device was removed as it was unnecessary.

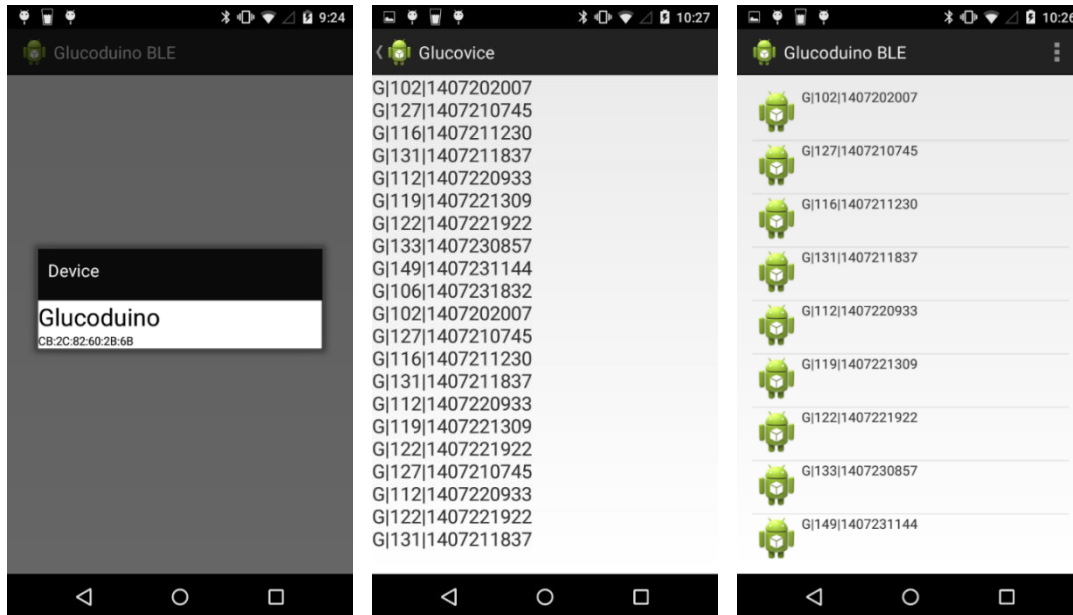


Figure 12 – Selecting a Bluetooth Device, Receiving Data and Displaying Data

When using the CSR chip, an application that uses Blood Glucose profiles must be used. Due to the lack of documentation or existing samples and complexity of Blood Glucose profile compatible applications, it was decided that CSR's existing application [73] would be used to debug the CSR chip and one was not created.

6 – Conclusion

To conclude, a prototype system for the remote monitoring of Glucose levels was successfully created to specification. The system was relatively low cost, depending on the method of Bluetooth used. This cost could have been reduced by creating a BLE shield with USB host features for this project. The system provides a reasonable alternative to ‘pen and paper’ methods.

The Arduino successfully reads data from the glucometer and processes it to be sent via Bluetooth, however is only compatible with the Contour Next USB meter and Contour Next Link glucometers.

The three different Bluetooth options fragmented the project, meaning there was an Arduino sketch for Classic Bluetooth, which simply writes the data to the serial port; BLE with the shield, which uses functions from the shield’s library and the CSR chip, which needs the date and time converted into Unix time before transmitting over serial port.

The hardware is easy to use, as it only requires the patient to connect the glucometer while the Arduino is powered up to begin the data collection and transmission process. The Arduino has only been tested while connected to the mains, ideally it would be powered off a chargeable battery to increase portability. Although low power was a design consideration design considerations to do with minimising power usage, such as coding a low power using ‘sleep’ mode for the device to enter when not in use were not inside the scope of this project. When the device is in operation there is no connection to a serial port to read the debug messages, thus it is difficult to diagnose any problems if the system fails to work correctly.

The application successfully reads data from the glucometer and displays it in a scrollable list for the user to read. The problem of fragmentation also arises here. The two separate applications for Classic Bluetooth and BLE perform the same functionality and re-use the majority of the same code, the only major difference is the different protocols used. Ideally one application would be able to use both Bluetooth forms, possibly with the addition of using Blood Glucose profiles too.

The application itself is very minimal and is only available for Android. The lack of features may detract from the user friendliness, for example a graphical library for Android, such as ‘AndroidPlot’ [89], could be used to display readings and there is no option to view the average reading over a certain period of time. There is an option to press on individual readings to get more data on them, however the functionality was not utilised and currently only displays dummy variables. There is no feature to alert the user if they have not taken readings for a period of time and patients are unable to manually add a reading.

There is also currently no way to upload the information to the webserver, this due to the webserver being under construction whilst this project was ongoing and there were several uncertainties on how the data will be uploaded, for example would the user have to log in via a name and password system or would they be identified by their glucometer’s serial number, etc. The issue of storage space was not a consideration in the specification, however alterations to the code could be made so the application uses less memory, such as removing older readings which have been uploaded to the webserver.

Overall, whilst both the hardware and software side have room for many improvements to be made on them to increase functionality or ease-of-use. Both sides meet the aims set by the project.

7 – Recommendations

There are several improvements that could be made on the system in its current state:

- Increased compatibility for glucometers – adding the functionality for more glucometers to be used will allow for the system to accommodate more patients who do not wish to change their glucometer.
- Reduce fragmentation – ideally the three different methods of Bluetooth communication could be compatible with a single application that would automatically find the correct device by searching for its name.
- Improved features of application – the application is currently very basic and for it to offer an alternative to other applications and pen and paper methods it must add more features such as: manual addition of readings, user reminders, graphical displays, average values of time, uploading data to webserver, etc.
- Increased application platforms – the applications are currently only available for Android phones, to accommodate for more patients the system must have applications available for iOS and Windows phones.

As for additions to the current system, one idea was to use near field communications (NFC) to improve both the user experience and device operations. NFC allows devices to communicate with each other in very close proximity [90]. This will be familiar to most users in the form of contactless payment systems. [91]. Due to the increasing popularity of contactless payment systems, more smartphones being produced contain NFC chips [92]. There are also ‘NFC tags’ small cards that contain NFC chips within them [93], these cards can be written to and read from using an NFC enabled device such as an NFC enabled smartphone. NFC shields are available for Arduino boards [94].

Possible ways the addition of NFC can improve this project are:

1. NFC could be used to simply turn-on Bluetooth. This would allow the user to simply hold their NFC enabled smartphone next to the NFC header on the Arduino. The Arduino will be programmed to stay in a low power consuming sleep mode until an NFC device wakes it up to turn on Bluetooth and begin transmitting data to the application.
2. NFC is also capable of data transfer and can act to replace Bluetooth. Although the data transfer rate is half of BLE, it has similar power consumption [95]. The halved transfer rate has minimal effect on the system as the data is transferred at great speed. This solution has the most effect on users who are not familiar with technology. NFC requires no pairing or confirmations, once the smartphone is within range the application will automatically open, transfer data and then alert the user once complete. All the user has to do is place their phone in close proximity to the NFC card.
3. A ‘glucose card’ could be created by using an NFC tag. Data would be transferred to the tag, which would hold selected measurements, such as readings taken in the last month. This could be used where patients have no internet access or do not possess a smartphone as it allows them to carry a large number of their readings in a compact form.

8 - References

- [1] [Online]. Available: <http://www.nhs.uk/Conditions/Diabetes-type1/Pages/Introduction.aspx>.
- [2] [Online]. Available: <http://www.nhs.uk/Conditions/Diabetes-type2/Pages/Introduction.aspx>.
- [3] [Online]. Available: <http://jdrf.org/life-with-t1d/type-1-diabetes-information/diabetes-complications/>.
- [4] [Online]. Available: <https://www.diabetes.org.uk/Documents/About%20Us/Statistics/Diabetes-key-stats-guidelines-April2014.pdf>.
- [5] [Online]. Available: http://www.diabetes.org.uk/About_us/News_Landing_Page/Number-of-people-diagnosed-with-diabetes-increases-to-32-million/.
- [6] [Online]. Available: http://www.diabetes.org.uk/documents/reports/silent_assassin_press_report.pdf.
- [7] [Online]. Available: http://www.diabetes.org.uk/About_us/News_Landing_Page/NHS-spending-on-diabetes-to-reach-169-billion-by-2035/.
- [8] [Online]. Available: <http://www.idf.org/millions-unite-diabetes-awareness-world-diabetes-day-2010>.
- [9] [Online]. Available: <https://www.diabetes.org.uk/Documents/About%20Us/Statistics/Diabetes-key-stats-guidelines-April2014.pdf>.
- [10] [Online]. Available: <http://www.diabetes.co.uk/blood-glucose/blood-glucose-monitoring-diaries.html>.
- [11] M. Benaissa, B. Malik, A. Kanakis and N. P. Wright, "Tele-Healthcare for Diabetes Management: A Low Cost Automatic Approach".
- [12] [Online]. Available: http://whqlibdoc.who.int/publications/2010/9789241564144_eng.pdf?ua=1.
- [13] [Online]. Available: <http://www.americantelemed.org/about-telemedicine/what-is-telemedicine>.
- [14] P. J. Heinzelmann, N. E. Lugn and J. C. Kvedar, "Telemedicine in the Future," *Journal of Telemedicine and Telecare*, vol. 11, no. 8, p. 384–390, 2005.
- [15] [Online]. Available: <http://www.setrc.us/index.php/what-is-telehealth/benefits-of-telehealth-telemedicine/>.
- [16] R. Wootton, L. S. Jebamani and S. A. Dow, "Telemedicine and Underserved Populations.," *Journal of Telemedicine and Telecare*, vol. 11, no. 5, pp. 221-224, 2005.

- [17] S. Allemann, C. Houriet, P. Diem and C. Stettler, "Self-monitoring of blood glucose in non-insulin treated patients with type 2 diabetes: a systematic review and meta-analysis," *Current Medical Research & Opinion*, vol. 25, no. 12, pp. 2903-2913, 2009.
- [18] J. Polisena, K. Tran, K. Cimon, B. Hutton, S. McGill and K. Palmer, "Home telehealth for diabetes management: a systematic review and meta-analysis," *Diabetes, Obesity and Metabolism*, vol. 11, no. 10, pp. 913-930, 2009.
- [19] A. Rao, P. Hou, T. Golnik, J. Flaherty and S. Vu, "Evolution of Data Management Tools for Managing Self-Monitoring," *Journal of Diabetes Science and Technology*, vol. 4, no. 4, pp. 949-957, 2010.
- [20] R. Wootton, "Telemedicine Support for the Developing World," *Journal of Telemedicine and Telecare*, vol. 14, no. 3, pp. 109-114, 2008.
- [21] J. Craig and V. Patterson, "Introduction to the Practice of Telemedicine," *Journal of Telemedicine and Telecare*, vol. 11, no. 1, pp. 3-9, 2005.
- [22] R. Currell, C. Urquhart, P. Wainwright and R. Lewis, "Telemedicine versus face to face patient care: effects on professional practice and health care outcomes," *Cochrane Database of Systematic Reviews*, 2010.
- [23] M. Bajpai, "Telemedicine: A Review," WebMedCentral, 2012.
- [24] [Online]. Available: <https://abbottdiabetescare.co.uk/our-products/freestyle-libre>.
- [25] E. Cengiz and W. Tamborlane, "A Tale of Two Compartments: Interstitial Versus Blood Glucose Monitoring," *Diabetes Technology & Therapeutics*, 2009.
- [26] Diabetes Research in Children Network (DirecNet) Study Group, "Youth and Parent Satisfaction with Clinical Use of the GlucoWatch G2 Biographer in the Management of Pediatric Type 1 Diabetes," *Diabetes Care*, 2005.
- [27] [Online]. Available: <http://www.ihealthlabs.com/glucometer/wireless-smart-gluco-monitoring-system/>.
- [28] [Online]. Available: <http://www.onetouch.com/veriosync>.
- [29] "Bluetooth Core Specification 4," 2010. [Online]. Available: https://www.bluetooth.org/en-us/specification/adopted-specifications?_ga=1.179896610.786830036.1426524894.
- [30] [Online]. Available: <http://www.walgreens.com/store/c/walgreens-blood-glucose-test-strips/ID=prod396869-product>.
- [31] [Online]. Available: <http://www.walgreens.com/store/c/onetouch-ultra-blue-test-strips/ID=prod1026579-product>.
- [32] "Intel® Edison Development Platform Product Brief," 2014. [Online]. Available: <https://communities.intel.com/docs/DOC-23139>.
- [33] [Online]. Available: <http://arduino.cc/en/Main/arduinoBoardUno>.

- [34] "Raspberry Pi B+ Data Sheet," [Online]. Available: <https://www.adafruit.com/datasheets/pi-specs.pdf>.
- [35] "Intel Edison Breakout Board Hardware Guide," 2015. [Online]. Available: <https://communities.intel.com/docs/DOC-23252>.
- [36] [Online]. Available: <http://redbearlab.com/bleshield/>.
- [37] [Online]. Available: http://www.circuitsathome.com/arduino_usb_host_shield_projects.
- [38] [Online]. Available: <http://www.adafruit.com/product/1327>.
- [39] [Online]. Available: http://www.conrad-electronic.co.uk/ce/en/product/1276274/?WT.srch=1&WT.mc_id=sea_shopping&gclid=CjwKEAjwr6ipBRCM7oqrj6O30jUSJACff2WHFsdqdtXPbGTPOIYmvh_ZrOKqtphpVMRsUnZM4DmZZBoCFvbw_wcB.
- [40] [Online]. Available: http://www.gearbest.com/development-boards/pp_62975.html?.
- [41] [Online]. Available: <http://www.dawnrobotics.co.uk/bluetooth-4-0-low-energy-ble-shield-v2-1/>.
- [42] [Online]. Available: <http://www.maplin.co.uk/p/raspberry-pi-model-b-512-mb-mainboard-n52dv>.
- [43] [Online]. Available: <http://www.adafruit.com/products/1327>.
- [44] [Online]. Available: <http://arduino.cc/en/Main/FAQ>.
- [45] [Online]. Available: <http://medea.mah.se/2013/04/arduino-faq/>.
- [46] [Online]. Available: <https://www.sparkfun.com/categories/240?page=all>.
- [47] [Online]. Available: <http://arduino.cc/en/Tutorial/Sketch>.
- [48] [Online]. Available: <http://arduino.cc/en/reference/libraries>.
- [49] [Online]. Available: <http://www.circuitsathome.com/products-page/arduino-shields>.
- [50] [Online]. Available: https://github.com/felis/USB_Host_Shield_2.0.
- [51] [Online]. Available: https://github.com/YuuichiAkagawa/USBH_MIDI.
- [52] [Online]. Available: <http://www.oshwa.org/2013/11/19/new-faq-on-usb-vendor-id-and-product-id/>.
- [53] [Online]. Available: <http://www.makelinux.net/ldd3/chp-13-sect-1>.
- [54] [Online]. Available: <http://www.beyondlogic.org/usbnutshell/usb4.shtml>.

- [55] [Online]. Available:
<http://web.archive.org/web/20080117000828/http://bluetooth.com/Bluetooth/Technology/Works/>.
- [56] Bluetooth Essentials for Programmers, 2007.
- [57] "Bluetooth Core Specification 4.0," [Online].
- [58] [Online]. Available: http://www.eetimes.com/document.asp?doc_id=1278966.
- [59] [Online]. Available: http://www.ebay.co.uk/itm/HC-05-Master-Slave-Module-Integration-WLAN-Wireless-Serial-Pass-Through-Module-/251930200173?pt=LH_DefaultDomain_3&hash=item3aa835c86d.
- [60] [Online]. Available: <http://www.broadband.se/en/shop/bluetooth/bluetooth-smart/devkit-bluetooth-smart/csr10x0-starter-development-kit/>.
- [61] "HC Serial Bluetooth Products - User Instruction Manual," [Online]. Available:
http://www.rcscomponents.kiev.ua/datasheets/hc_hc-05-user-instructions-bluetooth.pdf.
- [62] [Online]. Available: <http://redbearlab.com/bleshield/>.
- [63] [Online]. Available: <https://www.nordicsemi.com/eng/Products/Bluetooth-Smart-Bluetooth-low-energy/nRF8001>.
- [64] [Online]. Available: <http://redbearlab.com/getting-started-bleshield>.
- [65] [Online]. Available: <http://arduino.cc/en/Reference/SPI>.
- [66] [Online]. Available: <http://www.circuitsathome.com/mcu/running-multiple-slave-devices-on-arduino-spi-bus>.
- [67] [Online]. Available:
<https://www.csrsupport.com/download/46402/uEnergy%20Starter%20Dev%20Kit%20Product%20Brief%20CS-309362-DC.pdf>.
- [68] [Online]. Available: <http://www.csr.com/products/csr-energy-software-development-kit-sdk>.
- [69] [Online]. Available: https://wiki.csr.com/wiki/Main_Page.
- [70] [Online]. Available:
http://pubs.opengroup.org/onlinepubs/9699919799/basedefs/V1_chap04.html#tag_04_15.
- [71] [Online]. Available: <http://stackoverflow.com/questions/20822821/what-is-a-unix-timestamp-and-why-use-it>.
- [72] [Online]. Available: http://www.pjrc.com/teensy/td_libs_Time.html.
- [73] [Online]. Available: <https://itunes.apple.com/gb/app/ble-apps/id826487282>.
- [74] [Online]. Available: <https://source.android.com/>.

- [75] [Online]. Available: <http://www.businessinsider.com/androids-share-of-the-computing-market-2014-3?IR=T>.
- [76] [Online]. Available: <http://developer.android.com/tools/help/index.html>.
- [77] [Online]. Available: <http://developer.android.com/tools/studio/index.html>.
- [78] [Online]. Available: <http://developer.android.com/reference/android/app/Activity.html>.
- [79] [Online]. Available:
<http://developer.android.com/reference/android/view/View.OnClickListener.html>.
- [80] [Online]. Available: <http://developer.android.com/guide/topics/ui/notifiers/toasts.html>.
- [81] [Online]. Available: <http://developer.android.com/guide/topics/data/data-storage.html>.
- [82] [Online]. Available: <http://developer.android.com/guide/topics/ui/declaring-layout.html>.
- [83] [Online]. Available: <http://developer.android.com/guide/topics/manifest/manifest-intro.html>.
- [84] [Online]. Available: <https://github.com/RedBearLab/Android>.
- [85] [Online]. Available: <http://developer.android.com/reference/android/widget/Adapter.html>.
- [86] [Online]. Available: <http://developer.android.com/training/basics/data-storage/files.html>.
- [87] [Online]. Available: <https://developer.android.com/samples/BluetoothChat/index.html>.
- [88] [Online]. Available: <https://developer.android.com/guide/topics/connectivity/bluetooth-le.html>.
- [89] [Online]. Available: <http://androidplot.com/>.
- [90] [Online]. Available: <http://nfc-forum.org/what-is-nfc/about-the-technology/>.
- [91] [Online]. Available: <http://www.smartcardalliance.org/publications-contactless-payments-what-who-why/>.
- [92] [Online]. Available: <http://www.cnet.com/uk/news/nfc-enabled-cell-phones-to-hit-416-million-shipments-report/>.
- [93] [Online]. Available: http://kimtag.com/s/nfc_tags.
- [94] [Online]. Available: <http://www.adafruit.com/products/789>.
- [95] [Online]. Available: <http://blog.contus.com/near-field-communication-nfc-vs-bluetooth-first-ever-tabulated-comparison/>.

9 - Appendix

9.1 – Receiving Readings Walkthrough

Below is a walkthrough of the process for receiving readings from a Contour Next USB glucometer to the Arduino. This glucometer was chosen due to its popularity, detailed documentation and being easy to interface with. Although this walkthrough is specific to this glucometer, it can also be used as a general guide for all glucometers.

To begin data transfer from the glucometer to the Arduino an acknowledgement must be successfully sent from the Arduino to the glucometer (OUT transfer). The out transfer function was originally designed to send MIDI messages to be played as sound but was edited so send an array of length declared by the function. The acknowledgement is in the form of an array containing:

```
{0x00, 0x00, 0x00, 0x01, 0x06}
```

These are ASCII characters, displayed in hexadecimal value, and contain three null characters, a start-of-header (SOH) character and an acknowledgement (ACK) character. The first time the glucometer receives the acknowledgement it replies with a header (IN transfer), which it writes to a buffer. The IN transfer function was originally for receiving MIDI data (such as: which note is pressed and the notes velocity) from a MIDI keyboard. It was modified to read data and write it to an array. An example header is:

```
<STX>1H|\^&| |yheQqp|Bayer6300^01.07\01.04\24.12\01.00.A^6301-1C0CE72^0000-  
^C0CE72|A=1^C=23^G=en,da\de\en\es\fr\it\nl\no\pt\sl\fi\sv\el\hr^I=0200^R=0^S=01^U=1^V=206  
00^X=070070070070180130180250070130^Y=360126099054120054252099^Z=1^r=1|246| |||P|1  
|20110101010755<ETB>
```

STX is start transmission, ETB is end transmission block. The header contains information about the glucometer and the information contained on the glucometer, such as the interval between reminders to test, the glucose level boundaries to be considered a high or low reading, the number of readings on the device, etc. After the initial acknowledgement message is sent, each subsequent acknowledgement message receives a single reading. An example reading is:

```
<STX>5R|83|^ ^^Glucose|11.7|mmol/L^P| |F/P0/M0/T1| |20140420200739<CR><ETB>
```

<CR> is carriage return. This reading contains the number of the reading (83), the type of reading (glucose), the value read (11.7), the units (mmol/L) and the date/time (20/04/14 at 20:07:39). These readings can also be insulin or carbohydrate instead of glucose. Insulin readings state the type of insulin (fast-acting, slow-acting, etc.) and units. Carbohydrate readings simply state the amount in grams. The buffer that receives the readings needs to be cleared before the next reading is taken, or else the buffer will overflow and the glucometer will display an error and cease transmission.

Once the final reading has been sent from the glucometer to the Arduino, the next acknowledgement message receives an end of transmission (EOT) message ASCII character.

After each reading is received the Arduino sketch processes the reading to receive the vital information, namely: type of reading, value and date. It then writes this to a buffer that holds the data to be sent to the application via Bluetooth.

9.1 - Program Code

The code for the project has been uploaded onto GitHub and can be found at:

<https://github.com/bentrevelt>

All additional instructions and information specific to each code is contained within each 'readme'.

The code for the hardware side of the Glucoduino project can be found at:

<https://github.com/bentrevelt/Glucoduino>

This is the code that must be uploaded to the Arduino. It requires the Arduino IDE, an Arduino Uno, a USB host shield and a compatible glucometer (Contour NEXT USB or Contour NEXT Link).

The code for the software side of the Glucoduino project can be found at:

<https://github.com/bentrevelt/Glucoduino-Classic-Bluetooth-Application>

<https://github.com/bentrevelt/Glucoduino-Bluetooth-Low-Energy-Application>

This code must be ran on an Android phone. It requires Android Studio and a compatible Android phone. For the BLE application, the smartphone requires a BLE chip and Android version 4.3 or higher.

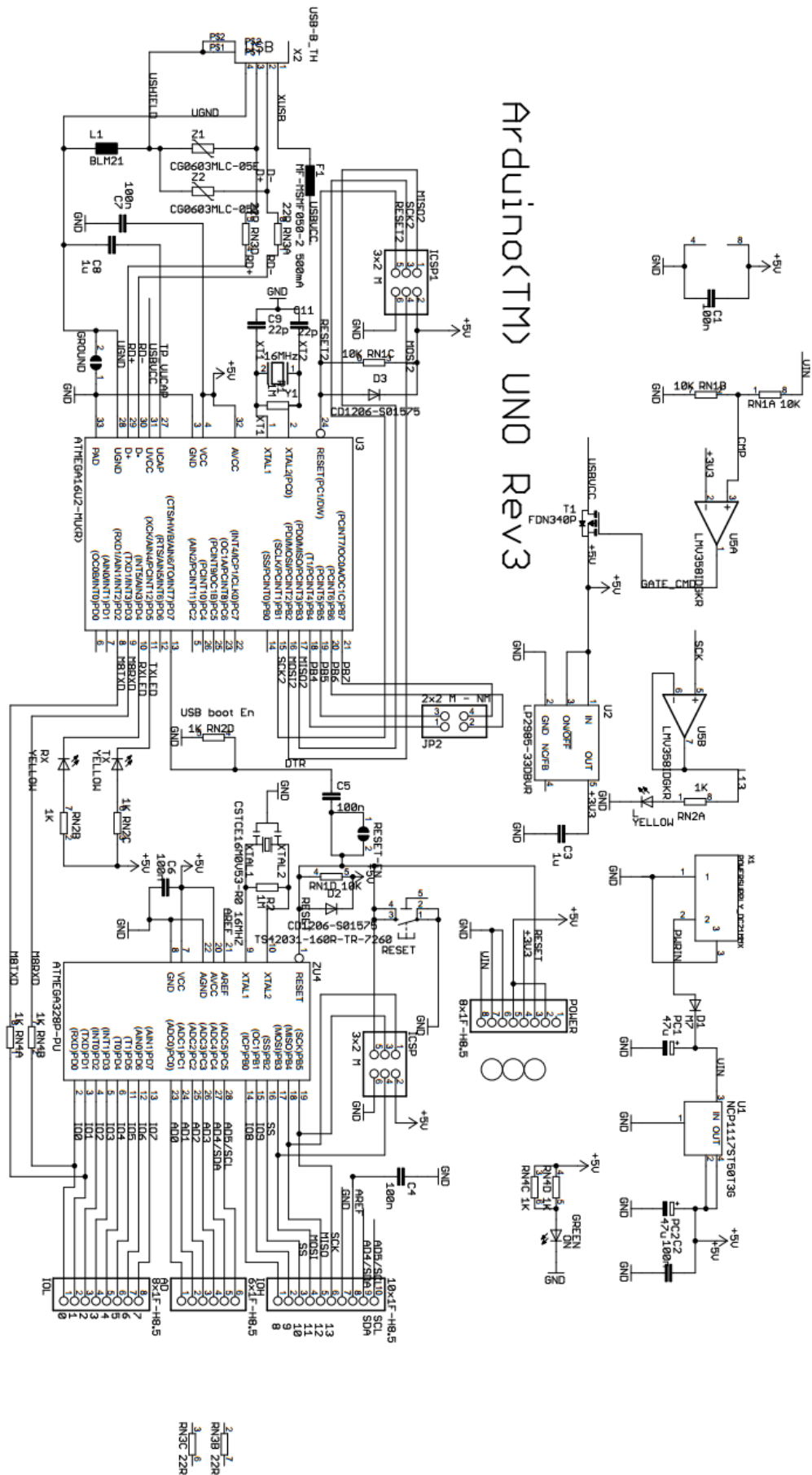
The code for using the CSR Bluetooth chip can be found at:

<https://github.com/bentrevelt/Glucoduino-CSR-Chip>

<https://github.com/bentrevelt/Glucoduino-CSR-Example>

The code for the CSR chip must be ran on CSR uEnergy SDK. It requires a CSR development kit. The example code shows how to send a reading to the CSR chip and requires the 'Time' library to be installed.

9.2 – Arduino Schematic



Progress Report 2

Automated low cost device for the remote monitoring of Glucose for type-1 Diabetes using mobile phones.

1 – Project Description

The project is to research, design and develop an automated low cost device that aids in the monitoring of glucose levels in Diabetes sufferers (primarily type-1 Diabetes) in combination with a complimentary smartphone application. The device will connect to the Glucometer, download the glucose readings and transfer them to the application. The application will offer a number of features, the main focus is periodically backing up measured glucose levels to a webserver which the patient's medical staff can remotely monitor. Other features will include displaying the data for the patient to view, reminding the patient when to take a glucose reading and more.

2 - Final Specification

The project is to create a 'system' for the remote monitoring of diabetes wherein data read by the patient's glucometer can be remotely viewed by the patient's medical staff. It consists of two main parts: the device and the application.

The device's functionality can be broken down into three main parts: sensing, processing and communicating. The sensing functionality is carried out by connecting the device to the patient's glucometer via a USB serial connection. The device will wait for a connection to be made and then receive the data (glucose levels, time and date, device ID, etc.) from the glucometer and store it into its memory to be processed. The device will then process the received data into a standard format ready for transfer. This is done by manipulating data using the glucometer's unique ID to identify both the patient and the type of device. Different devices have their data formatted in different ways, therefore the processing functionality must be able to distinguish which device the user has connected. Finally, the communication functionality will be carried out by the device by transferring data wirelessly via Bluetooth to the application.

The application's function can also be broken down into the same three main parts: sensing, processing and communicating. The sensing will be done via the Bluetooth communication functionality of the device. The glucometer data stored on the device will be transferred to the application. The processing will be done by analysing the data from the glucometer and displaying it to the user. The user will be able to view the data received from the device on the application either as individual readings in a list format or a summary over a certain timescale in a graphical format. The communication functionality will be carried out by uploading the data on the application to a webserver where it can be viewed by the patient's medical staff.

This project only covers the ‘system’ from the glucometer to the application. It does not include the webserver, which has already been created, with the exception of uploading glucometer data from the application to the webserver.

3 - Current Progress

3.1 – Diabetes Research

During the initial research into diabetes it was found that there are 366 million diabetics worldwide, this is expected to increase to 522 million by 2030 [1]. Globally, around 80,000 children are diagnosed every year [2]. There are a number of detrimental increased health risks caused by diabetes, such as: heart problems, kidney disease, loss of sight, poor circulation, etc. [3]. In 2010 the estimated global cost of diabetes was \$376 billion, equal to 11.6% of the global health spending [4]. This cost can be reduced by proper awareness, lifestyle choices and education programmes [5].

The traditional method of measuring diabetes is by recording the glucose readings any additional information (such as exercise or illness) within a ‘glucose diary’ [6], the patient then visits their medical staff every 2-4 months to receive advice on diet and exercise and possible change in insulin dosages [7]. This is deemed inefficient as the patient may forget to monitor their glucose levels and that the patient does not receive information on a regular enough basis.

3.2 – Currently Available Solutions

Research into currently available solutions was then carried out. Firstly, multiple applications were found that replaced the standard ‘glucose diary’, for example Glooko [8]. Applications such as these have multiple advantages over the standard pen and paper glucose diary such as:

- As they are on your phone, they are always at hand
- Adding new readings is done in a few simple button presses.
- Are able to hold a much larger record of data and are able to synchronise it so it is available on other devices.
- Can set reminders to send a notification to the patient to remind them to take a reading
- Can take data from different time ranges and display in an easy to read graphical form

There are also glucometers with Bluetooth built into them. Such as the OneTouch VerioSync [9] and iHealth Labs' BG5 [10].



Figure 1 – iHealth Labs' BG5

These have the advantage of being able to transfer data from the glucometer directly to the smartphone, which removes the need to manually enter data into the applications. However, these have numerous disadvantages:

- Both of them require you use the application unique to their device, forcing patients to switch from an application they may prefer.
- The VerioSync application is only available for iOS, the BG5 application is only available for Android and iOS and neither is available for Windows Phone.
- Both of them use Bluetooth 3.0, not BLE (Bluetooth Low Energy), therefore need to be charged more often.
- Both are only available in the USA.
- The BG5 doesn't display test results on screen.
- They require unique test strips which are usually more expensive, as standard strips are \$0.5 each [11] while the VerioSync strips are \$1.6 each [12].

The device in this project not designed to compete with other Bluetooth enabled glucometers. It will be designed to be compatible with as many existing glucometers as possible. This retrofitting of functionality will allow users to carry on using their preferred existing devices whilst adding enabling it to act as a glucometer with Bluetooth built in.

3.3 – Device Research

A chosen device was then researched and was narrowed down to 3 devices: Raspberry Pi B+, Arduino Uno and Intel Edison. Comparisons were made of the 3 devices to decide which is the best suited for the project. For a fair comparison, as the Arduino and Raspberry Pi B+ do not have Bluetooth built in they are compared with the additional cost of adding BLE functionality and the Intel Edison does not have a USB port without the ‘breakout board’.

	Arduino Uno [13] w/ BLE shield [14]	Raspberry Pi B+ [15] w/ BLE module [16]	Intel Edison [17] w/ Breakout Board [18]
Cost	\$55	\$50	\$70
Size	7.5 * 6.5 cm	8.5 * 5.5 cm	4.5 * 3 cm
RAM	2 kB	512 MB	1 GB
Memory	32 kB	Micro SD card	4 GB
I/O	1 USB	4 USB	2 USB
OS	None	Linux	Linux
IDE	Arduino	Any Linux supported	Arduino, Eclipse, XDK

Table 1 – Comparison of 3 devices

3.4 – Intel Edison



Figure 2 – Intel Edison with Breakout Board

The Intel Edison, from here on referred to as Edison, was chosen as the device to be used for the project. This is because even though it costs more than the Arduino and Raspberry Pi B+, it is superior in every other aspect. The smaller form factor allows it to be transported easier. Higher RAM means data processing is done faster and greater memory is used to store larger amount of data. The Linux OS is very flexible and allows the board to perform a wide variety of features and be programmed in a number of different languages, as shown by the three available IDEs.

The Edison can be powered by a compatible 400mAh battery pack [19] allowing for portability.

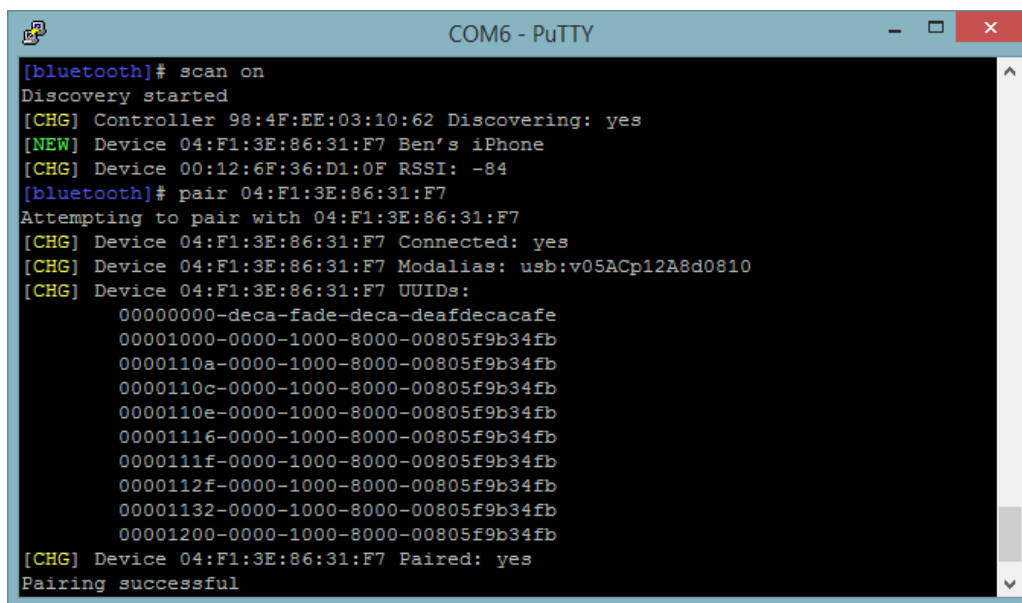
3.5 – Bluetooth Communication

Research was carried out on Bluetooth communication using the Arduino IDE and using the Linux OS command lines.

Bluetooth connectivity using Arduino IDE can be done by using a Bluetooth shield or by using a Bluetooth module. When using a shield, the libraries for that shield are imported into the IDE and the appropriate methods are used. When using a module, it is connected to the Tx and Rx pins of the board and Arduino's serial communication method is used. However, these two methods of Bluetooth functionality can only be carried out using the Arduino Board for the Edison.

Bluetooth through the Linux uses the official Bluetooth protocol stack, BlueZ [20]. This is done by connecting to the Edison using a terminal emulator, such as PuTTY [21], at 115200 baud and by passing the appropriate commands to establish a connection between devices. Currently the Edison can only connect and pair with devices and cannot transfer data between them.

See section '4.4 – Bluetooth' for issues.



```
[bluetooth]# scan on
Discovery started
[CHG] Controller 98:4F:EE:03:10:62 Discovering: yes
[NEW] Device 04:F1:3E:86:31:F7 Ben's iPhone
[CHG] Device 00:12:6F:36:D1:0F RSSI: -84
[bluetooth]# pair 04:F1:3E:86:31:F7
Attempting to pair with 04:F1:3E:86:31:F7
[CHG] Device 04:F1:3E:86:31:F7 Connected: yes
[CHG] Device 04:F1:3E:86:31:F7 Modalias: usb:v05ACp12A8d0810
[CHG] Device 04:F1:3E:86:31:F7 UUIDs:
00000000-deca-fade-deca-deafdecacafe
00001000-0000-1000-8000-00805f9b34fb
0000110a-0000-1000-8000-00805f9b34fb
0000110c-0000-1000-8000-00805f9b34fb
0000110e-0000-1000-8000-00805f9b34fb
00001116-0000-1000-8000-00805f9b34fb
0000111f-0000-1000-8000-00805f9b34fb
0000112f-0000-1000-8000-00805f9b34fb
00001132-0000-1000-8000-00805f9b34fb
00001200-0000-1000-8000-00805f9b34fb
[CHG] Device 04:F1:3E:86:31:F7 Paired: yes
Pairing successful
```

Figure 3 – Pairing to a Bluetooth device using Linux command lines

3.6 – Android Research

It was decided the application would be made for Android phones. This is due to Android being the most widely used mobile OS (greater than Windows and iOS + Mac OS combined) [22], being open source [23] and having a large set of free development tools, such as sample code, emulators, debuggers documentation and tutorials [24].

3.7 – Android Application

The application was created using the official IDE for Android, Eclipse using the ADT (Android Development Tools) plug-in [25]. This is whilst Android's own IDE, Android Studio, was still in beta. However, as of December 2014, Android Studio has now been released as is the official IDE. See section '4.5 – Android' for more.

Android applications consist of two main parts, Java and XML. The Java defines the functionality of the application. The XML is for defining the layout of the objects displayed to the user.

The XML is very flexible and allows for the objects to be displayed in multiple formats, such as RelativeLayout, where items are defined in relation to other objects, LinearLayout, where items are aligned horizontally or vertically, or in a ListView. ListView is the most important layout as it displays items in a scrollable list, with as many entries are available. The layout of each element in the list can be customised and 'adapted' to the data for each element using the adapter class, for example showing levels of glucose, time and date, etc.

User input takes the form of 'onClickListen' methods. When the appropriate object, i.e. a button or an item in a list, is pressed the application will perform the code inside the method for that object, i.e. display a message to the user in the form of a 'toast' or move to another 'activity'.

Each 'page' of the application is called an 'activity' and they can be changed between using 'intents'. Intents can also pass data between activities and delegate responsibilities to another application, i.e. open a URL in a browser.

Data can be stored locally in an application by using 'SharedPreferences'. This is done by storing data, such as a username and password, with their own unique 'key'. This data can be used when uploading the data to the webserver, ensuring it is uploaded to the correct patient.

An application was created using the above Android features. It consists of a login page, a 'home' page which displays the current Bluetooth connectivity status and a scrollable list of data entries, which is currently populated by dummy data using a POJO (Plain Old Java Object) [26]. When the Bluetooth connectivity is complete, the data from the device will fill the Java objects and be displayed in the ListView.

3.8 – Device Side Programming

Due to the pending Bluetooth update for the Edison, see '4.4 – Bluetooth', an Arduino Uno was purchased along with a USB Host Shield [27] and a BLE Shied [28] to allow for a temporary solution to provide the same functionality.

Research then continued to the transferring of data from the glucometer over USB. As each glucometer transfers the data via USB in a different format, they need to be looked at on an individual basis. The unique ID differentiating each glucometer was used to ensure that the data is formatted correctly.

4 - Issues

4.1 - Delays

There have been several issues faced during the project which have either delayed aspects of the project, changed how in what order aspects were undertaken or changed parts of the project itself. As detailed in the risk register of Progress Report 1, to minimise the delays caused by waiting for parts to be delivered they were ordered as soon as possible. However, even when ordered as soon as the Edison was decided on, there was still a while before it arrived which delayed the writing of the program.

4.2 - Documentation

As the Edison is a relatively new board (released October 2014) it has a much smaller amount of online resources compared to the Raspberry Pi and the Arduino. The documentation provided by Intel is also minimal as the board is still being actively developed with major releases still forthcoming. This has caused research to be difficult, slowing the progress made on researching the board and making programming more difficult.

4.3 - Forums

Due to Intel's minimal documentation, reliance has to be made on the Edison forums for additional resources on functionality. Although forums are a good way to find additional resources, they are not optimal as there are usually more people asking questions than there are posting solutions to these problems and some of these solutions are less than adequate, assuming the person asking the question has as much knowledge as them.

4.4 - Bluetooth

The main issue faced was that although the Edison has built-in Bluetooth technology, release 1 of the Edison is only able to pair and connect with other Bluetooth enabled devices. It is unable to transmit data between the two devices. Intel staff have repeatedly posted on the Edison forums stating that release 2 will be available before the end of 2014 and will include an unspecified updates to the Bluetooth functionality. To avoid delays, a compromise was planned to use the Arduino Breakout Board and a Bluetooth shield to temporarily add Bluetooth data transfer functionality. However, further research revealed that none of the Bluetooth shields have libraries that are compatible with the Edison.

As transmitting data over Bluetooth is a key feature of this project and the original time-scale had planned to complete all of the programming of the device side before 2014. Due to the incomplete Bluetooth functionality offered by the Edison, major revisions to the time-scale had to be made.

4.5 – Android

After revisions to the time-scale, significant progress was made on the Android application. As the Bluetooth functionality has not been implemented, the app merely displayed dummy data created by the application itself. The application was programmed in Eclipse using the ADT plug-in. This was the official Android IDE while Android's stand-alone IDE was being developed. Mid December 2014 Android released their stand-alone IDE, named Android Studio, and support for the ADT plug-in has now ceased. Although the application can still be maintained through the ADT plug-in, in order to ensure the application maintains compatibility with future Android releases, it should be ported to Android Studio, which will take additional time unaccounted for.

4.6 - Changes in Time Scale

The original scale is displayed in section '6.1 – Original Timescale'. The 'Initial Subject Research' and 'Hardware Research' sections were finished on time. Issues were raised during the 'Programming Research' section, which delayed the 'Writing Program' section. Although the risk register stated ordering components early to reduce delays to delivery time, the Edison was still not delivered until November. After the Edison arrived it was unable to be debugged over WiFi (the only currently available method) through Eclipse while at University, due to the University WiFi's security, this caused a switch to the Arduino IDE and a delay in programming. During the research it was found that Edison's built-in Bluetooth functionality had not been fully implemented in the current release, and that release 2, containing the improvement to the Bluetooth functionality, will be released by the 'end of 2014'. In order to avoid delays, research was done into using an Arduino Bluetooth shield to provide a temporary solution, however after further research it was found that none of the Bluetooth shield libraries are currently compatible with the Edison. To ensure completing the project on time, the time frame allotted to completing the application was switched with the time frame for programming the device side. As Android have changed the official IDE from the ADT plug-in to Android Studio, time must now be made to port the code over to ensure the application is compatible with future releases of Android.

4.7 - Revised Time Scale

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
A																				
B																				
C																				
D																				
E																				
F																				
G																				
H																				
I																				
J																				
K																				

- A. Initial Subject Research
- B. Hardware Research
- C. Programming Research
- D. Writing Program
- E. Debugging and Testing Program
- F. Application Research
- G. Application Design
- H. Writing Application
- I. Debugging and Testing Application
- J. Final Report Preparation
- K. Presentation Preparation

5 – References

- [1] - <http://www.idf.org/diabetesatlas/news/fifth-edition-release>
- [2] - <http://care.diabetesjournals.org/content/37/7/2034>
- [3] - https://www.health.ny.gov/diseases/conditions/diabetes/problems_caused_by_diabetes.htm
- [4] - <http://www.idf.org/millions-unite-diabetes-awareness-world-diabetes-day-2010>
- [5] - <http://www.who.int/mediacentre/factsheets/fs236/en/>
- [6] - <http://www.diabetes.co.uk/blood-glucose/blood-glucose-monitoring-diaries.html>
- [7] - Tele-Healthcare for Diabetes Management: A Low Cost Automatic Approach
- [8] - <https://glooko.com/>
- [9] - <http://www.onetouch.com/veriosync>
- [10] - <http://www.ihealthlabs.com/glucometer/wireless-smart-gluco-monitoring-system/>
- [11] - <http://www.walgreens.com/store/c/walgreens-blood-glucose-test-strips/ID=prod396869-product>
- [12] - <http://www.walgreens.com/store/c/onetouch-ultra-blue-test-strips/ID=prod1026579-product>
- [13] - <http://arduino.cc/en/Main/arduinoBoardUno>
- [14] - <http://redbearlab.com/bleshield/>
- [15] - <https://www.adafruit.com/datasheets/pi-specs.pdf>
- [16] - <http://www.adafruit.com/product/1327>
- [17] - <https://communities.intel.com/docs/DOC-23139>
- [18] - <https://communities.intel.com/docs/DOC-23252>
- [19] - <https://www.sparkfun.com/products/13037>
- [20] - <http://www.bluez.org/>
- [21] - <http://www.chiark.greenend.org.uk/~sgtatham/putty/>
- [22] - <http://www.businessinsider.com/androids-share-of-the-computing-market-2014-3?IR=T>
- [23] - <https://source.android.com/>
- [24] - <http://developer.android.com/tools/help/index.html>
- [25] - <http://developer.android.com/tools/help/adt.html>
- [26] - <http://crunchify.com/create-simple-pojo-and-multiple-java-reflection-examples/>
- [27] - <http://www.banggood.com/USB-Host-Shield-Compatible-Google-Android-ADK-Support-UNO-MEGA-p-90129.html>
- [28] - <http://redbearlab.com/bleshield/>

6 – Index

6.1 – Original Gantt Chart

Task	Week Number																			
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
A																				
B																				
C																				
D																				
E																				
F																				
G																				
H																				
I																				
J																				
K																				

- A. Initial Subject Research - researching into Diabetes, current available devices, specification development
- B. Hardware Research - researching different available hardware to see which is the best suited for the project
- C. Programming Research - researching which method of programming to use and how to execute the desired program
- D. Writing Program - writing of the program via the desired method
- E. Debugging and Testing Program - ensuring program does not have any bugs and that all potential processing by the program delivers appropriate results
- F. Application Research - researching how applications are designed and programmed
- G. Application Design - designing the application to fit the specification
- H. Writing Application - writing the application to fit the requirements of the specification
- I. Debugging and Testing Application - ensuring the application does not have any bugs and that all potential processing done by the application delivers appropriate results
- J. Final Report Preparation - writing the final report
- K. Presentation Preparation - preparing to deliver the presentation