CS3099 Final Submission

Group A-3

April 2019

University of St. Andrews
Houston Owen
Chance Carey
Jet Chew
Robin Finley Legg
Gopal Juneja

Abstract

This report summarizes the efforts of this group to fulfill the specification of developing software for a Microbit device that allows it to be a moderately competent smart phone replacement for social functionality. This report outlines every decision and deliberation made throughout development of the Microbit. In summary, our final product is able to wirelessly - through the use of an intermediary Microbit that proxies serial and radio communications - is able to retrieve information from the internet, as well as communicate to any nearby Microbit that implements the intergroup protocol.

Declaration

We declare that the material submitted for assessment is our own work except where credit is explicitly given to others by citation or acknowledgement. This work was performed during the current academic year except where otherwise stated. The main text of this project report is 9,584 words long, including project specification and plan.

In submitting this project report to the University of St Andrews, we give permission for it to be made available for use in accordance with the regulations of the University Library. We also give permission for the report to be made available on the Web, for this work to be used in research within the University of St Andrews, and for any software to be released on an open source basis.

We retain the copyright in this work, and ownership of any resulting intellectual property.

Contents

1 Introduction

Lead author Gopal Juneja and Robin Legg

1.1 Aim

Lead author Gopal Juneja
As an attempt at reversing the problematic affair of modern societies addiction to smartphones and its various engaging social media applications, this group has worked on the development of the Microbit to fulfill the requirements of modern society while trying to reduce the usage of other devices. There are multiple factors that come into play when the Microbit is being pitted against highly advanced daily-use devices of today's day and age. Apart from the basic requirement which was provided to us, we had to sensible choose additional features and establish a communication protocol, which would ensure successful functioning of the device and at the same time, solve the issue at hand.

1.2 Success

Lead author Robin Legg
Through the development of the Microbit the team feels that we have had moderate success in achieving this goal. This has been backed up by our user study which yielded interesting results which helped us reach this conclusion.

1.3 Key features

Lead author Robin Legg, Gopal Juneja
The key features that have helped us reach this aim are as follows:

- Twitter integration
- Weather information integration
- Sports Scores integration
- Easy to use user interface
- Simple Web app
- Encryption
- Piano
- Lock Screen

2 Project details

2.1 Microbit operation

Lead author Chance Carey
Edited by Robin Legg.

2.1.1 Navigation

Navigation on a device with as limited input and output options as the Microbit has been a difficult endeavor and so compromises were made in order to deliver what we would consider to be the best possible user experience.

Technically, there are seven possible inputs to the Microbit: for both the left (A) and right (B) buttons, you can click, long click, hold them, and additionally hold both A and B together. Realistically, however, there are only five options, as the user experience suffers significantly when implementing both long click and hold operations, as its very difficult to time it such that a button is held down long enough for it to not be a click, but not long enough for it to be a hold.

Under these limitations, we devised a stack-based menu system, where at each frame of the stack, the user can navigate between available options, and then choosing one, move to a frame below the current one. Using frames within a stack allows for easy rewind behavior as, to move to a previous menu, the current frame is simply popped from the stack as the history of previous menu choices is already recorded. In implementing this, we chose to use normal clicks of the A and B buttons to navigate between options of a single frame, and having a hold of the B (right) button be the select button, activating that option and potentially pushing a new menu frame to the stack. In order to move back up the stack (exiting the current menu and moving to the previous one), it was natural that we chose to use a hold of the A (left) button. It should be noted that we opted to use a hold action rather than a long click, as the action is immediately taken as soon as the click is confirmed to be longer than a normal click; this is in contrast to long click, where the user is left wondering how long exactly before they can release the button to indicate a long click.

2.1.2 Main menu

Immediately on powering up the Microbit, the user is presented with an initial menu of options. It was important that we use the benefits of the hierarchical menu approach available in order to streamline the user experience by prioritizing nested options in multiple menu frames as opposed to having one menu with more options than would be convenient to navigate. To achieve the desired user experience, we chose to include top level features on the main menu, with sub features nested behind these options. As a result of this, the top level items consisted of the main applications only; Twitter, Weather, Sports, Messaging (called Contacts), and Piano(an optional extra feature only available when the

piano is connected). Within the Twitter and Weather menus were more comprehensive options such as, in the case of Twitter, permitting the user to view tweets either by user, or by hashtag. This was performed by selecting either a user (at, @, icon) or hashtag (# icon) that would only then reveal the options for each choice.

An additional design feature that should be noted is that, on going back up the menu stack (navigating to previous menus from where the selected option had created the current menu), the text of the last selected item would be displayed. There is, however, a special case; when at the top level main menu, trying to navigate backwards instead resets the position to the beginning of the main menu, indicating to the user that the current menu is the highest and first menu on the stack.

2.1.3 Previous design concepts

Throughout development on the Microbit, we considered a multitude of ways for which the user could express intent through interacting with the Microbit. Some of the most promising options included using the GPIO pins (plated pins on the lower edge of the device). We discovered that, when holding on to the ground pin, the user could make contact with the other pins and we would be able to detect which pin they interacted with. This opened up more options than the limited two buttons that we considered to be the only feasible input options initially. On further testing, however, we noticed that using the GPIO pins was very much not a tenable option, as using them resulted in very inconsistent behavior as some pins seemed to barely be detectable (requiring pushing down on them very hard in order to make enough contact with the pin), and that they were difficult to use if the users hands were either too wet or dry.

Another option that was considered was that of using the inbuilt accelerometer in order to interact with the device. In a proof of concept, we allowed the user to navigate left and right through the menus simply by tilting the device in the desired direction. Once again, however, we were unable to reliably and accurately detect user intent, as the firmware only sent events indicating tilt when drastic motions were taken such as rapidly tilting the device by ninety degrees in any direction, which resulted in a difficult user experience.

2.1.4 Decisions on user input

Due to the limitations of the available input methods to the Microbit, constraints had to be placed on the accepted input methods to the device. Foremost amongst these was a limitation on free typing text entry to the device. We were unable to devise a user friendly way of typing messages into the device; direct entry (navigating left or right between alphanumeric characters) was a frustrating experience, and we were unable to implement a system such as T9 input (where the user first selects part of the alphabet before selecting the exact character) as this was similarly ineffective. Thus, after trying multiple options, we decided that text entry was unfeasible and unrealistic, and so compromised

with allowing the user to pre-program the Microbit with preset messages using the server web application. We feel that this approach is more than sufficient, as even reading long text entries on the Microbit is an unpleasant experience due to the extremely small 5x5 LED display.

2.2 Radio communication

Lead author Chance Carey
Edited by Robin Legg

2.2.1 Protocol

One of the most important features of this project was the idea of being able to use the Microbit as a communication device between users, each with their own Microbits. As we are one of a multitude of groups in a supergroup, representatives from each group met and decided on a shared protocol that all Microbits would implement. These Microbits would then be able to communicate equally well to each other through this interface, regardless of the individual functionality or implementation of the Microbits.

As we intended to support functionality specific to our own group in addition to that of the supergroup, the protocol allows a group to specify that the messages sent are custom to the group and no attempt should be made to interpret the message. This approach was satisfactory as it allowed us to fulfil the different requirements that were present for inter- and intra- group communication without having to standardize a protocol for intra-group communication. The following two sections discuss the portions of the protocol relevant to inter-group communication, while the section following those discusses our custom protocol messages.

The two most significant requirements of the protocol were that dynamic device discoverability would be supported, and that any length text messages could be sent from any one Microbit to another nearby.

2.2.2 Discoverability

In order to satisfy the requirement of discoverability, a signal called a heartbeat was chosen. This was a specific packet that contained the serial number of the Microbit, the group number that owned the Microbit, and the individual customizable name of the Microbit.

When the Microbit is powered on, this packet is constructed, and from then on continuously sent at a rate of once per second. Any Microbits within radio range would then be able to receive these packets and learn the exact identity of the Microbit sending the packet. In our implementation of the radio protocol, when our Microbit receives a heartbeat packet, it initializes an object containing all of the information about that Microbit, and then stores it (if not already present) in a table with what is effectively the contact information of all other Microbits.

This approach was chosen from multiple possible approaches. One approach that was discussed and dismissed was that of a request/receive, where a Microbit would request that any Microbits within radio range of it send the original Microbit their contact information. This had the benefits of not constantly sending packets when not needed - the Microbit would only request knowledge of other Microbits when it intended to populate a contact list - however, the downside was, at the worst case scenario of having n Microbits in an area, and all of them requesting contact information, up to n^2 packets would be sent. The heartbeat approach, in comparison, results in a constant, reliable, and passive stream of n packets.

It should be noted that, concurrent with sending heartbeat packets, key exchange packets are also sent; this is discussed in thorough detail later in the encryption section, but is nonetheless worth noting here.

2.2.3 Inter-group messaging

As the heartbeat protocol is passive - all nearby Microbits will be automatically detected at most 1 second after they go online and are in range - this allows for simple and smooth integration of these other Microbits into inter-group messaging. Multiple options were discussed and specified in the supergroup protocol, although naturally, as the Microbit is purposed as a social device, the most relevant part of the protocol was that of sending simple text messages to other Microbits.

The relevant section of the protocol is structured as so. After specifying that the packet is to be interpreted as a supergroup compatible message (as opposed to a custom intra-group packet), the device ID of the origin (sender) Microbit is specified, so that recipient devices know the origin and can match the ID against received heartbeat packets to discover the name of the device. The destination Microbit is then specified, followed by the (encrypted) content of the message itself. Important to note is that, when the destination is unspecified (as being all ASCII 0 characters), the content of the message is not encrypted. In this case all devices should process and possibly display the message as it is intended to be broadcast and read by all devices.

2.2.4 Intra-group messaging

In addition to the inter-group communication protocol, we developed a further protocol specific for radio communication for Microbits of our group that supported custom, specific functionality. This custom functionality is a realization of our intent to permit seamless interactivity between any Microbit within radio range of a Microbit connected to the server via serial.

In essence, we desired functionality that would allow us to treat the server as if it was directly connected to the Microbit, without having to concern ourselves with marshalling and unmarshalling data for transport over radio to an intermediary Microbit that was connected to the server and would act as a relay. This abstracting of the underlying protocol allowed us to rapidly write programs

that would work wirelessly where the users Microbit would be connected only to a power source, but, if in range of a proxy Microbit, would be able to access the internet to retrieve live data.

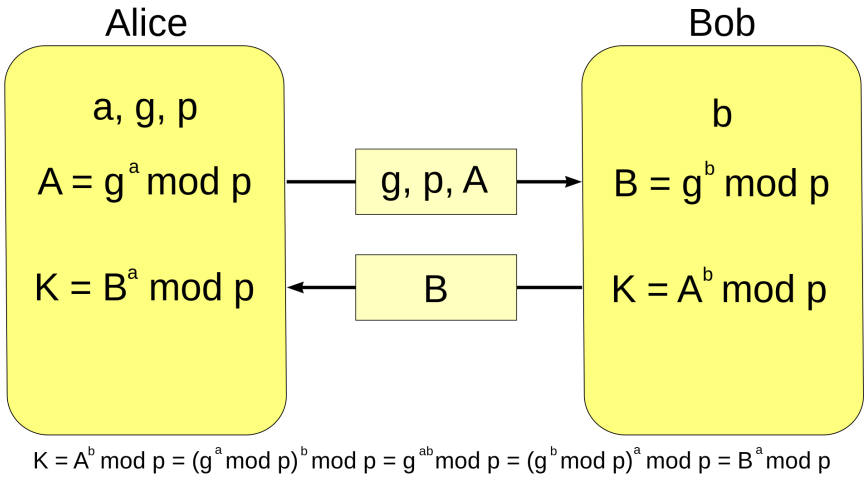
The serial and server aspects of this are discussed further in the Serial communication section. The protocol is as so: a packet is formed containing the identifier of the sending Microbit, as well as the payload which would be remotely sent via serial to the server. This packet is then broadcasted over radio. The Microbit acting in proxy mode then (if in range) receives this packet, noting the identifier of the sending Microbit. After sending the packet via serial to the server and receiving a response, it sends the packet over radio addressed to the initial user Microbit. This allows multiple Microbits to use the same server for internet connectivity without having the responses being handled by the wrong, or all, Microbits. Thus, the intra-group protocol acts, effectively, as a header and wrapper for the serial protocol between the Microbit and the server, as almost a transport layer protocol.

2.3 Encryption

Lead author Houston Owen
Edited by Robin Legg.

2.3.1 Key exchange

There are two parts to encryption of messages between Microbits. The first is the key generation and exchange. As agreed upon in supergroup meetings, Diffie Hellman will be used for the generation of a shared key, which is a single byte. Although this could have been extended, it would add unnecessary complexity to what is essentially a proof-of-concept implementation. As specified in the protocol, key exchange messages are a separate type of message (type 4). The formula used for the calculation of public keys and shared secret keys is detailed in the picture below:



Within our supergroup, the protocol for communication states that a base value of 5 (value for 'g' in the diagram) and a modulus of 23 (value for 'p' in diagram) should be used. Furthermore, the value of the secret key ('a' and 'b' in the diagram) shouldn't exceed 10, as it would cause an Integer overflow and therefore would not encrypt/decrypt correctly.

The original plan of sending out a key exchange message each time the list of nearby Microbits is added to has been changed to send out with the heartbeat messages once every second. This change was necessary as re-flashing or resetting a Microbit within the range of another Microbit would mean that it wouldn't receive a key exchange message from the Microbit which was continuously on. This is a result of the list of nearby Microbits not changing (on the side of the Microbit which remained on), and so no other key exchange messages are sent out. After successful key exchange happens between two Microbits, all further key exchange messages sent out are ignored. This part of encryption also happens automatically, as soon as Microbits enter each others' ranges.

Once generated, the shared keys are associated with a Microbit serial ID in the contacts list, so that the key exchange only has to be performed once per new Microbit encountered. Key exchanges will also have to be re-performed if the Microbit is flashed or reset, as the keys are not stored in persistent memory.

2.3.2 Symmetric encryption

The second part of encryption of preset messages involves using AES and the symmetric key generated by part 1. Only the payload of messages of type 1 are encrypted using this, as Microbits cannot know who the message is meant for and which key to use to decrypt if everything is encrypted. Broadcast messages are the only type 1 message which shouldn't be encrypted at all. The general function for AES encryption is used for both encryption and decryption.

Three AES functions are provided in 'nrf_ecb.c'; nrf_ecb_init(), nrf_ecb_crypt() and nrf_ecb_set_key(). A 16 byte key buffer is passed to the set_key function,

with the last byte of the buffer set to the symmetric key associated with the destination Microbit. A separate 16 byte counter buffer is passed to the crypt function, and the result is XOR'd with the first 16 bytes of the string to be encrypted. Every 16 bytes, the last byte of the counter buffer is incremented, and the crypt function is called again on this new counter buffer, and the XOR is done with this new counter buffer. For this to work properly, both the sending and receiving Microbits must start with the same counter values, and also must set the same bytes in both the key and counter buffers. Our supergroup convention is to start the counter from 0, and to always set the last byte.

2.3.3 Authentication

As well as encrypting the messages between Microbits, we were asked to also ensure that the messages sent were authenticated, so than no individual could impersonate someone else. After two Microbits enter each others range and send out key exchange messages (type 4 in the protocol), then the messages exchanged from then on between those two Microbits are authenticated to have come from that Microbit. This is because the Diffie Hellman exchange means that only those two Microbits will have the same key for the symmetric encryption. Any other Microbit attempting to impersonate one of them will have the wrong key and wont be able to decrypt any of the messages.

Before the key exchange happens however, there is no authentication. After some discussion at supergroup meetings we decided it is not necessary. Anyone can use any Microbit, and our protocol dictates that messages are addressed to other Microbits using their serial ID number, not a name. Since there is no central registry where we have assigned Microbit IDs to an individual, then there is no link between individuals and Microbits. It is possible for someone to spoof another Microbits serial ID in the packet header of a key exchange message, and therefore receive all the traffic meant for that Microbit. However, this violation of trust can only be solved by having a central certification authority, or by sharing a file with everyone in the supergroup.

2.4 Serial communication

Lead author Chance Carey
Edited by Robin Legg.

2.4.1 Microbit proxy mode

One of the focuses we had when developing support for allowing Microbits to effectively wirelessly access the server was that the process be as transparent and straightforward as possible. One requirement as a result of this was that any and all Microbits should run exactly the same software; thus, the process of a Microbit deciding to run as a proxy-node Microbit would have to be performed at runtime instead of, for example, different firmware having to be flashed to the device, or a software flag being used. The approach we chose allows any

Microbit to dynamically turn into a proxy-mode device, and is an operation that can be performed at any point while the Microbit is already running.

Our solution to this was to, when the Microbit is connected and the server starts up, send serial packets indicating that the Microbit had a server attached to it. Every Microbit would constantly be listening over serial, and on receiving one of these packets, would transition to operating in proxy mode. The differences in operation between running in normal mode and proxy mode are fairly significant. In proxy mode, the Microbit functions as a dedicated proxy device instead of allowing the user operations (navigating through menus, using applications, sending messages, and so on) to be used as normal. This decision was chosen as it vastly reduced the complexity of switching the behavior of the Microbit. As discussed previously, when in proxy mode, a Microbit forwards all radio packets intended for it to the server using serial protocol, and then forwards any response from the server back to the initial sending Microbit, using the ID of that Microbit that is stored on receipt of the initial message. A further examination of the protocol both in how it operates over radio and serial is given below. Overall, we are extremely satisfied with the approach and implementation of this aspect of the project, as it works reliably and is nearly entirely transparent when writing code to contact the server, which matches our requirements were exactly. In the future, this approach allows very easy expansion of other functionality of the Microbit and server, as wireless communication is seamless and, once implemented, needs no further maintenance or extension as it is purely a transport layer concern.

2.4.2 Protocol

The radio protocol for using the proxy device as a transport between a Microbit and server that are not connected has been discussed in previous sections. A brief overview, however, is that any Microbit may construct a packet and send it as if it is sending a serial packet to a server that is connected. Truly, the only thing that changed was that instead of calling a method to send via serial, another method is called that automatically frames the packet and sends it via radio. When the proxy Microbit device receives a packet via radio, it stores the sender ID (which is part of the packet), and sends the payload to the server via serial. On receiving data via serial, it is sent to the initial sender Microbit via the radio.

Almost every serial packet is handled like this. There are, however, two exceptions that are hard coded packets which the proxy device will not attempt to send over radio. These packets are purposefully for the proxy device itself. The first of these is the hello packet, sent from the server to indicate that the recipient Microbit is connected directly to the server and should transit to the proxy mode of operation. The sole other exception is that of the configuration packets, for multiple reasons; these will now be discussed.

The first packet types of the serial protocol to be discussed are that of the configuration packets. Unlike most server to Microbit interactions, the process of configuring a Microbit begins at the server side - it sends a request configu-

ration packet, which tells the Microbit that it should respond with its current configuration. As this process is server-initialized, the proxy Microbit doesnt know which wireless Microbit to send the packet to. It is further rationalized by the fact that when configuring a Microbit, it is likely that the Microbit is in the immediate users possession and can thus be plugged directly into the server; doing configuration wirelessly would result in a significant increase in complexity.

Similar to the supergroup radio protocol, serial packets begin with a single byte denoting the type of the packet. Each variable length field is prefixed with a single byte field denoting the length of the variable length field. This eliminates the need for using null terminator characters to denote the end of variable length fields, which would be inappropriate for a binary protocol such as this as the server may, for example, send binary data that would include a null byte which would erroneously be detected as a null terminator character.

The remaining packets - those aside from the hello and configuration packets - are exclusively request/response packets, in that the transmitting Microbit requests a resource (such as a series of tweets), and the server then sends a close to immediate reply containing the required information. This is true for the packets relating to requesting tweets from Twitter, which can be requested by either username or hashtag, and for which the response is a variable length array of variable length strings - multiple tweets are sent at the same time, which the user can then navigate through. Weather requests can also be made for specific locations, and sports requests operate similarly, although due to free API limitations, only a limited selection of sports results are available.

Overall, we consider the range of packets useful and sufficient for implementing the functionality that we intended to implement, and that the protocol(s) are sufficient for such tasks.

2.5 Microbit applications

2.5.1 Twitter

Lead author Robin Legg

The Microbit has access to twitter data for up to 5 usernames and 5 hashtags.

API and Design The initial plan for connecting the device to twitter data was through the use of the official Twitter API. However, after a large endeavour, we discovered that the API was too difficult to implement as it was overly complex. Instead we resorted to using *twitter_scraper* python package which allowed us to retrieve an array of tweets filtered by either username or hashtag all with a single command. A subsection of this array is then sent over to the Microbit which is displayed as a list for the user to browse through.

Preset Hashtags and Usernames The preset Hashtags and Usernames are selected and added via the web app. There can be up to 5 of each of these. One

of these is then selected by the user and requested from the server.

2.5.2 Weather

Lead author Robin Legg
The Microbit has access to current weather information for up to 5 locations anywhere in the world.

API The weather information is provided by OpenWeather API via the server. We have used this API as it is free, reliable and has a vast range of locations. The API also returns the data in json format which, in python, is easy to decode and retrieve the information we need.

Preset locations The preset locations are configured via the web app. There can be up to 5 of these and each is checked to be valid before being added to the microbit. One of these can then be selected and information on its weather is given to the user.

Weather Information We decided to show the user just the temperature and a brief description of the weather in order to keep the information brief and to the point. Especially as reading large amounts of text can be difficult on the Microbit's small screen.

2.5.3 Sports scores

Lead author Robin Legg
The Microbit has access to football scores from the last week for either the French Ligue 2 or the English Football League Championship.

API The football scores are provided by APIfootball. This API was chosen as it had a free package available that gave us access to data of two leagues for proof of concept for this functionality. It also returns JSON data that as previously stated is easy to deal with.

Preset League The choice of which league the user gets data from is configured via the web app.

Football Information We decided, once again, to keep the data displayed on the Microbit as simplified as possible. With this in mind, the Microbit only displays the basic the data in the basic 'Home-team Score Away-team' format. This allows the user to gauge the key information without any additional distraction of extra statistics like team sheets.

2.5.4 Piano

Lead author Houston Owen, Gopal Juneja

2.5.5 Play Piano

Lead author Gopal Juneja, Houston Owen

As soon as this sub feature of the piano is selected by the user, it allows the user to play the keys on the piano equipment to produce a sound with respect to the note of the key. To exit this mode, the user has to press button A on the Microbit. This feature is implemented to show that not only can there be sounds played by selecting one of the preset tunes on the Microbit, but also the user can interact with the Piano equipment in the same way one would interact with an actual Piano. This feature is also useful to gain knowledge about the notes of the piano, if one doesn't know them already. The range of notes on the keyboard is the 4th octave of a piano. The code provided on studres (which had been adapted from [1]) was inefficient, causing the piano to play notes after a long delay or sometimes not even detecting a note being played at all. A quick restructure allowed the Play Piano mode to be much more responsive.

2.5.6 Play RepeatME

Lead author Gopal Juneja
In an effort to do something unconventional with the piano equipment, a simple but enjoyable game was implemented for the user to play. The inspiration to make this game comes from the popular game known as "Simon Says", where one player takes the role of "Simon" and issues instructions to the other players, which should only be followed if prefaced with the phrase "Simon says". The RepeatMe is a more elementary version of this; where instructions are given by the piano equipment, and the game can be played by only one player at a time. As soon as the user selects this option, the piano equipment starts playing 5 random notes. After the random notes are played, the Microbit would display a message which indicates user to repeat the tune that was played by the piano equipment. If the user is able to successfully produce the same tune which was played by the piano, the user wins. The Microbit displays a message stating the same. If the set of notes entered by the user is wrong, then the user loses, and similar to the win scenario, the Microbit displays a message stating that the loss.

There are a some key points about the game, that need to be kept in mind when playing the game:

- The game can only be played using the black/sharp keys on the piano equipment. This is done in order to make the game of ideal difficulty

[1] https://raw.githubusercontent.com/KitronikLtd/micropython-microbit-kitronik-klef-piano/master/klef-piano.py

level, with the idea that the game is not too hard but not too easy as well.

- The user needs to enter at least 5 notes, whether those notes are right or wrong, in order to reach an outcome.
- The note that is being played on the piano is displayed on the Microbit. This is done to increase the usability scope of this feature as there might be users who have never played a piano and it might be impossible to recognise a note solely on the basis of audio sense.
- Similarly, when it's the user's turn to repeat the piano's composition, each note played by the user is displayed on the Microbit.

Logic By iterating 5 times through a loop, the piano is made to play 5 random notes on the piano. The random notes being played by the piano are stored in a vector data structure. Similarly, the notes played by the user are inputted using a loop, and the notes are then stored in another vector data structure. The two vectors are compared and hence the outcome of the game is determined, and the result displayed to the user.

Issues encountered Co-author Jet Chew

We first encountered our problem with the piano when we were developing our Random Notes game. We discovered that printing specific text in a specific order would cause the microbit to crash. Only later would we realise that scrollAsync was interrupting the microbit's fibers, which was the root of the crash issue. This only occured when radio was receiving in the background, since it is done in a while loop using a fiber. To circumvent this, we had to temporarily disable receiving on the radio until the game was over.

Another issue we encountered was the keyboard's hypersensitivity. Touching the power cable connected to the keyboard would often trigger the C key. While unavoidable, we were happy that it helped improve the Random Notes game's difficulty.

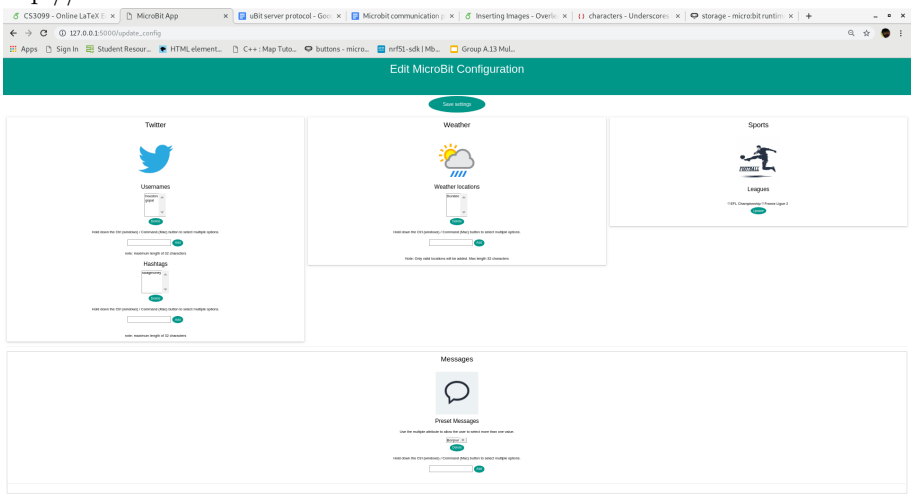
2.6 Configuring the Microbit

Lead author Jet Chew
Edited by Houston Owen

2.6.1 Server web interface

The user is be able to select the content that they wish to consume, e.g. Twitter usernames and hashtags they wish to follow, weather locations, and the sports scores. Additionally, the user can add their own preset messages which they will be able to send between Microbits while disconnected from the server. We wanted to have a convenient way for the user to be able to achieve this functionality, so at first we considered configuring it directly on the Microbit. After initial trials, we concluded that typing on the Microbit is clumsy at best. Hence,

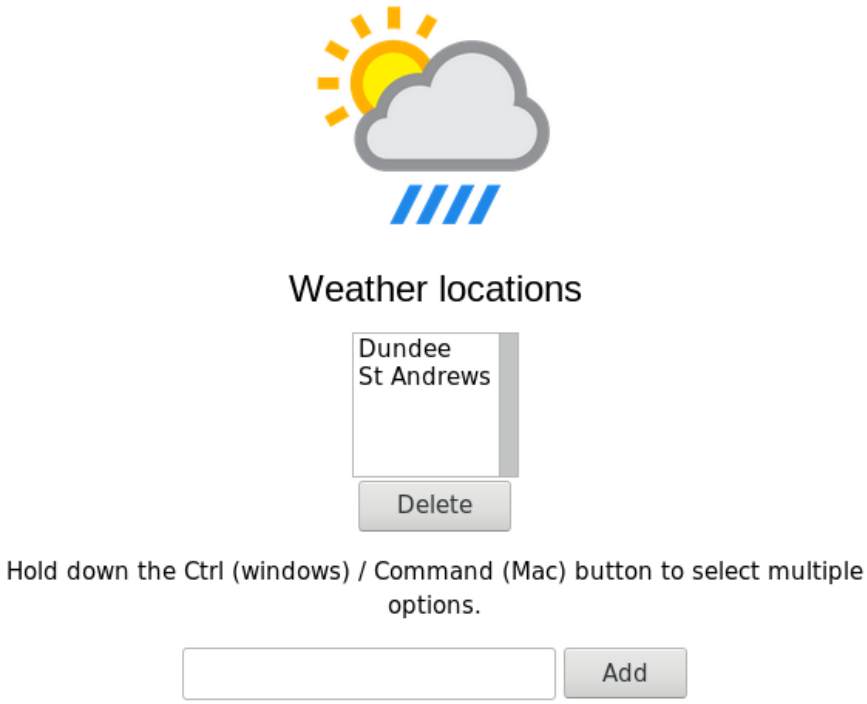
our solution to this problem is to allow the user to configure the settings directly from a web browser. The website for configuring the microbit is located at http://localhost:5000.



Design The idea is that all configuration should be stored in a JSON file, due to its cross-compatibility across different platforms and its lightweight nature. To manipulate this JSON file from a website, we have decided to use Python for the webserver. This allows for easy integration into our preexisting Python server code base.

MConfig Object A Python class was created to mirror the content that would be stored in the JSON file. We have 4 lists to represent the types of social media content we can obtain (Twitter usernames, hashtags, weather locations, sports scores). modify_config.py contains functions to manipulate the JSON file and the lists. It also validates input, i.e. values entered through the web interface needs to be valid, else they will be rejected.

Flask Web Server Flask was selected as the server architecture due to its setup ease and high functionality. What makes Flask special and easy to use is the use of routing, where functions can be binded to specific URLs and called that way. In addition, the template for the website is coded in HTML and allows for the injection of Python code. By exploiting this feature in combination with HTML select lists, we can add and delete multiple items on a displayed list, which is immediately updated and rendered on the webpage. The addition/deletion function is done through HTML forms, and the action executed calls the URL corresponding to the list. Then, the appropriate Python function is called via routing, which calls functions from the MConfig class to update the JSON file. The items to be added/deleted are obtained through Flask's request variable.



Web server + Communication server Since the web server is done in Python, we can share the MConfig object directly with the microbit's communication server. This way, the MConfig object is always up to date between both servers. To implement this, we run the communication server in parallel the webserver code by executing the communcation server code on a separate thread using Python's Thread from threading. We can now run both servers at the same time by calling *flask run*.

2.6.2 Configuration persistence

Once the configuration is loaded onto the microbit, we want the information to persist through power cycles. Otherwise, the device needs to fetch the configuration from the server upon each boot. The only time that the Microbits flash memory will be overwritten is when new code is flashed to the Microbit, in which case the menus will be populated with some default values.

Implementation We will be using microbit runtime's storage implementation by reading and writing to a set of KeyValuePair objects. Each entry in the configuration file will be assigned a key and stored as a value. However, each value can only hold a maximum size of 32 bytes, and due to the microbit's limited storage capacity, there can only be 21 pairs. To overcome this limitation, we have decided to allow a total of 20 entries in the configuration file, translating

to 5 of each option. The first 2 bytes of each value would translate to the size of the entry, and the next 30 bytes would contain the value. 2 bytes was used to represent each char in the size part, e.g. "st andrews" would be stored as "10st andrews". To further reinforce this 30-byte entry limit, we have assigned a limit to how long entries added through the web app can be, so that the Microbit won't attempt to save values longer than the maximum size allocated, preventing future errors.

To keep track of where each configuration is stored, we assigned a specific key to each entry. The key for each entry is a ManagedString of the index, and the entry is determined by the index table. Value entries are of the form of uint8_t arrays. The table below will denote the assignment

Index	**Entry**
0-4	Twitter usernames
5-9	Twitter hashtags
10-14	Weather locations
15-19	Sports teams

Issues encountered The biggest issue encountered is the limitation of having to access the storage through the KeyValuePair interface. Ideally, we would be able to store an entire Config object in memory, removing the need for breaking it down into bits every single time we wish to access/write to it. Plus, we won't be limited by just the 21 entries.

Other issues related to the Microbits limited memory include the tendency to drop radio datagram packets which were too large - as a supergroup, we decided that datagram packets of size 255 bytes or under was appropriate.

2.7 Web page design

Lead author Robin Legg

The basic web page template was based off a W3-schools template which is credited within the code. Further styling choices such as the buttons and colours also used W3-Schools CSS files. We decided to use these CSS files as they were convenient, reliable and looked good. This enabled us to create a professional looking environment quickly and effectively, by cherry-picking the features with the nicest aesthetic. The main aim in choosing this kind of simple design was to give the user a straight forward overview of the configuration of the Microbit, and to allow the user to easily make changes and update the settings.

2.8 Additional features

Lead author Gopal Juneja

Apart from implementing functionality that the basic specification and the given extensions of the piano and the encryption required from the group, the group decided to add a couple more small features. This was done keeping the

main objective of this project in mind; make the Microbit a device, which can serve as a viable alternative to a smartphone.

2.8.1 Secure lock functionality

Lead author Gopal Juneja

As a part of the extension released earlier this semester, the secure lock functionality has been implemented to prevent execution of any of the other features that the user can interact with. As soon as both the buttons present on either side of the 5x5 display of the Microbit are pressed simultaneously, the microbit enters the locked mode. An image of a lock is displayed, which indicates that the Microbit is now locked. To unlock the Microbit, the user needs to press a very simple combination of the forenamed buttons; button B has to be pressed for one second, at first, followed by a rapid pressing of button A. If the said task is performed in the correct time and order, the power on welcome animation becomes active, and the user is redirected to the main menu. This feature serves the purpose of making the Microbit more like a device which gives a more personalized experience to the user, in a similar way a smartphone or a personal computer would do. Note that while locked, the Microbit will not display any messages it receives.

2.8.2 Power on welcome animation

Lead author Jet Chew
When the user powers on the microbit, they are welcomed with a wink. We believe that this adds a personality factor to the device. The personality factor changes the frame at which the user perceives the device - it is no longer just a gadget. It becomes the users' companion. This inclusion, although very simple, gives the impression that the device is easy, fun, friendly to use, plus highly interactive. Our aim is for the user to not just use the device, but also like it.

2.8.3 Preset Piano Songs

Lead author Houston Owen
Although the speaker on the Piano attachment can only play a single frequency at a time and therefore cannot play anything of substantial complexity, music was still able to be added to the Piano app. Many people enjoy using their phones to listen to music, and as an additional offline capability this could potentially be an attractive feature to individuals using the Microbit as a minimal social media device.

3 Inter and Intra Group Work

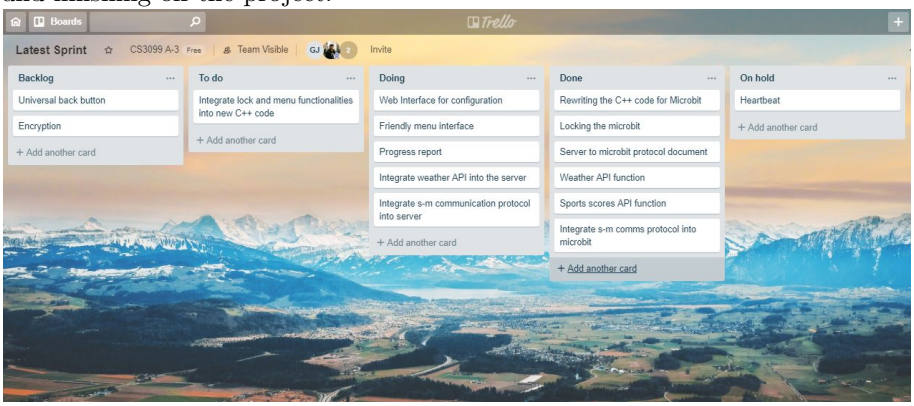
3.1 Scrum review

Lead author Gopal Juneja

Edited by Robin Legg.

In contrast with the plan discussed in the first hand-in, the team has made a reasonable effort in following the agile development methodology and adhered to the plan as closely as was possible. The various attributes that were laid out at the start of the project were not changed at any point, and effective efforts were made to adhere to those attributes. The group has used bi-weekly sprint cycles to achieve a reasonable number of tasks, meeting with our supervisor in the middle of the sprint for any required guidance on the tasks being performed and on any realistic plans we laid out for the future. Trello Boards have been used effectively, as planned, in order to keep track of progress as well as to provide a reference for tasks to complete during the current sprint. Continuous remote communication within the group existed through our private channel on Slack, and within the supergroup via a separate supergroup channel, established on said application.

The tasks were compartmentalized into multiple phases depending upon any upcoming submissions for this project at the time. So, the first set of meetings were mainly based around the first hand-in, followed by the group meeting for the remainder of the semester to discuss the submission plans for the minimum viable product. A similar approach was taken in the second semester, with the first set being based around the mid-semester report and the latter being based around finishing off the project.



With respect to the execution of SCRUM, the group had a lucrative start in the first semester with meetings taking place in a timely fashion, and the continuous progress in the development of features on the Microbit. There were hinderances in the middle of the semester, when the stand-up meetings didnt occur at a consistent frequency every week, due to some members having personal commitments and deadlines for other modules coming thick and fast.

However, during this period, the team made good use of the slack channel, keeping in regular contact.

Compared to first semester, the group had a somewhat slower start in the second semester, as hinted in the last hand-in, since group meetings, and meetings with the supervisor had to be rescheduled on a few occasions, due to time constraints caused by either personal commitments, or by work due from other academic modules. However, as we got closer to the deadline, the group gradually progressed back towards meeting when planned and increase the pace of completing the work.

The group has faced hindrances when it came to in-person communication as mentioned above, however those hindrances were used as an opportunity to propel the efficiency with which the group communicated remotely.

In the end, all the members have worked in sync with each other with pair programming playing an important role in the completion of the majority of the functionality of the Microbit, not only for the official submissions, but also for the sprint deadlines set within the group.

3.2 Supergroup Interaction

Lead author Houston Owen

As well as the SCRUM meetings we had within our group, there were supergroup meetings almost every week involving a representative from each group. The focus of these meetings was initially around designing a protocol for inter group communication on the Microbits (mostly in the first semester). At the start of the second semester encryption of messages between Microbit in the same supergroup was added as a compulsory element of the spec. There were additional supergroup meetings focusing on what kind of encryption to use as well as how to authenticate individuals using the Microbits. Both encryption and inter Microbit communication protocol have been described in detail earlier in this report.

4 User Testing

Lead author Robin Legg

4.1 Introduction

In order to see how successful our project has been we decided to construct a small user test. To do this we gave our Microbit to some users and constructed a feedback form to gather some results.

4.2 Creating the feedback form

The creation of the feedback form was crucial to quality of the results that we could gather. We tried to keep the questions focused on two main areas. Firstly, how easy the system is to use. This area was studied as we felt that if the system was difficult to use then it would discourage the user from using it and thus we could never hope to achieve a reduction in the users social media and smartphone usage. The second main area we looked at was whether the product could reduce the users social media smartphone consumption. Unfortunately, we did not have enough time to run a full study with empirical evidence as to whether the product does reduce usage so we settled for the users opinion on whether they believed it could. The final section is then for general feedback and whether there was anything the user thought could be improved. An uncompleted feedback report is attached in the appendices.

4.3 User Test

During the feedback tests, we gave each user a walk through of the system, highlighting the key features and how to use the various aspects. We then gave them a short time to use it on their own and form their opinions.

4.4 Results

Below is a discussion of the results from the user study. The feedback forms are all available in the appendices.

4.4.1 Usability

The first two questions on our feedback form were based around usability of the system.

Microbit The average score out of 10 (with 10 being super easy) was 8. This shows that users find our system accessible and weren't confused by any of the sections. The menu's were also clearly found to be intuitive. Although, some feedback was given about the scrolling text not being ideal and the possibility

of using icons instead. The responsiveness of the buttons was also commented upon but we feel that this is more of a hardware issue.

Web App The average score out of 10 (with 10 being super easy) was 9. This shows again that users found our system easy to pick up and understand. The simplicity of the page was commented on, both as helping the page to be easily understood but also as a negative when it came to giving feedback to what the user was doing. Dialogue boxes and animated buttons were some of the additions that users said may have been useful for future implementations.

4.4.2 Features

Weather Every user found weather integration useful and stated that this was something they did on their phones currently. This means that the implementation of this feature is helping towards reducing a users need to use a smartphone and thus reducing the possible usage of social media.

Sports Scores Two thirds of users found the sports scores useful and something that they did on their smartphones via social media. The other third stated that the only reason they didn't find it useful was because they were not really interested in the data and so never previously used their phones to check scores. The positive feedback means that implementing this feature was useful in reducing social media usage. However, there was feedback as to the limitation of this effect with the current limited selection of leagues, although, this doesn't bother our test too much as the only reason that other leagues aren't implemented is cost.

Twitter All users found twitter integration useful with them all again saying that they would use their mobile's for this previously. This is a big plus for our testing as if users find the implementation of this useful then it is likely that they will use the device to check this social media rather than their phones.

4.4.3 Further feedback

Most of the additional feedback received was to do with expanding the current selection of apps available on the device. Some commented that Facebook or Instagram notifications would have been useful. These would be great next steps for the project if we had more time and would definitely help us to reach our aim of reducing screen time on social media.

4.4.4 Reducing Social Media usage

Every single user said that this device could reduce their social media usage. This is a good result for us as it shows that, as we had previously suspected, that we have made a great attempt at achieving our goal of reducing social media usage.

4.5 Evaluation of Using Testing

The test results provide a lot of positive feedback and evidence that our project has achieved its goal. However, the test results aren't perfect and are by no means definitive proof of the achievement. First of all, our study sample was quite small with just three users. For further validation and better feedback we would have liked to expand this. Also, this test doesn't provide empirical evidence. In order to get this, we would want to monitor some users social media usage with and without a Microbit and then use statistical techniques to show the validity of these results. However, this study is a good step towards proving that our product really does do what we set out to do.

5 Evaluation and critical appraisal

Lead author Jet Chew

5.1 As a usable smartphone replacement for social media services

Lead author Jet Chew
Originally, we discussed creating a device with the intention of it being a usable replacement for many of the social media services that smartphones offer, while also reducing the time spent engaging in those services. This idea came up as research shows that individuals find it difficult to overcome this addiction due to the nature of how smartphones are designed: they are built to be addicting [2]. We hace succesfully implemented this functionality, users can access social media apps like Twitter with content tailored towards their preferences.

5.2 Creative use of input and output techniques

Lead author Jet Chew
The hardware we will be using to facilitate our goals of reducing time spent on smartphones looking at social media is limited in nature, with only a 5x5 screen and two buttons. This calls for creative use of input and output techniques when using the apps. During initial trials we tested out typing on the microbit, but have decided to phase it out due to its clumsiness. Instead, we send saved presets as input, which is quicker and easier to do. Output wise, the default output technique of the microbit is quite slow, and performs synchronously. We have decided to modify the way we output data, by calling the asynchronous scoll function, at a comfortable reading speed. This way, no time is wasted slowly scrolling through a large string of text.

5.3 Consistent and well defined way of navigating through the device

Lead author W
e aimed to have our micro:bit be feature rich while remaining straightforward to use. This is because we know that people who wish to curb their social media addiction would not mind a bit of a learning curve, but we would not want them to have to tolerate a device that is too inconvenient for effective use, as this may steer them away from using the device at all. We consider that one of the best ways to accomplish this is through having a consistent and well defined way of navigating through the device, with for example the same key combinations

[2] Haubursin, C. (2018). Its not you. Phones are designed to be addicting.. [online] Vox. Available at: https://www.vox.com/2018/2/27/17053758/phone-addictive-design-google-apple [Accessed 10 Oct. 2018].

used to confirm or cancel an operation, navigate backwards, or return to the main menu.

5.4 Additional features

Lead author Jet Chew
We previously discussed having the device include novel features outside of the basic functionalities. The features implemented includes a game we can play with the piano, and a selection of songs the microbit can perform. Plus, there is an additional security feature implemented - the microbit can be locked, which unlocks with a correct combination.

6 Conclusions

Lead author Gopal Juneja
Overall, the group has managed to complete the objective of this practical, by adding all the functionality on the Microbit, that is a compulsory part of this project, which entails a multiple implementations. This has been evidenced in the user study. Various features like Twitter integration, weather integration, developing a web app, establishing communication between Microbits, and establishing communication between Microbit and the server have been implemented. Furthermore, The extensions mentioned in the second semester, which included making the Microbit functional with the piano equipment, and enabling encryption have also been implemented successfully. It is safe to say that the implementation of all the features on the Microbit, some of them forenamed above, have made the Microbit the required minimal device, which would encourage users for staying in contact with the world while staying in contact with their surroundings.

This project gave the group members an experience of working in a team and solving various types of problems together, whether those problems concern the project itself, or whether they concern a certain group member. Learning about concepts like SCRUM, pair programming, and many other logical topics included in the development of the Microbit has engrossed the group members knowledge and also helped the team gain hands-on experience on what its like to work on a project quite similar to the work performed in the professional world.

7 Appendices

Lead author Houston Owen

7.1 Software Modules Used

There were two main software modules used in this project from outside sources; the AES symmetric encryption functions, provided in the nrf_ecb_.c file, and the functions to initialize the Piano and get the key press. The Piano code was adapted from a file posted on studres, which in turn was a C++ translation of the original file, written in python. The source file for the python can be found here: https://raw.githubusercontent.com/KitronikLtd/micropython-

microbit-kitronik-klef-piano/master/klef-piano.py The nrf file was taken from

the nrf-51sdk on Mbed docs, found at: https://os.mbed.com/users/rengro01/code/nrf51-

sdk/docs/tip/nrf__ecb_8c_source.html Other resources used in this project was

CSS styling pages from W3Schools and our web app also uses the base HTML from a W3Schools template.

7.2 Steps to run the server

Lead author Jet Chew

- Navigate to the server directory, i.e. cd server
- Create a virtual enviroment, i.e. virtualenv -p /usr/bin/python3.7 venv
- Run the virtual environment, i.e. source venv/bin/activate
- Install pipenv, i.e. pip3 install pipenv
- Add the directory where pipenv is installed to the PATH, i.e. export PATH=$PATH:path/to/bin
- Install the dependencies in the pipfile, i.e. pipenv install
- Navigate to the flaskapp directory in server, i.e. cd flaskapp
- Run the server, i.e. flask run

7.3 User Test Feedback forms

Lead author

Microbit User Feedback

1. How easy did you find the microbit to use? (1-10, 10 being super easy)
2. How easy did you find the web server to use? (1-10)
3. Is being able to access the weather useful and something you would have previously used your smartphone or social media for?
4. Is being able to get football scores useful and something you would have previously used your smartphone or social media for?
5. Is being able to get twitter data useful and something you would have previously used your smartphone or social media for?
6. Anything you wished was different?
7. Could this reduce your social media phone usage?
8. If not, why not?

Any additional feedback:

Microbit User Feedback

1. How easy did you find the microbit to use? (1-10, 10 being super easy)

   9 - menus were easy, buttons were a little unresponsive

2. How easy did you find the web server to use? (1-10)

   10 - very simple layout, easy to work with and understand

3. Is being able to access the weather useful and something you would have previously used your smartphone or social media for?

   Yes for both

4. Is being able to get football scores useful and something you would have previously used your smartphone or social media for?

   No - I don't like football and so I don't use my phone or social media for score information

5. Is being able to get twitter data useful and something you would have previously used your smartphone or social media for?

   Yes to both

6. Anything you wished was different?

   Addition of Instagram notifications

7. Could this reduce your social media phone usage?

   Yes, I could use this device instead of my phone to check for tweets from my favourite people and for weather.9

8. If not, why not?

Any additional feedback:

I like the piano feature and displaying the notes of well known songs is nice touch

Microbit User Feedback

1. How easy did you find the microbit to use? (1-10, 10 being super easy)

   7 - Reading text can be tiresome and buttons sometimes slow to respond

2. How easy did you find the web server to use? (1-10)

   8 - Layout is pretty and simple to pick up, however some better feedback when pressing buttons would be great

3. Is being able to access the weather useful and something you would have previously used your smartphone or social media for?

   It is useful and I would have previously used my phone

4. Is being able to get football scores useful and something you would have previously used your smartphone or social media for?

   100%, I often use my phone and twitter to check the scores

5. Is being able to get twitter data useful and something you would have previously used your smartphone or social media for?

   As just mentioned I use twitter to get sports scores and to keep up with the news so twitter integration is useful

6. Anything you wished was different?

   Perhaps symbols instead of text on menus as scrolling text is tiresome to read
   Also more leagues in the scores part would be good as I generally follow Premier League teams.

7. Could this reduce your social media phone usage?

   Yes, I could use the device to check twitter and the football scores

8. If not, why not?

Any additional feedback:

The piano is a cool gimmick
Love despacito

Microbit User Feedback

1. How easy did you find the microbit to use? (1-10, 10 being super easy)

8 - Fairly easy interface to navigate, and was slow enough to read but also allowed you to skip ahead if you knew what you were doing.

2. How easy did you find the web server to use? (1-10)

9 - As easy as could be, was just point and click in terms of giving input. Only downside was no dialog boxes to tell you if you gave invalid input - it just never gets entered.

3. Is being able to access the weather useful and something you would have previously used your smartphone or social media for?

   Checking the weather is something I do on my phone.

4. Is being able to get football scores useful and something you would have previously used your smartphone or social media for?

   Yes, but the leagues available were severely limited.

5. Is being able to get twitter data useful and something you would have previously used your smartphone or social media for?

   It was cool but I don't use twitter very much

6. Anything you wished was different?

   More options for social media apps, such as Facebook. Also would have been nice to get the time.

7. Could this reduce your social media phone usage?

   I could see it reducing it slightly, although the apps i use most were not covered so wouldn't reduce my time that much.

8. If not, why not?

As stated above..

Any additional feedback:

More offline games would be nice

8 Testing summary

Lead author Chance Carey
Describing the steps taken to debug, test, verify or otherwise confirm the correctness of the various modules and their combination. We employed a continuous approach to testing the correctness of our code throughout the development of the project. As each feature usually had multiple people involved during the development of the feature at any one time, it would be common for one person to continuously code whilst the other person would focus on flashing the code to the Microbit.

During the development of a feature, we would connect the Microbit to the computer that we were developing on and use the serial output of the device to print debug messages. We employed this as a mode of developing as the Microbit's 5x5 screen was simply too small to be able to display any sort of useful debug message effectively. Using the serial connection, we would print the internal state of the Microbit (such as various variables that we would like to debug) and be able to find any issues much faster than we would if we were attempting to debug using the Microbit's display.

To preserve code quality, for merge requests we had a requirement for the git repository that would require another maintainer to veirfy the code of any merge request prior to it being accepted. This person would test the code and ensure it worked effectively and as was expected, and if it did, would accept the merge request, and the code would be merged.

This dual approach of debugging as we wrote the code and testing by other members of the team when merging in feature branches allowed us to maintain the working nature of the master branch at all times, and ensure that there was always a successful build avaialble for the Microbit at any time.