Hugh Wells 2019

An investigation into attendance monitoring systems

Working Title

Wells, Hugh

A report submitted in part fulfillment of the degree of
BSc (Hons) in Computer Science

Supervised by:
Professor Hugh Shanahan

Department of Computer Science
Royal Holloway, University of London

Abstract

Royal Holloway keeps track of attendance in lectures both to ensure that students are regularly attending and also to satisfy legal requirements regarding the visas of overseas students (Home Office (UK Government) 2019). It is essential that this data is gathered and analysed efficiently and accurately.

In the past attendance has been tracked using signatures on registers. More recently, due to the lack of scalability of this former approach, a system of clickers has been employed (Royal Holloway Department of Computer Science 2018). This latter system has proved to be non-optimal and insecure.

The aim of this research is to investigate existing attendance monitoring solutions and existing academic research into the problem to determine a more optimal solution for Royal Holloway. An MVP (Minimum Viable Product) will be built to test assumptions and demonstrate the core ideas of the proposed solution. The system will need to be user friendly and satisfy Royal Holloway's requirements whilst also not becoming burdensome on lecturers, students and administrative staff.

Motivation

The primary motivation is to offer a more secure and convenient system for students. The author is well acquainted with both the paper based and clicker systems in use, and the associated merits and pitfalls. An improvement in the process not only allows less detractions from the lecture content but, as Universities UK (2019) writes: "Attendance monitoring can also be perceived as unfair and harm international student experience." - a more low key and low effort system reduces the potential for discrimination as a result of the perception of international students imposing rules on domestic students, as a result of their presence.

Universities UK (2019) continues, "The current system imposes a significant administrative burden on both institutions and the Home Office..." with a survey conducted by Universities UK concluding the the total cost of compliance with Tier 4 rules being £40 million to the UK Higher Education sector (Universities UK 2019). A separate study by EY (2019) for the Russell Group noted "Attendance monitoring is particularly time consuming across such a large university with many different modes of study. Collating the data, analysing it and escalating cases for investigation/explanation has created an industry of work for very little tangible benefit given that HEI students are very low risk of visa abuse.".

It is therefore clear that a more streamlined, automated and secure system is a clear benefit to both the University and Higher Education sector alike, as well as to students.

First Term Progress

Work in the first time hasn't gone entirely as planned, however some interesting discoveries were made in the reports and I have made significant progress on my understanding of the overall Bluetooth standards and clarified how my proposed implementation will work in practice.

The hardware elements of the project have taken longer than expected, in both the clicker research and Bluetooth implementation. This has mainly been due to communication issues between the BLE/Nordic devices and the various microcontrollers. The delays here have been disappointing, especially in the BLE case where a simple move to a different version of the device fixed the issues. The work to get the devices to communicate has allowed considerable research into troubleshooting techniques for UART and I learned a lot about flow control, which I hadn't come across before.

The clicker project remains unfinished at the moment. I intend to try and eliminate the encoding issues with the Arduino by connecting directly to the NRF chip from a Raspberry Pi. This will also require working on a basic library for the NRF chip, based on the code already in use on the clicker basestation, by Mooney (2019). I hope to achieve this over Christmas to complete that report.

The BLE work has been delayed by the issues with

the clicker, and then extensive issues with the BLE device itself as described in the report. However, once working there has been good progress and work will now continue on developing a working Web Bluetooth PoC for integration with the overall system.

Work on the cryptography report was pushed back, although the PoC program was completed and is ready to be implemented. As the signing mechanism isn't the most crucial part of the project (as long as it works and uses an established and peer reviewed algorithm) the report here was likely to contain less information that the aforementioned research.

Overall design work on the system continues on track as the user stories and functional requirements have been defined. The full design with appropriate schemas, UML and mock user interfaces is to be developed for the draft report.

An additional report on browser fingerprinting was added as this was discovered to be a bigger part of the project than originally envisioned. This did take up additional time however was extremely useful research and yielded an interesting PoC program.

In summary, although there have been significant challenges in several areas, the identification of these risks early on allowed mitigation within the project by rescheduling other workstreams and running parts in parallel to allow for greater flexibility whilst troubleshooting. Background research has gone well and provided additional context when troubleshooting issues and also when designing the overall system, to ensure constraints are observed.

The reports collated document this work thus far.

Project Diary

October 21, 2019

Monday:

- Have spent the day debugging issues with serial communication
- Have written a small Python script to try sending data which isn't working
- Discovered that I'm not actually broadcasting the hardcoded messages I thought I was (or at least they aren't valid). I'm trying to determine why this is...

October 28, 2019

- Finished writing up report on clicker emulation for now
- Additionally, gave up on getting the final pieces of clicker emulation to work for the moment: I will come back to it over a period of time once I've had a chance to consult the internet and Nuno some more, to work out what is going wrong!
- Begun researching communication methods for my proposed solution and preemptively ordered a HM-10 Bluetooth 4 module for further investigation

November 4, 2019

- Wrote some more about clicker communication
- Got a Raspberry Pi working over Serial with Ethernet passthrough
- Determined that it's not possible to connect to a Pi over Serial and utilize UART on the GPIO. Although the Broadcom chip support this, it is not connected in a way that allows both these Serial connections at once unless you use the Compute Module (the professional version of the Pi)
- Going to look into other options here as I want to avoid using an Arduino as I would like the networking capabilities of the Pi and the additional processing capacity. It also allows me to work with more standard cryptography libraries.

November 4, 2019

- Switched from a Pi Zero to a Pi 2 and now connecting via SSH with ethernet passthrough
- UART is now working, testing to try and get communications working with the HM-10

November 5, 2019

- I have a serial connection open, but I am not receiving data back. The board does have UART flow control pins and although I have tried the hacks (hold the enable pin low), I am still not getting any data back. Enabling hardware flow control on the Pi is possible, but involved. I have been testing with minicom and all of the Google solutions are still not enabling it.

November 13, 2019

Yesterday:

- Worked on more issues with the UART. I think I have found a solution but it seems it requires setting a config parameter on the board itself, so I have had to purchase some additional equipment to do this.

Today:

- Researched and wrote a PoC for fingerprinting web browsers and devices
- Researched PKI and RSA vs ECDSA algorithms. Wrote a quick Adaptor class with tests to play around with a library for this. To be extended.

Current blocker is getting the Bluetooth working. Once that is done it should be fairly easy to send data end to end (he says...)

November 14, 2019

- CeDAS workshop on report writing with very useful context and helpful information about how to structure the report. Have begun laying out interim report.
- Meeting with supervisor. Discussed progress and where to concentrate work from now on. Discussed how reports are not final and feedback is really useful on work that isn't quite complete. Structure of interim report clarified.
- Started writing up investigations into browser fingerprinting. I shall probably provide an extra POC program here for illustrative purposes.

November 18, 2019

- Discovered an article that suggested this might well be a logic level issue on the RX pin (3.3v vs 5v) and that there are two different baud rates depending on software.
- Took notes on BLE specification stuff for a report on communication options
- Discussed project in general with Dave – the various reports and the structure the project should take, plus the specific Bluetooth issues. Dave has suggested that as per another article there could be an issue with AT commands missing a line feed/carriage return, causing the null byte issue I'm seeing

November 20, 2019

Yesterday:

- Chatted with supervisor about clicker report. Feedback about general tone and structure of report.
- Discussed general project outline and progress to date
- Feedback on current report proposals. Agreed to add an additional report on general project ideas and outline, as without it the context of the project is somewhat confusing for an outsider.
- Communication report to be written, cryptography report to be delayed in light on the additional outline report and browser fingerprinting work.

Today:

- Implemented changes suggested in clicker report: https://github.com/RHUL-CS-Projects/FullUnit_1920_HughWells/pull/5
- Working on outline report

November 28, 2019

Earlier this week:

- Finished writing up the Outline report covering the basic system design and user stories
- Meeting with Supervisor to discuss this report, feedback and changes

- Significant research and compilation of notes on BLE specifications

Yesterday:

- Progress on Communications report theory. MIFARE attack covered and Bluetooth section complete up to part way through security features

Today:

- Finally managed to get HM-10 devices working properly. This has been delayed by waiting for hardware to arrive in dribs and drabs. An HM-10 from a different supplier and manufacturer arrived and, with no changed settings, has "magically" started working.
- I can now send data from a phone, through the HM-10 to a serial console and back again. My script to test AT commands works and various notifications are logged to the serial port when device changes take place (eg. disconnection). This is huge progress and I shall now be able to write a driver for the HM-10.
- Started work on a Web Bluetooth implementation. Have discovered that browser support is *shocking*. I currently am running experimental versions of Chromium (with flags enabled) and bluez (flags enabled) and am still encountering errors. I shall need to look further into this but it may prove unfortunatly rather fatal if I cannot find some combination of hardware that has reasonable support. Will discuss with supervisor.

Continued Risks

There are continued risks in the project, and additional risks have been identified.

Clicker emulator encoding

- **Impact:** Medium

This risk currently exists and is being mitigated. It is not a crucial part of the project and although it serves as a useful demonstration and makes for an interesting report, lack of a fully working system is not critical to success of the project. The plan is to attempt a different approach, however this will be timeboxed to ensure excessive time is not wasted.

Web Bluetooth API

- **Impact:** High

The Web Bluetooth API is less mature and less well supported than originally hoped. On the one hand, this does make a more interesting and relevant project (as there are few applications that use the API) however support is limited both in browsers and in documentation. Currently, although the API works on a specific Chromium set-up there are driver issues on Linux. Workarounds are being investigated and anecdotal research suggests that Google Chrome on Windows 10 may provide a more stable development environment. This has not yet been tested on any smartphones - this is an area to investigate.

Should the API fail to work with any software combination, it will be necessary to seek another solution. The implementation would be possible using standard HTTP client/server architecture on a dedicated wireless network; however this is not ideal for various reasons. There may also be other communication mediums commonly supported on smartphones, for example NFC, that could be investigated should the need arise.

The current plan is to continue working towards a Web Bluetooth implementation, but should this not prove possible more mature technologies are available and there is valuable research in looking at the current usability of Web Bluetooth.

Directory Listing

The repository has the following structure:

- proof_of_concept_programs
  - clicker_basestation
  - clicker_emulator
  - crytographic_signing
  - device_fingerprinting
  - hm-10-investigations
- reports
  - clicker_report

- communication_report
- fingerprinting_report
- interim
- outline_report
- plan
- proposal

Reports are written in "Frankenstein" Markdown and LaTeX. They are compiled using Pandoc via the Bash generation script in each directory. Compilation is non-trivial without all of the packages installed and thus it is recommended to refer to the end of this document for the outputs. Source is provided for reference.

Each proof of concept program directory contains source code and where not covered in a report, a README for installation and execution.

Running Programs

It should be noted that most programs submitted require hardware which I am happy to provide to those assessing the programs should they require it.

Reports

Reports are included in the following order, at the end of this document:

- Project Plan (included for context)
- Clicker Report
- Browser Fingerprinting Report
- Communication Report
- Outline System Design

Acknowledgements

Thanks to Tom Pollard et al. (2016) for the front cover template which I have adapted, Marco Torchiano (2015) for the Pandoc table preamble and Cohen and Matos (2013) for the Final Year Project guide and suggested layouts.

Bibiography

Cohen, Dave, and Carlos Matos. 2013. "Third Year Projects – Rules and Guidelines." Royal Holloway, University of London.

EY. 2019. "Challenges and Costs of the UK Immigration System for Russell Group Universities." https://russellgroup.ac.uk/media/5750/challenges-and-costs-of-the-uk-immigration-system-for-russell-group-universities.pdf.

Home Office (UK Government). 2019. "Tier4 of the Points Based System: Guidance for Sponsors Document 2: Sponsorship Duties." https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/824003/Tier_4_Sponsor_Guidance_-_Doc_2_-_Sponsorship_Duties_2019-08_1.1.pdf.

Marco Torchiano. 2015. "How to Solve Longtable Is Not in 1-Column Mode Error?" *TeX - LaTeX Stack Exchange*. https://tex.stackexchange.com/questions/161431/how-to-solve-longtable-is-not-in-1-column-mode-error.

Mooney, Nick. 2019. "Nickmooney/Turning-Clicker." https://github.com/nickmooney/turning-clicker.

Royal Holloway Department of Computer Science. 2018. "DEPARTMENT OF COMPUTER SCIENCE Undergraduatestudent HANDBOOK." https://intranet.royalholloway.ac.uk/computerscience/documents/pdf/currentug/ug-student-handbook-2018-19.pdf.

Tom Pollard, Marvin Reimer, David San, Arco Mul, Matthew Gwynfryn Thomas, Jakub Nowosad, Dennis Weissmann, and W. Caleb McDaniel. 2016. "Template for Writing a PhD Thesis in Markdown." Zenodo. https://doi.org/10.5281/zenodo.58490.

Universities UK. 2019. "THE STUDENT VISA SYSTEM: PRINCIPLES TO REFORM." https://www.universitiesuk.ac.uk/policy-and-analysis/reports/Documents/2019/student-visa-system-principles-to-reform.pdf.

Hugh Wells 2019

An investigation into attendance monitoring systems

Working Title

Wells, Hugh

A report submitted in part fulfillment of the degree of
BSc (Hons) in Computer Science

Supervised by:
Professor Hugh Shanahan

Department of Computer Science
Royal Holloway, University of London

Abstract

Royal Holloway keeps track of attendance in lectures both to ensure that students are regularly attending and also to satisfy legal requirements regarding the visas of overseas students (Home Office (UK Government) 2019). It is essential that this data is gathered and analysed efficiently and accurately.

In the past attendance has been tracked using signatures on registers. More recently, due to the lack of scalability of this former approach, a system of clickers has been employed (Royal Holloway Department of Computer Science 2018). This latter system has proved to be non-optimal and insecure.

The aim of this research is to investigate existing attendance monitoring solutions and existing academic research into the problem to determine a more optimal solution for Royal Holloway. An MVP (Minimum Viable Product) will be built to test assumptions and demonstrate the core ideas of the proposed solution. The system will need to be user friendly and satisfy Royal Holloway's requirements whilst also not becoming burdensome on lecturers, students and administrative staff.

Motivation

The primary motivation is to offer a more secure and convenient system for students. The author is well acquainted with both the paper based and clicker systems in use, and the associated merits and pitfalls. An improvement in the process not only allows less detractions from the lecture content but, as Universities UK (2019) writes: "Attendance monitoring can also be perceived as unfair and harm international student experience." - a more low key and low effort system reduces the potential for discrimination as a result of the perception of international students imposing rules on domestic students, as a result of their presence.

Universities UK (2019) continues, "The current system imposes a significant administrative burden on both institutions and the Home Office..." with a survey conducted by Universities UK concluding the the total cost of compliance with Tier 4 rules being £40 million to the UK Higher Education sector (Universities UK 2019). A separate study by EY (2019) for the Russell Group noted "Attendance monitoring is particularly time consuming across such a large university with many different modes of study. Collating the data, analysing it and escalating cases for investigation/explanation has created an industry of work for very little tangible benefit given that HEI students are very low risk of visa abuse.".

It is therefore clear that a more streamlined, automated and secure system is a clear benefit to both the University and Higher Education sector alike, as well as to students.

Introduction

The signed registers, as in Figure 1, consist of a dossier of pages circulated throughout the class during a lecture. These pages are printed in advance by the departmental office based on the course registration list and subsequent timetables. The four columns are reserved for the printed student ID number, printed student name and their signature which is added during the lecture as proof of their attendance. The registers are collected by the lecturer at the end of a session, certified and returned to the office where they are analysed.

Several complications have arisen with this system. Some lectures are big enough that at least two registers are required for them to circulate throughout the session and avoid a rush to sign at the end of a lecture - this of course creates added complication as to where the register needs to be passed next. It then also requires at least double the effort in correlating and combining the signatures by the administrative staff and there is no guarantee that both sections will remain together in transit!

Of course, there are rather more obvious issues such as the environmental impact of reams of paper each day and forging of signatures; the latter is explicitly mentioned in the departmental handbook "You must not sign the attendance register for anyone else, or

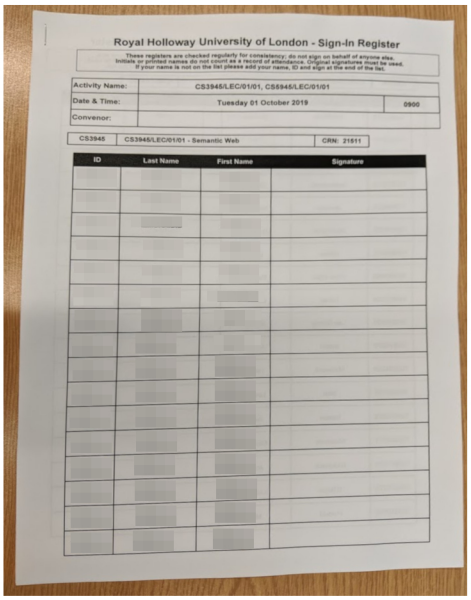


Figure 1: A redacted sign in sheet for a lecture

allow someone else to sign the register on your behalf. . . These behaviours are fraudulent and will result in disciplinary action being taken." (Royal Holloway Department of Computer Science 2018).

For the 2018 intake of first year undergraduates it was decided that due to the class size, paper registers were infeasible. The clicker systems was proposed and developed. This uses a Turning Technologies "Response Card" (Figure 2) typically used to respond to interactive questionnaires as part of a slideshow. The device communicates with a base station connected to a computer via USB (Universal Serial Bus) when a key option is pressed (eg. "1/A") and transmits the unique ID of the device and the key press. The message is acknowledged by the base station and the user is given visual affirmation on the device that their response was counted. The results are then stored in a semi-proprietary format attached to the slideshow which is decoded, processed and analysed by the department using a collection of scripts, Excel spreadsheets and custom software.

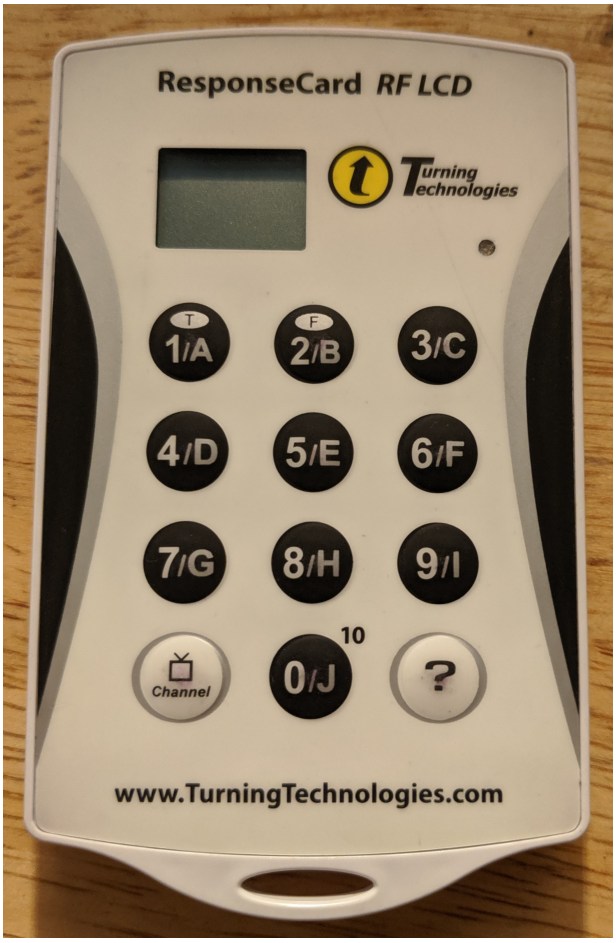


Figure 2: A Turning Technologies "Response Card"

The entirely custom decoding and data processing is due to the device being used outside its intended purpose. Turing Technologies ("LCD Clicker Response Devices," n.d.) advertise its use purely as a device[89] for polling students as they interact with a slideshow[90] during a class. The device level data appears to have[91] been intended to prevent duplication and to allow[92] scoring across multiple questions. Several problems[93] have been reported with processing the data saved[94] by the slideshow; the most interesting of which relates to the unique identifies for each device which are a hexadecimal string. For certain combinations (eg. "100000"), the spreadsheet program used for data processing parses this as an integer and will "helpfully" rectify it to the exponential form,

$$1E^6$$

This then requires manual correction for each row affected.

However, this research stems from a much larger flaw the author discovered when reading a blog (Goodspeed 2010) on a similar device by Turning Technologies. The blogger writes "It can be seen from the code that the 0x1A IRAM byte holds the channel number. That is, if 0x20 is stored at 0x1A, the radio will be configured to 2,432 MHz. The other configuration bytes reveal that the MAC addresses are 24 bits, the checksum is 16 bits, and the device broadcasts at maximum power sourced from a 16MHz crystal". Whilst this analysis might seem innocuous the crucial discovery made in that blog post is the entire register is sent directly to the radio chip - there is no encryption. As stated later on "... packets could be broadcast by a reprogrammed Clicker or NHBadge to make a student in virtual attendance..." which is a very dangerous for a device intended to be used in Computer Science classes!

Goodspeed (2010) concentrates on the earlier and less advanced "Response Card" which does not have an LCD screen, as opposed to the device in Figure 1. Their work was referenced and expanded in another blog by Killian (2012) for which the code (now no longer directly available) was modified and re-worked by Mooney (2019). This software emulates the base station connected to the computer and outputs the results:

```
Serial.print(F("incoming: "));
for (int i = 0; i < BUFSIZE; i++) {
    printf("%02x", incomingData[i]);
}
printf(" --> %c", incomingData[ADDR_WIDTH]);
Serial.println();
```

In order to produce a working proof of concept attack on the attendance system I will first verify that Mooney (2019) works with my personal clicker before developing software on another Arduino to work with the emulated base station to spoof multiple clickers on demand. I will then test my attack on an actual base station to prove the results of the sign in slideshow can still be processed by the department.

First Term Milestones

Please see Appendix 1 for a Gantt chart overview.

Reports

- **Week 3** A short history of attendance monitoring and the solutions that have been proposed and implemented for various practical purposes.
- **Week 3** A report detailing analysis of the existing clicker system compared to the register system exploring security and usability issues with both systems.
- **Week 5** A report looking at effective means for attendance monitoring covering existing solutions and technical methods to ensure trustworthiness and security. This will cover analysis of technologies such as smartcards, magstripe, on-device authentication etc. to present an overview of the landscape and potential cryptographic solutions.
- **Week 6** A report examining communication technologies between devices looking particularly at ease of use, ease of association (with a base station), security and flexibility (is custom hardware required). This will augment the previous weeks' report and provisionally look at the Bluetooth standards and Web Bluetooth API.

- **Week 12** A draft document utilizing findings from the aforementioned reports and proof of concept programs describing the proposed overall solution, the user stories it solves as well as a full design description of the proposed system including UML. This will explain the design choices for the functional parts of the system, the technology chosen and any specific software or hardware requirements required - with justification. It is not expected it will expand in great depth into the User Experience and visual design elements.

Programs

- **Week 3** Accompanying the report, a proof of concept program demonstrating how to manipulate the clicker system using cheap, off the shelf hardware.
- **Week 8** A proof of concept program using the selected communication technology to allow a device to connect to another device and transmit data from one webpage to the other.
- **Week 10** A proof of concept program of confirming attendance utilizing the previous communication proof of concept and the learnings from the report into cryptographic methods of verifying attendance. This will be a functional prototype, although it is not expected it will be very useable.

Second Term Milestones

Please see Appendix 1 for a Gantt chart overview.

Reports

- **Week 18** Expansion of the draft specification of the system. Minor clarifications to the core functionality will be made but greater emphasis will be placed on the User Interface design and proposed design sketches. This will include references to relevant Human Computer Interaction concepts to justify the design decisions and resulting User Experience for both student and staff users. It is anticipated that the data analytics functionality of the final system will be limited (it is not the main goal of the project) but the extent of this will be documented. The document will be complete enough for the system to be built independently from the author.

- **Week 23** A full draft of the complete project report encompassing all the previous reports, proof of concept programs and research (the interim project report).

Programs

- **Week 22** Substantially working system as laid out in the aforementioned documentation which is broadly complete and can be tested and demonstrated. Minor bugs are present, small features are incomplete and stretch goals are being planned.
- **Week 24** Complete and working system which has been tested against the specification. All required features have been implemented and some stretch goals have been achieved.

Key Risks

Device communication

- **Likelihood:** Possible
- **Impact:** Medium

Previously in the project proposal it was stated that the Web Bluetooth API would be used. I have decided to carry out additional research into the various communications technologies available to determine the most appropriate for this project. It is possible that there will be no good solution for direct device to device communication (this is intended to provide the “presence” factor of the authentication) in which case a more traditional challenge-response protocol with rotating codes displayed to students, may be appropriate. This reduces security but the ultimate aim is for this to be practical to implement. It is not envisioned this will be a “showstopper”, but it would require additional research and analysis to determine an appropriate way forward.

Analysis of clicker system

- **Likelihood:** Unlikely
- **Impact:** Medium

This relies on the research of previous (cited) authors and it is not feasible for the investigation they carried out to be completed for this project. The intention of this section is to demonstrate a live proof of concept using their discoveries applied to the specific Royal Holloway situation - this will require developing software and a test environment. It is possible that Turning Technologies have changed the way their devices work - in this case reasonable efforts can be made to reverse engineer the devices further and research more up to date analysis. At the point firmware is needing to be dumped the benefits of including this in the report start to outweigh the time needed to complete this section, and it detracts from the rest of the project. This section will produce useful learnings either way, but as with all penetration testing, the exact outcome cannot be predicted in advance.

Frontloading functional design

- **Likelihood:** Likely
- **Impact:**: Low

The milestones above call for the functional design to be broadly complete by the end of the first term, based on the research and proof of concept programs already developed. It is envisioned that much of the code developed during that term will be reworked and make it's way into the final release. This creates the danger that when integrating the system and building the wireframes out as a frontend panels that the architecture may need to change as the result of either changed requirements or oversights in the design process. The only mitigation to this is to ensure the code is well designed and documented, such that it can be modified as required to adapt to changing requirements. The design process should also be robust and based on the research, but it is not reasonable to expect no changes to be made further down the line.

Additional Risks

There are always additional risks relating to slipping deadlines, general time management, issues obtaining resources and problems relating to third party libraries, software and documentation. These are not discussed in detail as they occur frequently (eg. a proposed library does not have a version supporting a particular language) but as they are small and relate to parts of the implementation rather than the holistic project, the risk and impact is minimal.

Acknowledgements

Thanks to Tom Pollard et al. (2016) for the front cover template which I have adapted and "corentin" for the Gantt chart example in LaTeX("Pgfgantt - Gantt Chart Package," n.d.).

Bibiography

Due to the nature of this project, available references on the subject are limited and will be confined to more general information security concepts as well as attendance monitoring. In this report the available references are provided by those who have investigated these devices beforehand.

EY. 2019. "Challenges and Costs of the UK Immigration System for Russell Group Universities." https://russellgroup.ac.uk/media/5750/challenges-and-costs-of-the-uk-immigration-system-for-russell-group-universities.pdf.

Goodspeed, Travis. 2010. "Travis Goodspeed's Blog: Reversing an RF Clicker." *Travis Goodspeed's Blog*. https://travisgoodspeed.blogspot.com/2010/07/reversing-rf-clicker.html.

Home Office (UK Government). 2019. "Tier4 of the Points Based System: Guidance for Sponsors Document 2: Sponsorship Duties." https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/824003/Tier_4_Sponsor_Guidance_-_Doc_2_-_Sponsorship_Duties_2019-08_1.1.pdf.

Killian, Taylor. 2012. "Turning Point Clicker Emulation with Arduino and nRF24L01." Blog. *Taylor Killian*. http://www.taylorkillian.com/2012/11/turning-point-clicker-emulation-with.html.

"LCD Clicker Response Devices." n.d. *Turning Technologies*. https://www.turningtechnologies.com/lcd/.

Mooney, Nick. 2019. "Nickmooney/Turning-Clicker." https://github.com/nickmooney/turning-clicker.

"Pgfgantt - Gantt Chart Package." n.d. *TeX - LaTeX Stack Exchange*. https://tex.stackexchange.com/questions/63877/gantt-chart-package.

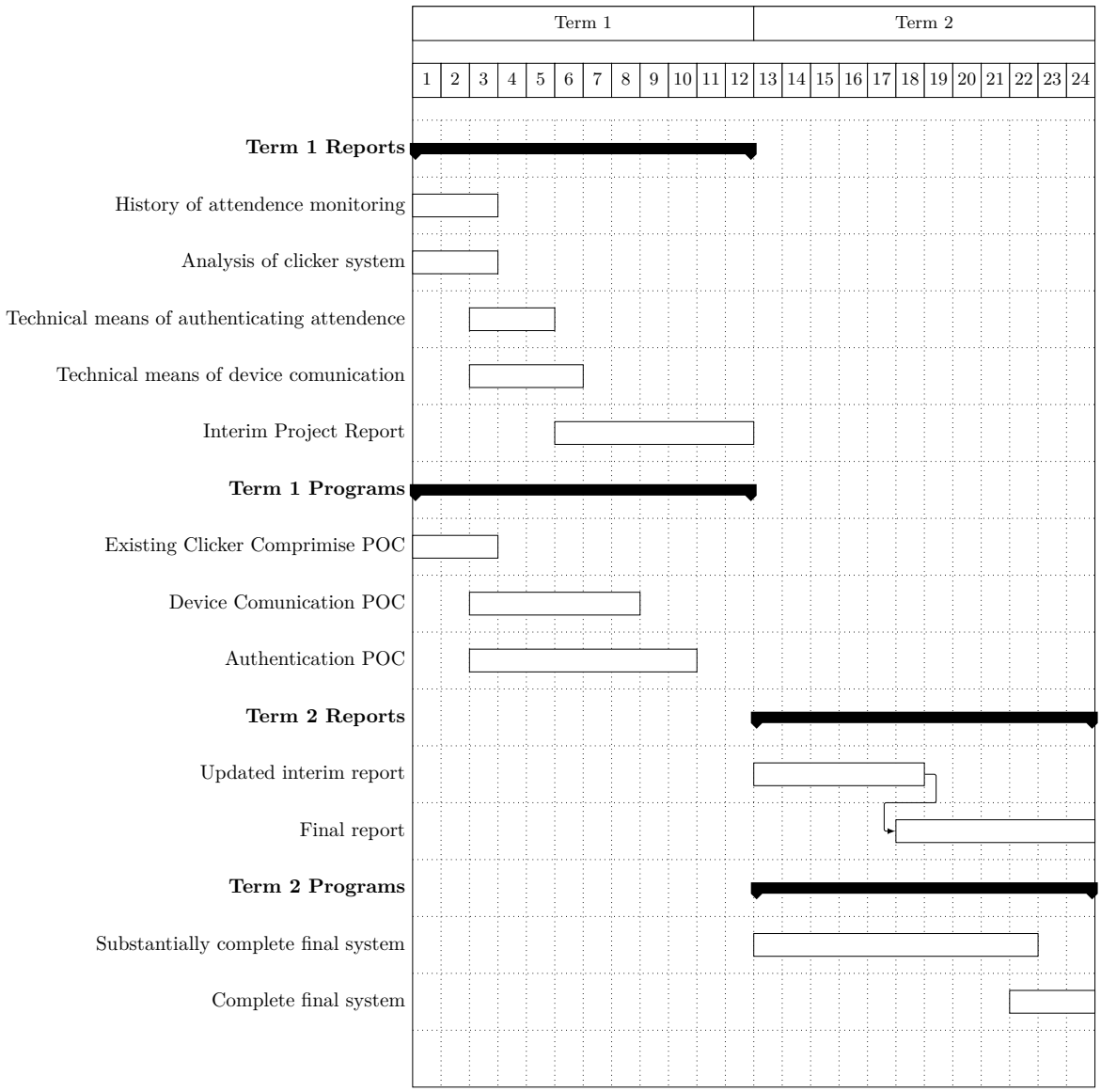
Royal Holloway Department of Computer Science. 2018. "DEPARTMENT OF COMPUTER SCIENCE Undergraduatestudent HANDBOOK." https://intranet.royalholloway.ac.uk/computerscience/documents/pdf/currentug/ug-student-handbook-2018-19.pdf.

Tom Pollard, Marvin Reimer, David San, Arco Mul, Matthew Gwynfryn Thomas, Jakub Nowosad, Dennis Weissmann, and W. Caleb McDaniel. 2016. "Template for Writing a PhD Thesis in Markdown." Zenodo. https://doi.org/10.5281/zenodo.58490.

Universities UK. 2019. "THE STUDENT VISA SYSTEM: PRINCIPLES TO REFORM." https://www.universitiesuk.ac.uk/policy-and-analysis/reports/Documents/2019/student-visa-system-principles-to-reform.pdf.

Appendices

Appendix 1



Investigation into Turning Point Clickers

Hugh Wells - 864564

28th October 2019

At Royal Holloway, for the 2018 intake of first year undergraduates it was decided that due to the class size, paper registers were infeasible. The clicker systems was proposed and developed. This uses a Turning Technologies "Response Card" (Figure 1) typically used to respond to interactive questionnaires as part of a slideshow, or with other proprietary software. The device communicates with a base station connected to a computer via USB (Universal Serial Bus) when a key option is pressed (eg. "1/A") and transmits the unique ID of the device and the key press. The message is acknowledged by the base station and the user is given visual affirmation on the device that their response was counted. The results are then stored in a proprietary format attached to the slideshow which is decoded, processed and analysed by the department using a collection of scripts, Excel spreadsheets and custom software.

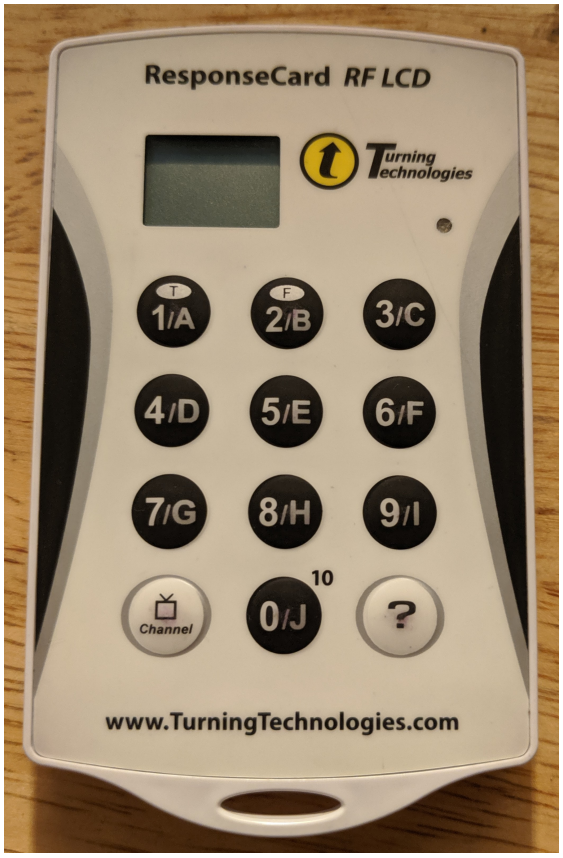


Figure 1: A Turning Technologies "Response Card"

Aims

The aim of this report is to investigate the Turning Point clickers to learn more about how they work in practice and to see to what extent it is possible to reverse engineer, intercept and spoof communications between the clicker and the basestation. The report will look at the theoretical technical specifications of the clickers, including any technical methods to validate and verify data. It will also include a practical investigation into the clickers, building on research previously carried out.

Background Research

Whilst researching the device I discovered the work of Goodspeed (2010) who reverse engineered a similar, but older device. This was achieved by dumping the firmware of the device allowing analysis of the way the device operated and how packets were structured and sent. A number of important discoveries were made: the System on Chip is a Nordic nRF24E1 chip, which is a Intel MCS-51 (8051 microcontroller) with an nRF2401 radio transceiver. The nRF2401 is a 2.4GHz,

serial radio transceiver.(Nordic Semiconductor ASA 2004) The datasheet lists a number of potential applications including telemetry, keyless entry and home security and automation. The chip also uses what the datasheet describes as a "3-wire serial interface." - this is otherwise know as SPI (Serial Peripheral Interface) and allows easy interface with devices such as a Raspberry Pi and Arduino. Secondly, there is no encryption and in the aforementioned research it was noted through analysis of the firmware that the packets take the structure of three bytes target MAC, 3 bytes source MAC and then a single byte for the button selection. A CRC is calculated and added by the radio.

This means that the source, destination and the parameter (the button pressed) are transmitted in cleartext with the only validation being a CRC (Cyclic Redundancy Check). The message is not signed, so there is no way to verify that a message has indeed come from the advertised source - this allows an attacker to arbitrarily spoof messages purporting to come from any source MAC address.

Cyclic Redundancy Check

CRCs are a method of error checking that are widely used in serial communications(Borrelli 2001). Traditionally, parity bits have been used to ensure data integrity across a communication channel. These indicate whether the data value is expected to be even or odd, but suffers from two major flaws. Firstly, it is not possible for you to determine where the corruption has ocurred in a piece of data - it is not possible to repair the message and it has to be discarded and resent. Secondly, if two corruptions occur then the data may end up passing the parity check but still be invalid. Additional measures (such as length checking, strict processing validation etc.) can be included to reduce the risks of bad data being processed. Whilst a parity bit is technically a 1 bit CRC (CRC-1), greater check values are now used as the greater bit value allows you to determine reliably how many bits have been corrupted and therefore protects against corruptions that fail to modify the overall parity.

CRCs works using modulo arithmetic on some generated polynomials. The general formula to calculate a CRC is:

$$CRC = remainderof\left[M(x) \times \frac{x^n}{G(x)}\right]$$

Where `M(x)` is the sum of the message used as the coefficients in a polynomial.

`G(x)` is known as the "generating polynomial" and is generally defined by the type of CRC you are using. For CRC-16 (which is the type used for the nRF24E1) `G(x)` is defined as:

$$G(x) = (1 \times x^{16}) + (0 \times x^{15}) + (1 \times x^2) + 1$$

This produces a value (the polynomial) when is then used to divide `M(x)`, with the remainder used as the CRC.

In this particular situation, it is possible to calculate the CRC on the nRF24E1 in the ShockBurst™configuration settings by setting the `CRC_EN` flag bit. This means the chip will calculate and append the CRC as well as strip and validate the CRC on messages received.(Nordic Semiconductor ASA 2004) In the Mooney (2019) implementation, CRC is calculated on the Arduino not on the nRF24E1 due to technical issues getting this working. The CRC is instead calculated and checked using an alternative library.(Frank 2019)

Hardware

It is possible to use any Arduino and nRF24E1 breakout board for this project - I elected to use the Arduino Nano and the cheapest nRF24E1 board I could find on eBay. The Arduino is placed on a breadboard and the nRF24E1 connected via Dupont leads to the requisite pins.

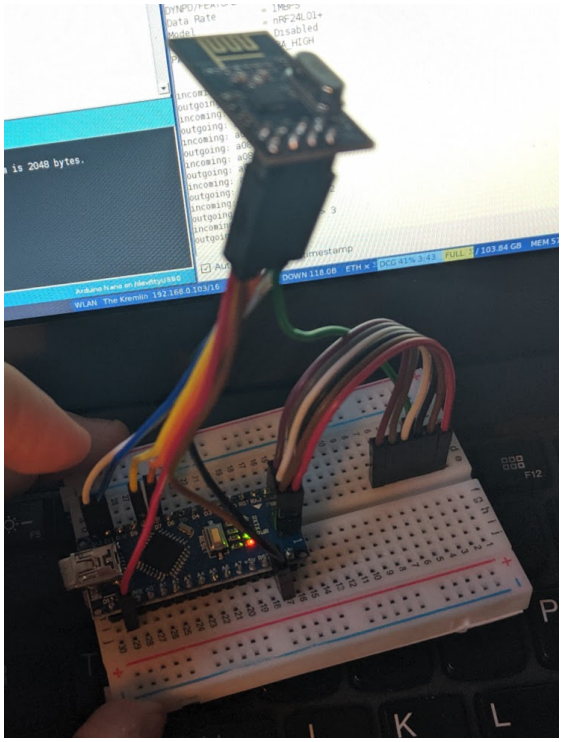


Figure 2: The wired up Arduino breadboard

Table 1: The chosen pinout.

Name	nRF24E1	Arduino
VCC	2	3.3v
GND	1	GND
IRQ	8	GND
CE	3	D7
CSN	4	D8
MOSI	6	D11
MISO	7	D12
SCK	5	D13

As always, this can be a somewhat complex operation and it did require consultation of not only the nRF24E1's advertised pin layout("1/2/3/5/10 Arduino NRF24L01+ 2.4GHz Wireless RF Transceiver Module UK Seller" n.d.) but also a setup guide for a similar, but not identical product from Sparkfun ("nRF24L01+ Transceiver Hookup Guide - Learn.sparkfun.com" n.d.).

The pinout I used is listed in Table 1

With the addition of a mini USB cable to connect the Arduino to your computer, the hardware setup is complete. It is important to note that the hardware required for the base station emulator is identical to that required for the clicker emulator.

Installation

The setup of the software environment on the computer is relatively simple and is based around the Arduino IDE which is available from: https://www.arduino.cc/en/main/software. It appears there is now an online version, but this has not been tested with this project.

Once the Arduino IDE is installed you should try flashing the Arduino with the example "blink" code, available from File → Examples → Basic.

Assuming you have installed the IDE correctly and have specified the serial port and Arduino type a blinking status LED will be present.

You can also compile and upload code from within other IDEs using plugins. VSCode has various plugins for this, setup of which is beyond the scope of this document. However, since it took me a while to discover this - it is important that you set up a VSCode Workspace so the Arduino Sketch file (containing running preferences such as the serial port, baud rate etc.) can be created and saved. Otherwise, you cannot upload any code.

To run the project code, two libraries are required - RF24 (a library to handle communication with the nRF24E1) and FastCRC (to calculate the CRC checksums). For the former, Sparkfun have published a guide on installation the RF24 library into the correct folder in your filesystem: https://learn.sparkfun.com/tutorials/nrf24l01-transceiver-hookup-guide/arduino-code The FastCRC library is installed in a similar fashion.

You should then be able to verify and upload the project code! If you get library errors, ensure the libraries have been imported correctly and the IDE shows them in the GUI.

It is also important to note that the baud rate used in the project is 115200 - without that set correctly it will not be possible to communicate with the Arduino via the IDE's serial console.

The Python scripts to manage serial communication with the Arduinos are designed to be run in Python3. No libraries not available via Pip are used, but what is installed by default varies per system so it is advised to try running the code and install any missing libraries as required.

Frequent Issues

Two issues encountered and which took significant effort to identify the root cause of are included with some basic debugging.

```
avrdude: stk500_getsync() attempt 1 of
10: not in sync: resp=0x00
```

This error occurs when either:

- The Arduino is disconnected or cannot be connected to: check the USB
- The wrong Arduino type is selected: check the board configuration and the type of CPU used

```
Cannot upload: device/resource busy
```

This occurs when the serial connection to the serial port is already in use. Ensure you don't have any serial consoles open, or any of the Python scripts running. If that doesn't work, you may be specifying the wrong serial port.

It is possible to identify a process using a serial port with `lsof` and then terminate this process using `pkill`.

Clicker Basestation

The script to emulate the basestation of the clicker system is made up of two parts, the Arduino code and Python script. The former is essentially an unmodified version of the Mooney (2019) file, which was developed a number of years ago. The interesting point here is that his work was developed on an entirely different device, and shows how widely this system has been deployed and the range of different devices available all using the same protocol. The Arduino code listens for a transmission on the configured channel and then checks the CRC (discussed above). It then prints out the received packet to the serial console and returns the "accepted" message to the clicker. This causes the clicker to flash a green LED instead of a red one to show proper acknowledgement to a message by the basestation.

The Python script is substantially modified from the original Mooney (2019) code and provides a basic serial program to interact with the basestation emulator. It connects to the serial port and parses the aforementioned outputs, before saving them to a specified output file in a CSV format. Because of the way the serial printing is formatted, regex is required - the expression is the work of Mooney (2019).

An example output from the Python script:

```
address_from,address_to,button
a080f3,31e600,1
a080f3,35a684,5
a080f3,396708,9
a080f3,396708,9
a080f3,387729,8
a08167,35459b,5
a08167,35459b,5
a08167,35459b,5
a08167,35459b,5
a08167,35459b,5
a08167,35459b,5
a08167,35459b,5
a08167,35459b,5
```

As you can see, the from address (the clicker address), the to address (the hardcoded basestation address) and the button pressed are included. This is a significant improvement on the output from the current solution as the output is in a text file format (CSV) which does not require preprocessing from a proprietary format.

The output on the serial console is shown in Figure 3.

```
*** booting ***
STATUS           = 0x0e RX_DR=0 TX_DS=0 MAX_RT=0 RX_P_NO=7 TX_FULL=0
RX_ADDR_P0-1     = 0xe7e7e7 0x123456
RX_ADDR_P2-5     = 0xc3 0xc4 0xc5 0xc6
TX_ADDR          = 0xe7e7e7
RX_PW_P0-6       = 0x00 0x06 0x00 0x00 0x00 0x00
EN_AA            = 0x00
EN_RXADDR        = 0x02
RF_CH            = 0x29
RF_SETUP         = 0x05
CONFIG           = 0x07
DYNPD/FEATURE    = 0x00 0x00
Data Rate        = 1MBPS
Model            = nRF24L01+
CRC Length       = Disabled
PA Power         = PA_HIGH


incoming: a080f334b6a5 --> 4
outgoing: a080f30673b8
incoming: a080f335a684 --> 5
outgoing: a080f30673b8
incoming: a080f332d663 --> 2
outgoing: a080f30673b8
incoming: a080f333c642 --> 3
outgoing: a080f30673b8
incoming: a080f331e600 --> 1
outgoing: a080f30673b8
incoming: a080f332d663 --> 2
outgoing: a080f30673b8
incoming: a080f333c642 --> 3
outgoing: a080f30673b8
```

Figure 3: Clicker basestation serial console

Clicker emulator

The hardware for the emulation of a clicker is exactly the same as that required for a basestation - the difference is purely in the code and associated processing of data. My high level plan was:

- Modify Arduino script to send to the hardcoded basestation address
- Modify Arduino script to take input of "sent from" addresses over serial
- Write a Python utility script to feed the Arduino with clicker addresses to emulate

I first decided to tackle getting addresses into the Arduino over a serial connection - it seems deceptively simple! I intially planned to use the `String` components of the Arduino `Serial` library - `Serial.readStringUntil`("Arduino Reference" n.d.), for example. This had the advantage of avoiding having to worry about lower level buffer constructs and also allowed abstraction from having to handle reading from the Serial buffer until a certain point (end of string etc.). This did not work for a few reasons. The addresses being sent are actually binary values, usually represented in hexadecimal, and the String library cannot deal with this raw data. The "until" also proved to be somewhat unreliable and would not terminate necessarily on a line feed. This may well have been the product of other issues, however.

If you cannot use the String abstraction, you have to handle raw `Chars` - these are individual bytes that you place into a buffer of suitable

length. Arduinos use Harvard rather than Von Neumann CPU architectures so the data and system memory (RAM) are on different buses - this makes a buffer overflow exploit difficult, but not impossible. You do therefore need to be careful when handling buffers to ensure you do not deliberately overflow a buffer.

The basic code (encapsulated in a loop) is as follows:

```
if(Serial.available() > 0 &&
    counter != BUFSIZE){

    char incoming = Serial.read();

    if (incoming == '\r') {

    } else {
        incomingData[counter] = incoming;
        counter++;
    }
}
```

When the Serial port is available and the counter has not reached the buffer size - Read a single character from the Serial port - If it is a `\r` (line feed) then handle that case - Otherwise, place it in the buffer and increment the counter

This is all well and good but you need to get an address over the Serial connection in the first place! Sadly, you cannot just send `OxA0` - by doing so you are sending the literal ASCII codes for each individual char (eg. `A` = 65) you need to send the "hex values" for each byte, which is then a single char. One way to get a hex ASCII value is via `echo -ne "OxA0` which will print out the char you want. Sadly, when I sent this to the Arduino (using Screen) I ended up filling the buffer several times over with values that were completely unrelated. I still do not understand the cause of this issue - I suspect it may have been triggering some Screen escape codes.

I then moved to using a Python script which was more easily rerun. In order to both read and send data over the Serial connection I had two threads one to receive data and another to send characters as required. This was based on a StackOverflow post and used some example code, but the concept is not complex. In order to be able to both send and receive characters on a serial line in Python you need at least two threads - one to process anything received on the serial line and print it to the console, and another to send data when instructed without blocking the receiving thread:

```
conn.write(str.encode("160"))
conn.write(str.encode("160"))
conn.write(str.encode("160"))
conn.write(str.encode("\r"))
```

This more predictably populated the buffer at the other end, as shown in Figure 4. In the clicker basestation you can see one packet received - this is from other (presumably IoT) devices that use the same channel and addressing structure. You can also see the attempt to send data on the right with the various interesting echos back with different values to that which are sent, from the Arduino. The one outgoing packet shown to be sent was actually hardcoded and exposed another bug.

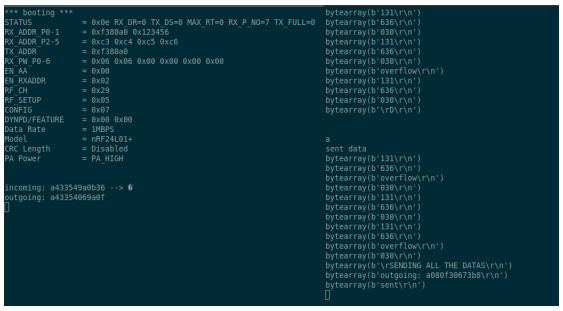


Figure 4: Clicker basestation serial console on the left, clicker emulator Python Script on the right

It turns out that although the outgoing packet is correctly formatted (when compared to incoming packets from a clicker as received by the emulator basestation) no packet is actually sent, or at least received at the other end. This is a fairly critical error and I think may stem from my misunderstanding over how the code works (which address is which). I have not yet resolved this and I may not - I am quickly going down a rabbit hole of ASCII encoding already.

Conclusion

I have successfully implemented (using code from Mooney (2019)) a clicker basestation and associated Python script which I believe actually has a practical use, to replace the current PowerPoint slide system. The emulation of the clickers themselves has proved more tricky and the adaptation of the code has been difficult - I am looking at how to resolve this but I think it will require rewriting a substantial portion of the original code. The encoding issue over serial is a problem, but not a huge disaster at the moment. I am unlikely to expend significant effort on this

now, until I can guarantee I can actually send packets between the two Arduinos.

Acknowledgements

Thanks to Tom Pollard et al. (2016) for the front cover template which I have adapted, Marco Torchiano (2015) for the Pandoc table preamble and Cohen and Matos (2013) for the Final Year Project guide and suggested layouts.

Bibiography

"1/2/3/5/10 Arduino NRF24L01+ 2.4GHz Wireless RF Transceiver Module UK Seller." n.d. *eBay*. Accessed October 25, 2019. https://i.ebayimg.com/images/g/rkQAAOSw7k9dHxoc/s-l400.jpg.

"Arduino Reference." n.d. Accessed October 28, 2019. https://www.arduino.cc/reference/en/language/functions/communication/serial/readstringuntil/.

Borrelli, Chris. 2001. "IEEE 802.3 Cyclic Redundancy Check." Xilinx. https://www-inst.cs.berkeley.edu/~cs150/fa04/Lecture/xapp209.pdf.

Cohen, Dave, and Carlos Matos. 2013. "Third Year Projects – Rules and Guidelines." Royal Holloway, University of London.

Frank. 2019. "FrankBoesing/FastCRC." https://github.com/FrankBoesing/FastCRC.

Goodspeed, Travis. 2010. "Travis Goodspeed's Blog: Reversing an RF Clicker." *Travis Goodspeed's Blog*. https://travisgoodspeed.blogspot.com/2010/07/reversing-rf-clicker.html.

Marco Torchiano. 2015. "How to Solve Longtable Is Not in 1-Column Mode Error?" *TeX - LaTeX Stack Exchange*. https://tex.stackexchange.com/questions/161431/how-to-solve-longtable-is-not-in-1-column-mode-error.

Mooney, Nick. 2019. "Nickmooney/Turning-Clicker." https://github.com/nickmooney/turning-clicker.

Nordic Semiconductor ASA. 2004. "Single Chip 2.4 GHz Transceiver - nRF2401." https://www.sparkfun.com/datasheets/RF/nRF2401rev1_1.pdf.

"nRF24L01+ Transceiver Hookup Guide - Learn.sparkfun.com." n.d. Accessed October 15, 2019. https://learn.sparkfun.com/tutorials/nrf24l01-transceiver-hookup-guide.

Tom Pollard, Marvin Reimer, David San, Arco Mul, Matthew Gwynfryn Thomas, Jakub Nowosad, Dennis Weissmann, and W. Caleb McDaniel. 2016. "Template for Writing a PhD Thesis in Markdown." Zenodo. https://doi.org/10.5281/zenodo.58490.

Browser Fingerprinting - a practical analysis

Hugh Wells - 864564

14th November 2019

Tracking a user within a website (or indeed across the internet) has been possible either by checking the IP address of the origin or by use of cookies for state management, the mechanism of which was described in the 1997 RFC (Kristol and Montulli n.d.). As part of the new clicker system it is envisioned that some method of identifying the device the student is signing in on will be required, to allow detection of one device signing multiple people in. The IP address is too general - those connected to college WiFi or using the same network and mobile phone mast would have identical IP addresses, session cookies can be set to have long expiries but a user can remove them from their browser regardless. In order to track devices with some degree of certainty, it is therefore necessary to look at alternative means of identification.

This type of tracking is not uncontroversial. From 2012 to 2014, Verizon (a US network carrier) injected unique identifiers into network traffic without their customers being aware of such 'supercookies' being attached to their data. (Brodkin 2016) ("Verizon, AT&T Tracking Their Users with 'Supercookies' - the Washington Post" n.d.) It transpired that not only were Verizon using these supercookies themselves, but third parties had discovered their existence and were using them to track individual devices for purposes such as advertising. The FCC's investigation determined that Verizon should have sought explicit opt-in consent from customers for the direct sharing of what the FCC referred to as UIDH (unique identifier headers) and given the option for customers to opt out of their use by Verizon internally. A specific case cited related to a third party advertiser using supercookies to continue tracking customers after they had explicitly removed normal cookies from their devices. In the UK, the European Union General Data Protection Regulations apply. The law contains specific provision for what it calls "Special category Data". This is "personal data which the GDPR says is more sensitive, and so needs more protection." (GOV.UK n.d.) and includes "biometric" data which traditionally has been used to refer to specific human characteristics (such as retina data) however could arguably be applied to specific characteristics of a device a user owns; in the same way that an IP address is considered Personally Identifiable Information.("EUR-Lex - 62014CJ0582 - EN - EUR-Lex" n.d.) The Electronic Frontier Foundation run an online service, ("Panopticlick About" n.d.), which will attempt to fingerprint your browser in a sandbox, displaying the results. Their test, however, is somewhat dated as Boda et al. (2012) observes - relying on Flash or Java plugins to properly fingerprint the available fonts. They also note the potential for plugin detection - where the available plugins to the browser are queried and used as identifiers - given how unique the combination can be.

Browser fingerprinting takes various characteristics within a browser via the JavaScript API (and formally via Flash and Java virtual machines too) and uses these to essentially narrow down a browser to a very small intersection of sets. Taking a simplistic example, a browser with resolution of 1080x760px might have 100 possible other configurations, if we then also include the CPU class that might narrow down the intersection to 50 devices. Then taking the system language it might be possible to reduce the intersection to 40 devices, and so on. On their own, these identifiers are not unique - it is only when they are combined that you reduce the probability of finding another device with an identical configuration. There are some examples (Mozilla n.d.) of identifiers that might be used:

- `browserSettings.openUrlbarResultsInNewTabs`: whether URL bar autocomplete search results open in a new tab or not
- `browserSettings.homepageOverride`: the current value of the home button URL
- `runtime.PlatformOs`: the platform Operating System
- `runtime.PlatformArch` the platform architecture

These seem fairly obvious but there are more advanced techniques such as HTML5 web canvas and WebGL fingerprinting. WebGL is a graph-

ics API that allows the programmatic drawing and rendering of images on a canvas.(Kobusińska, Brzeziński, and Pawulczuk 2017) The technique used is to draw an image onto the canvas and then convert the result back into text which can be compared against other results to determine uniqueness. In this particular study the large changes seen in a controlled environment and relatively large execution time led that WebGL fingerprinting contained too much entropy to be useful as an additional identifier to measure. The Kobusińska, Brzeziński, and Pawulczuk (2017) method was not described in detail however a novel approach was presented by Mowery and Shacham (2012) which utilized both HTML5 and WebGL. Key to their approach was the rendering of text which they matched against known samples (using automated methods that are not relevant here) - this was done with both WebFont (HTML5) and standard font face rendering. The WebGL test used a black and white image containing 200 polygons - derived from the ISO 12233 standard - and the various rendered images were then subtracted from a control to give a "diff" between the generated outputs for each browser, which could then be compared. The paper contains some illustrations to demonstrate the point.

The actual result from a browser fingerprint is a hash of all of the attributes selected to identify the browser. This hash can then be stored, to be checked against future computations of the hash. As Ashouri (n.d.) notes into research reverse engineering fingerprinting solutions, these hashes are increasingly calculated server side with the raw values extracted sent to the server. Not only is this not ideal as there is a significant transfer of quite sensitive data, but the calculations are carried out in a black box that cannot be analysed. Another variant mentioned by Ashouri (n.d.) is commercially available scripts that are loaded from a third party, returning the result directly to them for analysis and storage. An example here is provided by Adyen("Device Fingerprinting Adyen Docs" n.d.) who provide customers with a script used to calculated a browser fingerprint, for the purposes of their own internal fraud prevention and 3DSecure Version 2.

Proof of Concept

There are two commonly used and freely available libraries for calculating browser fingerprints - Fingerprint2 (V 2019) and ("ClientJS" n.d.), the latter of which implements some of Fingerprint2.

In order to test both libraries I designed a simple HTML webpage with some basic HTML elements to visualize the fingerprint outputs. I subsequently wrote according to the respective manuals some basic fingerprinting for both Fingerprint2 and ClientJS.

It is important to note the timeout is set to allow a certain amount of time for the page to load, as partially loaded pages generate inconsistent fingerprints.

In Figure 1 you can see a demonstration:

Your Fingerprint2 is: `e0c4d17cbea1bdf2b801011890436d87`

Your ClientJS fingerprint is: `4217023516`

Figure 1: The initial output of the fingerprint output test

Upon a refresh in Figure 2 you will note that the ClientJS fingerprint remains static, whilst the Fingerprint2 example does not. The exact reasons for this will be down to the specific parameters each chooses to use in fingerprinting. There are many options - it would take a long time to fine tune these algorithms!

Your Fingerprint2 is: `49a352086af089a875fd8b2a020688e7`

Your ClientJS fingerprint is: `4217023516`

Figure 2: The refreshed output of the fingerprint output test

I also compared the outputs in an incognito window and in Figure 3 again the Fingerprint2 results do not match, whilst the ClientJS fingerprint has correctly identified the browser - despite it being in incognito mode and therefore with no cookies and no tracking.

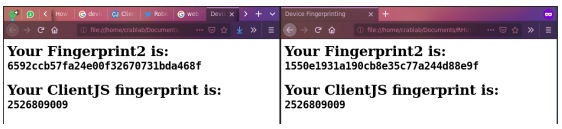


Figure 3: Left: normal browser, Right: incognito mode

Finally in Figure 4, I compared the results across two browsers - Chrome and Firefox. Both Fin-

gerprint2 and ClientJS identify them as different browsers, which is to be expected!

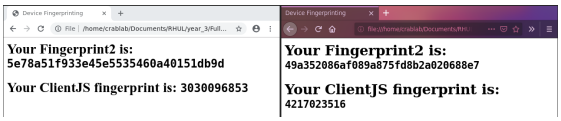


Figure 4: Left: Chrome, Right: Firefox

Acknowledgements

Thanks to Marco Torchiano (2015) for the Pandoc table preamble and Cohen and Matos (2013) for the Final Year Project guide and suggested layouts.

Bibiography

Ashouri, Mohammadreza. n.d. "A Large-Scale Analysis of Browser Fingerprinting via Chrome Instrumentation," 12.

Boda, Károly, Ádám Máté Földes, Gábor György Gulyás, and Sándor Imre. 2012. "User Tracking on the Web via Cross-Browser Fingerprinting." In *Information Security Technology for Applications*, edited by Peeter Laud, 31–46. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer. https://doi.org/10.1007/978-3-642-29615-4_4.

Brodkin, Jon. 2016. "Verizon's 'Supercookies' Violated Net Neutrality Transparency Rule." *Ars Technica*. https://arstechnica.com/information-technology/2016/03/verizons-supercookies-violated-net-neutrality-transparency-rule/.

"ClientJS." n.d. Accessed November 13, 2019. https://clientjs.org/.

Cohen, Dave, and Carlos Matos. 2013. "Third Year Projects – Rules and Guidelines." Royal Holloway, University of London.

"Device Fingerprinting Adyen Docs." n.d. Accessed November 13, 2019. https://docs.adyen.com/risk-management/device-fingerprinting.

"EUR-Lex - 62014CJ0582 - EN - EUR-Lex." n.d. Accessed November 14, 2019. https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX%3A62014CJ0582.

GOV.UK. n.d. "Guide to the General Data Protection Regulation." *GOV.UK*. Accessed November 14, 2019. https://www.gov.uk/government/publications/guide-to-the-general-data-protection-regulation.

Kobusińska, Anna, Jerzy Brzeziński, and Kamil Pawulczuk. 2017. "Device Fingerprinting: Analysis of Chosen Fingerprinting Methods:" In *Proceedings of the 2nd International Conference on Internet of Things, Big Data and Security*, 167–77. Porto, Portugal: SCITEPRESS - Science; Technology Publications. https://doi.org/10.5220/0006375701670177.

Kristol, David M., and Lou Montulli. n.d. "HTTP State Management Mechanism." Accessed November 14, 2019. https://tools.ietf.org/html/rfc2109.

Marco Torchiano. 2015. "How to Solve Longtable Is Not in 1-Column Mode Error?" *TeX - LaTeX Stack Exchange*. https://tex.stackexchange.com/questions/161431/how-to-solve-longtable-is-not-in-1-column-mode-error.

Mowery, Keaton, and Hovav Shacham. 2012. "Pixel Perfect: Fingerprinting Canvas in HTML5." In *Proceedings of W2sp 2012*, edited by Matt Fredrikson. IEEE Computer Society.

Mozilla. n.d. "Browser Extensions." *MDN Web Docs*. Accessed November 15, 2019. https://developer.mozilla.org/en-US/docs/Mozilla/Add-ons/WebExtensions.

"Panopticlick About." n.d. Accessed November 14, 2019. https://panopticlick.eff.org/about.

V, Valentin. 2019. "Valve/Fingerprintjs2." https://github.com/Valve/fingerprintjs2.

"Verizon, AT&T Tracking Their Users with 'Supercookies' - the Washington Post." n.d. Accessed November 14, 2019. https://www.washingtonpost.com/business/technology/verizon-atandt-tracking-their-users-with-super-cookies/2014/11/03/7bbbf382-6395-11e4-bb14-4cfea1e742d5_story.html.

Appendix

Appendix 1

```
<html>

<head>
    <title>Device Fingerprinting</title>
    <script type="text/javascript" src="fingerprint2.js"></script>
    <script src="client.min.js"></script>
</head>

<body>
    <h1>Your Fingerprint2 is: <code id="fp1"></code></h1>
    <h1>Your ClientJS fingerprint is: <code id="fp2"></code></h1>
</body>

</html>
```

Appendix 2

```
setTimeout(function() {
    // Fingerprint 2
    var options = {
        excludes: {
            userAgent: true,
            language: true
        }
    }
    // Based off https://github.com/Valve/fingerprintjs2#usage
    Fingerprint2.get(options, function(components) {
        var values = components.map(function(component) {
            return component.value
        })
        var murmur = Fingerprint2.x64hash128(values.join(''), 31)

        document.getElementById("fp1").innerHTML = murmur;
    })

    // ClientJS
    var client = new ClientJS();

    var fingerprint = client.getFingerprint();

    document.getElementById("fp2").innerHTML = fingerprint;

}, 500)
```

Communication methods for attendance monitoring

Hugh Wells - 864564

20th November 2019

Abstract

At Royal Holloway, for the 2018 intake of first year undergraduates it was decided that due to the class size, paper registers were infeasible. The clicker system was proposed and developed which has been discussed in previous reports; particularly with regard to the system security. The key advantage of the clicker system was that it did not rely on the signature of students (it used a wireless electronic device) and therefore did not require the laborious administration overheard of transposing the signatures into an electronic database. Fundamentally, it was not more secure and this element of an improved system will be discussed in a future report.

One of the key advantages that the clicker system had over the registers was that it utilise wireless communication such that students could 'click in' on a short range handheld device, rather than pass around a physical register sheet. The range of the clickers is advertised as being around a 200ft radius ("C21 Clicker" n.d.) which in the context of a lecture theatre, is actually quite large! Based on practical experience (and not wishing to go down a rabbit hole to find out definitively), I postulate that for most lecture theatres that would include the entire room plus a short distance outside - taking into account the attenuation of the signal through the walls.

Considering the range of other wireless communication technologies:

- Radio Frequency Identification (RFID): a maximum of 20ft (Nikitin and Rao 2006) with a passive tag.
- Near Field Communication (NFC): a few centimeters (Fischer 2009)
- Bluetooth: "short range" (Heydon 2013)

Student ID Cards

At the beginning of the 2019 Winter Term, the College began issuing new student ID cards and reissuing older cards. The new cards were found to contain a MIFARE Classic chip (Figure 1). The MIFARE chip was originally released in 1994 by what became NXP (Mayes and Cid 2010) as a product primarily for mass transit cards, but also later for door key cards and the like. The MIFARE design was very proprietary with only the (48 bit) key length disclosed to the security community. For example, the chip used an undisclosed encryption algorithm developed by Phillips (who became NXP) that had not been researched or investigated by the wider security community. As is well accepted in Kerckhoff's Principle, we should always assume the attacker has the maximum knowledge of the system including details of the encryption algorithms used, nonce generation algorithms and other design features. The only item assumed not to be known by default is any private keys generated. This principle ensures that any tendency to default to "security by obscurity" is avoided as, as in the case of MIFARE, if your security model relies on design details being disclosed it is very vulnerable to espionage, reverse engineering or simply errors in parties disclosing too many details.

In December 2007 at the Chaos Computer Conference (Nohl and Plötz n.d.) (Nohl, Evans, and Plotz, n.d.), a presentation was given where researchers presented analysis where it was demonstrated it was possible to generate the nonces used in the chip (as the randomness is determined by the

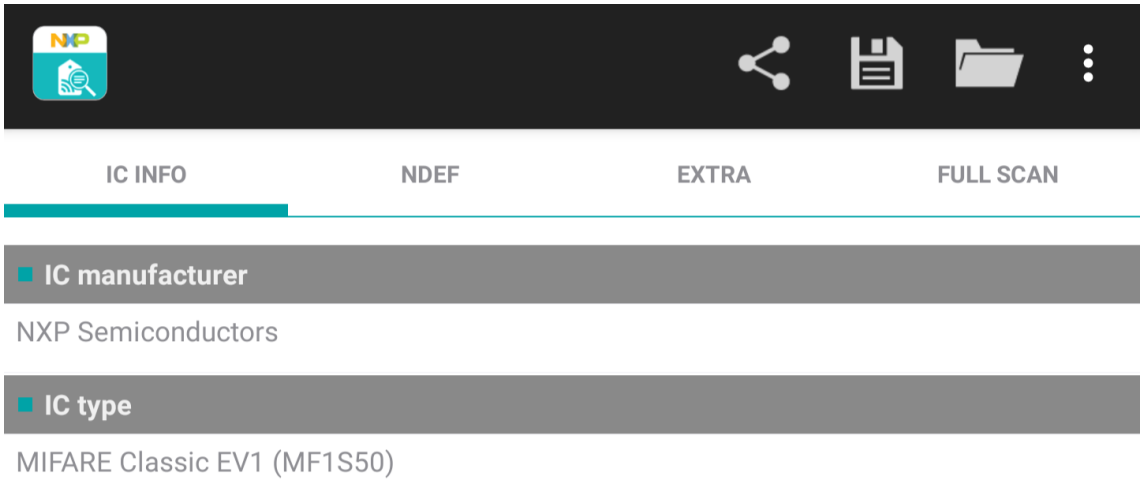


Figure 1: Screenshot of the tag read from the NXP "TagInfo" Android application

number of clock cycles since the chip is initialized) and by analyzing 27 of mutual authentication captures repetition of the challenge response was observed. It was also shown by trial by error in bit flipping the unique identifier transmitted by the card was linked to the keys used and such there exists for each session key as a result of the mutual authentication, for a given key and unique identifier. This allows recovery of the key used based on the unique identifier presented by the chip and the challenge response sent.

This has been developed further and it has been possible to generate key rainbow tables (as they only have a 48 bit length) and use these tables to carry out brute force searches on encrypted sectors for arbitrary cards. Note in the dump in Appendix 1 sector 6 is unreadable - this is the secured sector for which a key is needed for the reader to access the sector. Using software implementing the later nested attack on MIFARE ("Nfc-Tools/Mfcuk" 2019) and an Android application to interrogate the card (Klostermeier 2019) it was possible to recover this sector:

```
30313836343536342020202020202020
00000000000000000000000000000000
00000000000000000000000000000000
------------7877880045FA49A0C327
```

Where, `45FA49A0C327` is the key, `7877880045` the Access Control bits and the hexadecimal beginning `303138...` the stored data. Converted from hexadecimal to ASCII it reads `01864564`, which is my student ID number (as printed on the front of the card). With the key it is possible to rewrite any student ID card (or any MIFARE Classic card for that matter) with any ID number, or arbitrary data. Through a separate vulnerability, it is possible to retrieve any card ID number from the College LDAP extended attributes. So, it is possible to write any of these cards (used for access control, payments etc.) with any ID number and they offers no security above the magnetic stripe technology in use on the older cards.

I therefore propose not to use the student ID cards for authentication, as they are demonstrably insecure with little knowledge of the MIFARE technology or cryptography.

Bluetooth Low Energy

Section with reference to Gomez, Oller, and Paradells (2012), Heydon (2013), Townsend et al. (2014) and Bluetooth SIG (2010). Given the former books are drawn from the specification and crossover is large between all three sources, exact citation is not provided.

In the 4th Edition of the Bluetooth Specification (Bluetooth SIG 2010), issued by the Bluetooth

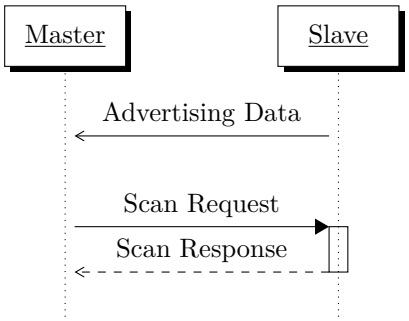
Special Interest Group the Bluetooth Low Energy devices were introduced. These devices were to in addition to what is known as Bluetooth Classic - the most well known version of Bluetooth used in all manner of consumer accessories from headphones to keyboards, and now adopted in scientific and industrial applications. Three classes of devices were available: Bluetooth Classic (only), Bluetooth Low Energy and Bluetooth Smart Ready; the latter supporting both Classic and Low Energy protocols. The intention behind Low Energy was to go almost in the opposite direction to Classic - a very low bandwidth device which would not remain constantly connected designed around applications that issued sporadic commands to retrieve data or initiate control sequences. In comparison to Bluetooth Classic at the time, Bluetooth Low Energy has a theoretical bandwidth of 1MB/s vs 54 MB/s with Classic (Heydon 2013) (Gomez, Oller, and Paradells 2012). Low Energy operates on the 2.4GHz frequency band with a channel spacing of 2MHz (40 channels) with 3 channels dedicated for advertisement broadcasts - channels 37, 38 and 39 (Townsend et al. 2014). In order to minimize interference with other Bluetooth devices and other standards on the same frequency (WiFi, Bluetooth Classic, ZigBee) frequency hopping is used where the channel is changed upon each interaction based on the following formula (Townsend et al. 2014):

```
channel = (curr_channel + hop) mod 37
```

The `hop` is predetermined at the initial authentication with the Bluetooth device and remains constant throughout the life of the connection.

Gaussian Frequency Shift Keying (GFSK) is used as the modulation for transmitted signals. This is fairly common for devices operating on the 2.4GHz band and was filed for patent in 1998 Hsiang-Te Ho, Hsing-Ya Chiang, and Sheau-Chien Wang (n.d.). When radio signals are transmitted they are modulated (mixed) with a carrier signal at the frequency of transmission. GSFK uses a filter based on the Gaussian function. The exact reasons for doing this are beyond the scope of this report, but they relate to reducing the effects of interference during transmission.

The Bluetooth 4.0 specification defined two actors: a master and a slave. During the asymmetric connection process, the slave advertises itself on the advertising channels with a 31 byte packet. It is possible to send an additional 31 byte packet with additional data upon interrogation by a Master (sending a Scan Request packet) - this is Active Scanning and Passive Scanning is merely monitoring the data sent on the advertising channels.



In Bluetooth 4.0 the slave may be connected to one master device with master devices permitted to connect to a number of slave devices. This was updated in version 4.1 of the specification (Bluetooth SIG 2013) to allow slaves to have multiple connections to a master at any one time, although slave to slave communication remains unsupported. Low Energy uses Time Division Multiplexing to achieve long periods with the radio off (to conserve power) transmitting and receiving packets only at pre-determined intervals. This can give a battery life measured in years, but does mean the bandwidth is limited, as previously discussed.

The Bluetooth stack uses L2CAP (Logical Link Control and Adaptation Protocol) at second lowest level of the stack (the Physical Layer handling the modulation and transmission of packets is below) and this is responsible for the assembling of packets and managing the Physical layer to ensure downstream units, such as the HCI, are not overwhelmed.

Bluetooth defines a number of modules including the Generic Attribute Profile and the Generic

Access Profile, which help abstract services away from the hardware and provide common interfaces between devices. The exact mechanisms of these and the general design is beyond the scope of this report - we simply accept that the Bluetooth specification provides these services for our convenience.

However, the security mechanisms provided are of interest especially given the weaknesses noted in MIFARE.

Security Manager

One module is the Security Manager which handles the mutual generation of keys for encrypted communications and interfaces with the L2CAP module and is only present in Low Energy devices (in Classic device it is integrated in the Controller). Using hardware modules it is able to provide various cryptographic functions for the key exchanges.

The security function used is an AES-128-bit block cipher. The Advanced Encryption Standard (National and Institute of Standards and Technology 2001) was chosen as a result of a RFP issued by the United States National Institute of Standards and Technology looking to replace the then aging and provably broken Data Encryption Standard (DES), developed many years before. The algorithm chosen was Rijndael (Daemen 1999) and operated on 128, 196 and 256 bit blocks and is a symmetric cipher. AES works by applying a round function to the data input using the expanded key as a state. On each round a byte substitution, then row movement and finally column function is applied to the data. Each round the designed part of the key is XOR'd with the state. There are 10, 12 or 14 rounds dependant on key length and the last round does not mix the columns. Applying the algorithm in reverse is possible and allows decryption of the ciphertext.

When Bluetooth devices are paired an STK (Short Term Key) is generated to encrypt data symetricall with AES in transit. Pairing is carried out via three methods:

- Just Works: values to generate STKs are exchanged with no encryption over the link offering no man-in-the-middle protection. Interception of the values allows an attacker to generate their own copy of the STK and decrypt and potentially spoof further communications.
- Passkey: a 6 digit numeric code is assigned to the Slave device and on pairing the Master requires user input of this code which is then used to generate a key to encrypt exchanged values for the STK generation.
- Out of band: values are exchanged outside of the Bluetooth protocol to allow encryption of the values to generate the short term key.

It should be noted that in the proposed implementation, Just Works pairing will be used. It is possible to use Passkey pairing however since each student would know the identical code it would offer no protection against them intercepting their own packets and thus has negligible security benefits as they are the main adversaries. There are technical limitations to dynamically changing the Passkey on Bluetooth Low Energy Slaves and this is currently not properly supported.

BLE Device Investigations

Investigations into practical applications and small projects with Bluetooth Low Energy yielded several articles mentioning the HM-10. (Loginov 2019) The HM-10 is based on a Texas Instruments BLE System On Chip mounted as a daughter board to breakout a serial UART connection. At this time only BLE devices are officially supported via the Web Bluetooth APIs. (WebBluetoothCG 2019)

As this device provides a serial interface, an additional microprocessor will be required to interface with the device and to receive, process and then respond to data sent to the device. A Raspbery Pi was chosen over an Arduino due to the increased processing power, onboard networking and full operating system. This also allows for higher level languages like Python to be used instead of C++, speeding up development. As no analogue inputs are required there is no real advantage to using an Arduino here. A Rasperry Pi Zero was configured to run in headless mode with a terminal

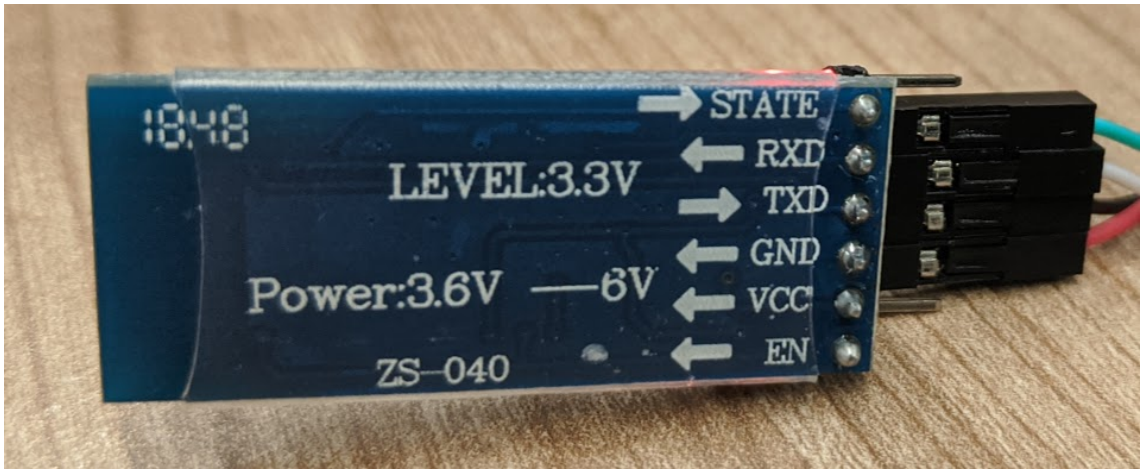


Figure 2: A HM-10 module

over USB (gbaman n.d.) however it transpired that whilst the Raspberry Pi does have two UARTs available, one is used for the onboard Bluetooth and the UART used for the Linux console is shared with the UART exposed on the GPIO. ("The Raspberry Pi UARTs - Raspberry Pi Documentation" n.d.) It was therefore necessary to change approach and a Raspberry Pi 2B was provisioned with SSH enabled (add an empty file: `/boot/ssh`) was configured such that direct a ethernet connection to the laptop was possible, with ethernet passthrough to the laptop's wireless connection.

The HM-10 was then attached with Dupont leads to the GPIO of the Raspberry Pi using pinouts for the Pi and the markings on the HM-10 breakout board. ("The Pi4J Project – Pin Numbering - Raspberry Pi 2 Model B" n.d.)

Initially, the device was detected on the serial port and a connection could be established. However, there was no data sent by the device and no acknowledgement of data sent down the serial port. The HM-10 uses AT commands which are commonly found in GSM modules and the like - the HM-10 likely uses these given the industry crossover. The manual for the HM-10 (JNHuaMao Technology Company 2014) lists a few basic AT commands which can be used to interrogate basic data such as the device status:

- `AT` should return the device status: `OK` or `OK/LOST`
- `AT+ADDR?` should return the device MAC: `OK+ADDR:{MAC address}`

The datasheet also describes that a string over 80 characters should be sent to take the device out of standby. The string data is not relevant, just the length.

The manual states the serial parameters are:

- 9600 Baud
- No parity
- 1 stop bit
- 8 bit byte length

A few attempts were made using these settings and different serial decoders (Minicom, Screen) - no response was received. Appendix 2 contains an output from Minicom for one attempt (including configuration).

It transpired that AT commands are actually a time based command set. They do not rely on a carriage return or line feed to indicate the end of a command, but they do timeout - within 100ms usually. (Milosevich 2006) A timeout on a valid command will cause the command to be executed and the result returned - an invalid command may generate no response at all.

Looking at the Raspberry Pi forums identified a small number of questions relating to this, including a thread where a user had reported some working test code. ("How to Communicate with Serial

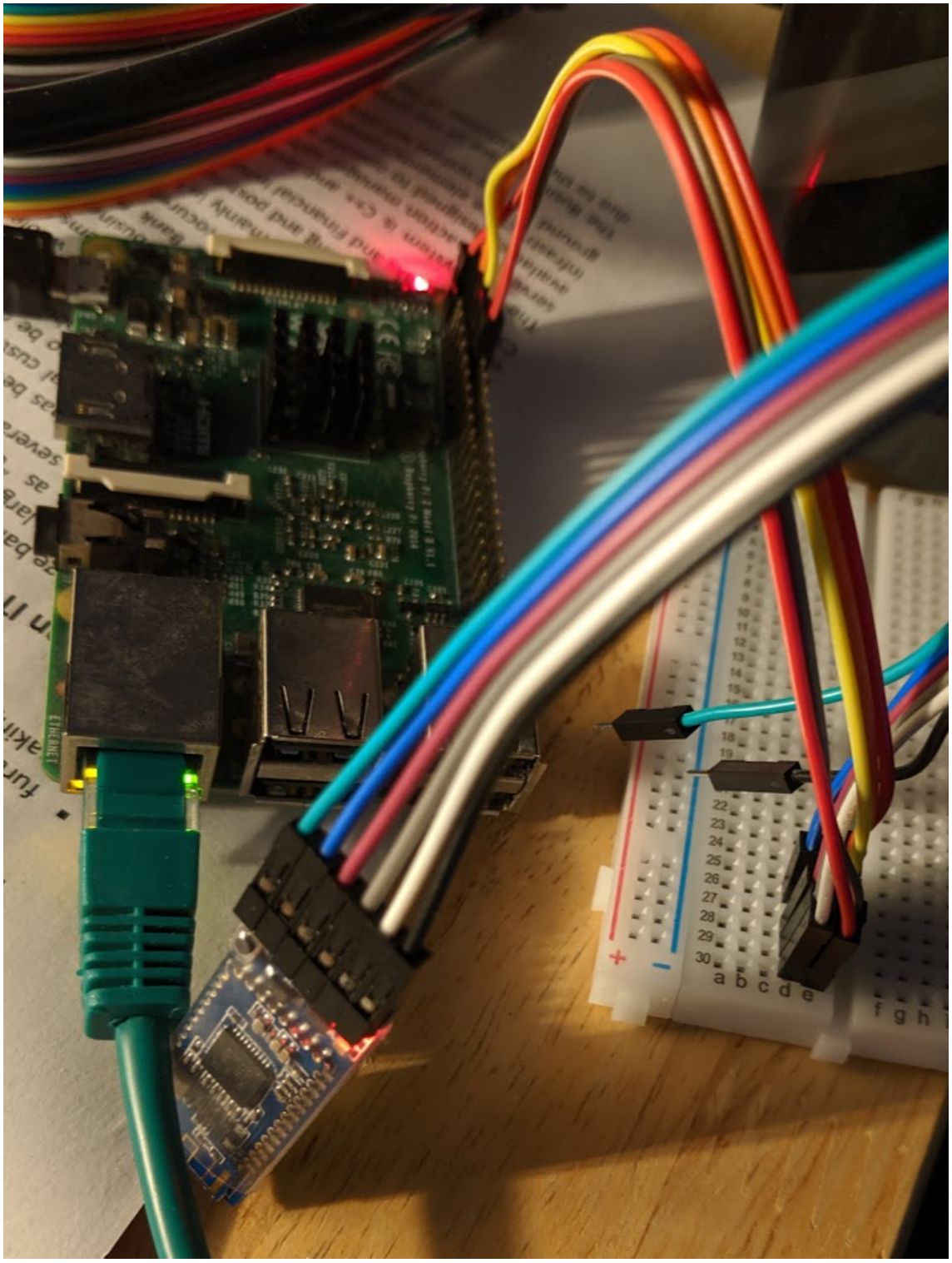


Figure 3: Raspberry Pi 2B with an HM-1o breakout board attached

Uart Bouetooth Module on RPi2 - Raspberry Pi Forums" n.d.) The code is actually very trivial - it opens a serial connection with the correct parameters and then sends data before listening. Because AT commands use a 100ms timeout it is not necessary to run two threads to send and receive data - after sending a command it is simply necessary to wait for any data on the serial buffer for 100ms before continuing. I modified the script to:

- Send the long string (above 80 characters)
- Send the `AT` command and listen for a response
- Send the `AT+ADDR?` command and listen for a response

The script is provided in Appendix 3.

Running this script several times with different RX/TX pin configurations and baud rates did not yield any results and the only success was some null bytes, as in Figure 4.

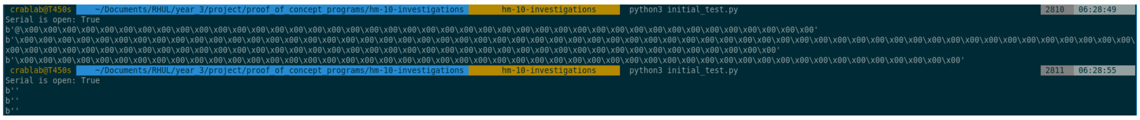


Figure 4: Run one with null bytes, second run with nothing. The change between runs is swapping the TX/RX pins on the UART.

Reading around the UART protocol, it transpired that flow control is sometimes required for devices. (Mark Duncombe n.d.) Flow control is used to indicate to the other party when it is 'safe' to send data and when it is not. As UART is so ubiquitous it is likely there will be cases where the buffer size and the processing capabilities between devices is vastly different. To prevent more capable devices flooding low powered devices (either leading to lost data or exceptions) the crosscoupled `state` (RTS) and `enable` (CTS) supplement the data lines and when a device needs to inhibit further data it will bring the `state` line to Logic 0 (`enable` on the partner device). (Silicon Labs 2017)

The Raspberry Pi does support hardware flow control, however enabling it is non-trivial and it was not possible to enable it in Minicom, after looking into several methods of enabling it. (Vince Deater n.d.)

Upon speaking with Dave Cohen and investigating further it was suggested that as the RX pin of the HM-10 was technically 3.3V, providing a 5V signal could cause issues. A potential divider was tried but this also failed to resolve the issues. (Martyn Currey 2017) It also appeared that later versions of firmware used the higher 15200 baud rate - this was also tested.

It was also observed that using a BLE Scanner Android application (Figure 5), it was possible to connect to the HM-10 and read out the GATT data - no acknowledgement of the connection was observed on the serial port as expected and documented. This suggested the device was, in part, functioning as expected.

Up until this point, two HM-10 devices from the same source had been used. Due to the issues encountered another was procured from an alternative supplier. Upon swapping it like for like with the other HM-10, it worked first time with the script provided in Appendix 3. This does validate that this was a hardware issue rather than "user error" however the exact issue is unknown.

Using a BLE Terminal app it was then possible to send data via the HM-10 to a running screen session as in Figures 8 and 9:

Focus is then on integrating this with the Web Bluetooth APIs to transmit data from a browser down to the Bluetooth device.

The support for the Web Bluetooth APIs is rather more limited than I had believed and only Chromium and Opera offer any support for the APIs. (Mozilla n.d.) The officially updated development progress can be found here.

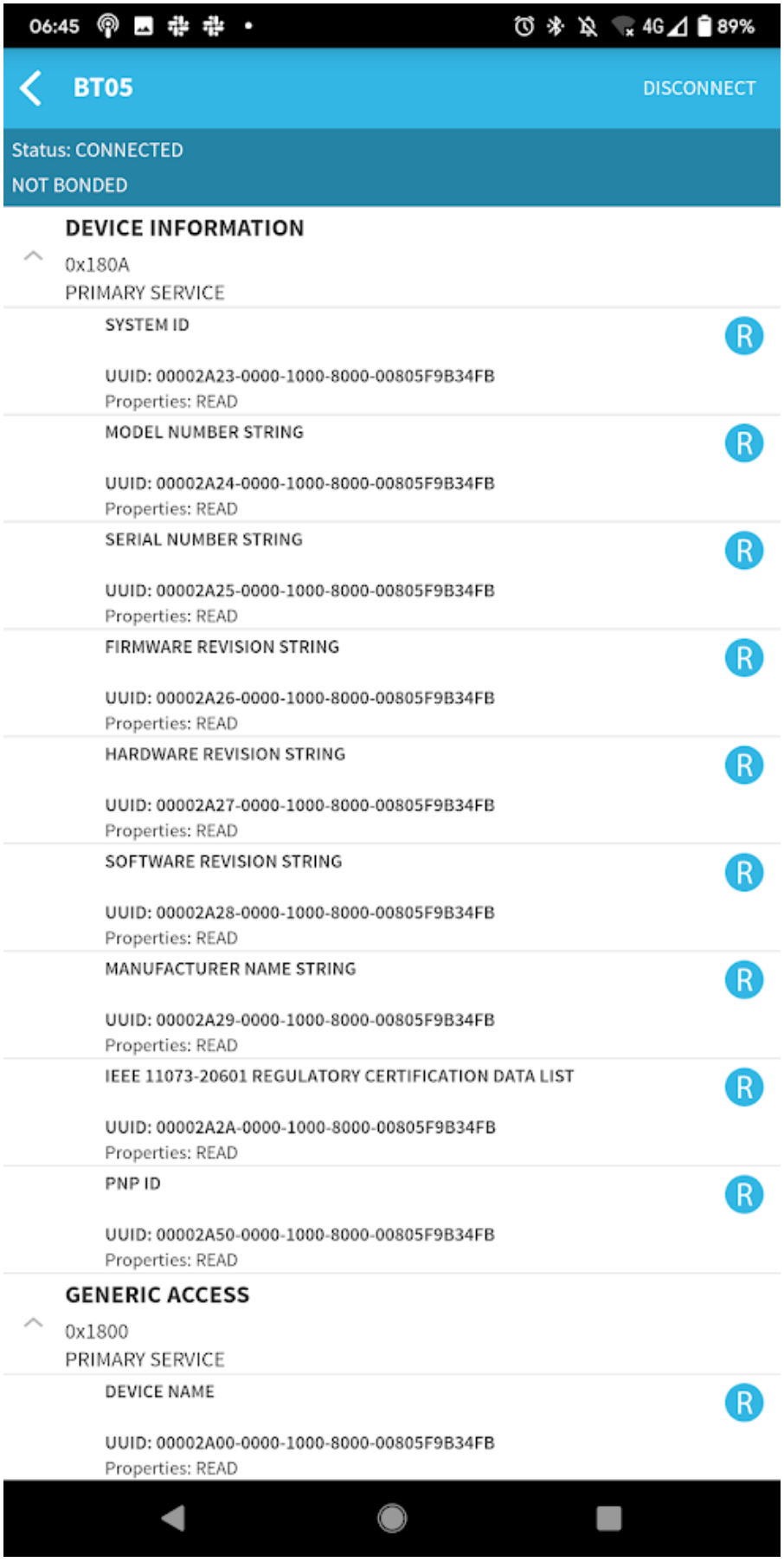


Figure 5: Using the BLE Scanner Android application to connect to the HM-10

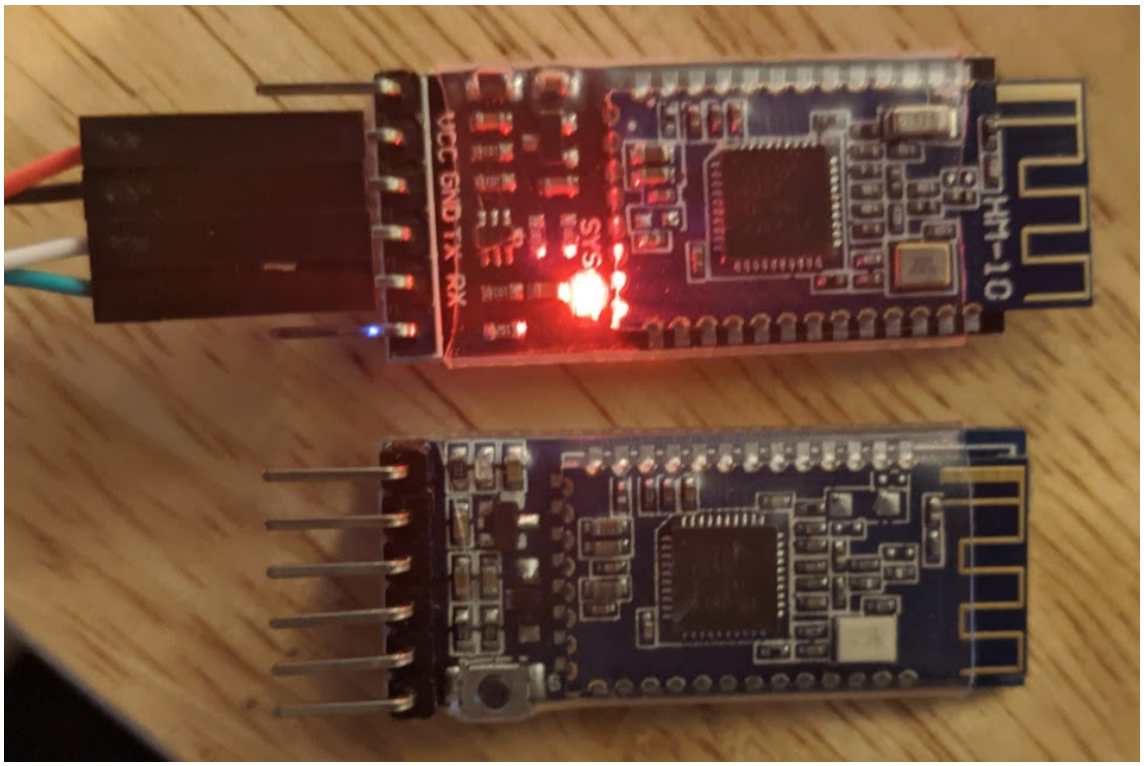


Figure 6: The two HM-10s side by side. The new (and working) device has the LED illuminated

```
crablab@T450s ~/Documents/RHUL/year 3/project/proof_of_concept_programs/hm-10-investigations hm-10-investig
ations python initial_test.py 3144 02:34:26
Serial is open: True
b''
b'OK+LOSTOK+CONN'
b''
```

Figure 7: The expected output of the script in Appendix 3 with a working HM-10

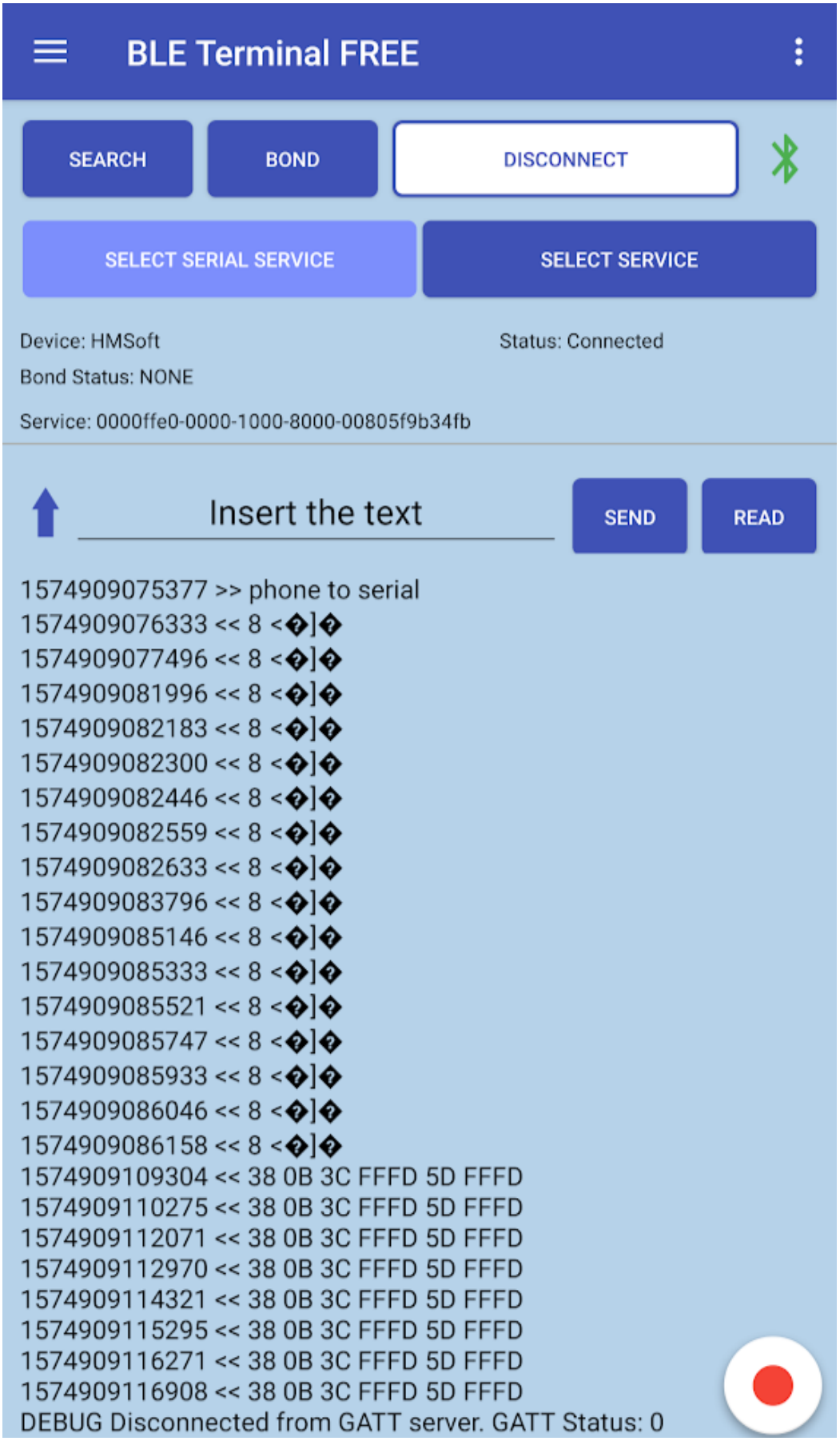


Figure 8: The BLE Terminal Android application connected to the HM-10 and transmitting and receiving text.

Figure 9: The equivilant end with a screen session attached, showing the text and the AT messages when GATT changes take place.

Based on a development guide from Google (written by a member of the Web Bluetooth Community Group) (Beaufort 2015) a very basic solution was developed (Appendix 4) and there are several items to note.

The Web Bluetooth API requires user interaction on the page before scanning can even take place, and in any event the user needs to explicitly provide consent to pair with a device via a prompt. In the solution, an `onClick` event from a button is used to call the function. Once called the function scans devices and looks for an explicit unique identifier for the specific HM-10 in use (transcribed from the BLE Scanner app): `0x484D536F6674`. It is possible to apply filters based on GATT characteristics however for simplicity this is avoided. The name of the device should then be logged, and the pairing sequence initiated (requiring user confirmation).

Web Bluetooth is so experimental that it requires enabling a Chromium feature flag ("Web Bluetooth API - Chrome Platform Status" n.d.) and on Linux, enabling an experimental flag for Bluez (acassis 2016).

At the moment issues with Chromium detecting available Bluetooth cards via Bluez persist (Figure 10) and it has not been able to properly test the solution. Trials with other operating systems may be necassry to have a working Proof of Concept.

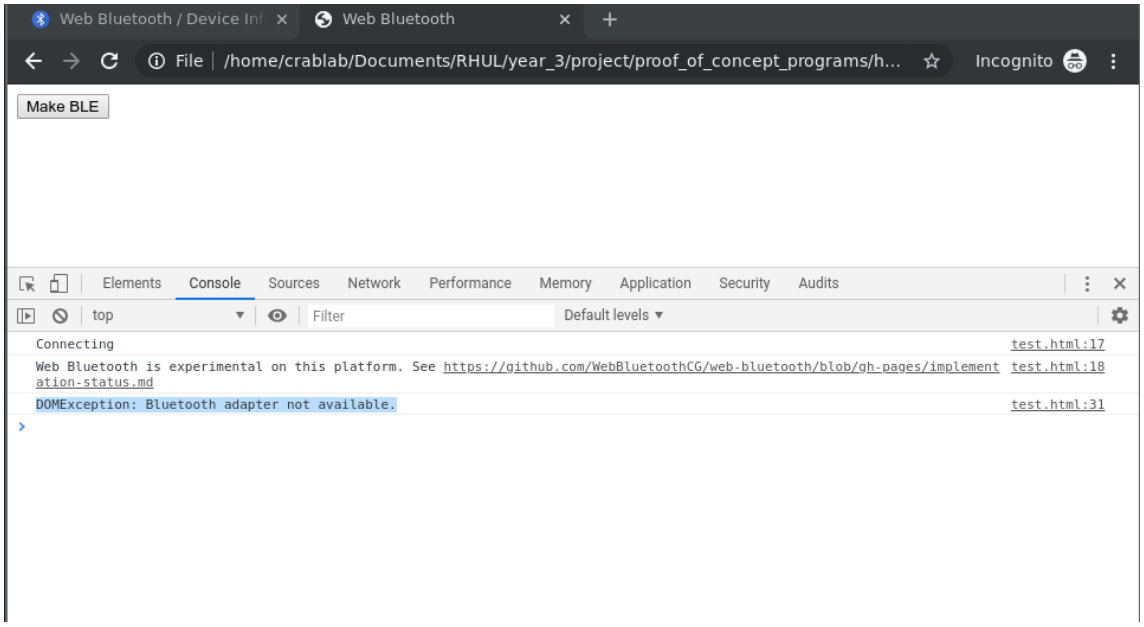


Figure 10: Web Bluetooth API issues with Linux Bluetooth drivers

Acknowledgements

Thanks to Tom Pollard et al. (2016) for the front cover template which I have adapted, Marco Torchiano (2015) for the Pandoc table preamble and Cohen and Matos (2013) for the Final Year Project guide and suggested layouts.

Bibiography

acassis. 2016. “How to Get Chrome Web Bluetooth Working on Linux.” *Alan C. Assis*. https://acassis.wordpress.com/2016/06/28/how-to-get-chrome-web-bluetooth-working-on-linux/.

Beaufort, François. 2015. “Interact with Bluetooth Devices on the Web.” *Google Developers*. https://developers.google.com/web/updates/2015/07/interact-with-ble-devices-on-the-web.

Bluetooth SIG. 2010. *Specification of the Bluetooth System*. 4.0 ed. https://www.bluetooth.com/specifications/archived-specifications/.

———. 2013. *Specification of the Bluetooth System*. 4.1 ed. https://www.bluetooth.com/specifications/archived-specifications/.

“C21 Clicker.” n.d. Accessed October 30, 2019. https://www.c21technoventures.com/clicker.php.

Cohen, Dave, and Carlos Matos. 2013. “Third Year Projects – Rules and Guidelines.” Royal Holloway, University of London.

Daemen, Joan. 1999. “The Rijndael Block Cipher,” September, 45.

Fischer, Jeffrey. 2009. “NFC in Cell Phones: The New Paradigm for an Interactive World [Near-Field Communications].” *IEEE Communications Magazine* 47 (6): 22–28. https://doi.org/10.1109/MCOM.2009.5116794.

gbaman. n.d. “Simple Guide for Setting up OTG Modes on the Raspberry Pi Zero, the Fast Way!” *Gist*. Accessed November 28, 2019. https://gist.github.com/gbaman/975e2db164b3ca2b51ae11e45e8fd40a.

Gomez, Carles, Joaquim Oller, and Josep Paradells. 2012. “Overview and Evaluation of Bluetooth Low Energy: An Emerging Low-Power Wireless Technology.” *Sensors* 12 (9): 11734–53. https://doi.org/10.3390/s120911734.

Heydon, Robin. 2013. *Bluetooth Low Energy the Developer’s Handbook*. Upper Saddle River, N.J.: Prentice Hall.

“How to Communicate with Serial Uart Bouetooth Module on RPi2 - Raspberry Pi Forums.” n.d. Accessed November 4, 2019. https://www.raspberrypi.org/forums/viewtopic.php?f=91&t=158856&p=1032763#p1032763.

Hsiang-Te Ho, Hsing-Ya Chiang, and Sheau-Chien Wang. n.d. US6480553.pdf. US6480553. Accessed November 27, 2019. https://patentimages.storage.googleapis.com/6b/98/35/f35acbc4cac38e/US6480553.pdf.

JNHuaMao Technology Company. 2014. “HM-10-Datasheet.pdf.” https://xdripkit.co.uk/HM-10-datasheet.pdf.

Klostermeier, Gerhard. 2019. “Ikarus23/MifareClassicTool.” https://github.com/ikarus23/MifareClassicTool.

Loginov, Danila. 2019. “How to Make a Web App for Your Own Bluetooth Low Energy Device?” *Medium*. https://medium.com/@loginov_rocks/how-to-make-a-web-app-for-your-own-bluetooth-low-energy-device-arduino-2af8d16fdbe8.

Marco Torchiano. 2015. “How to Solve Longtable Is Not in 1-Column Mode Error?” *TeX - LaTeX Stack Exchange*. https://tex.stackexchange.com/questions/161431/how-to-solve-longtable-is-not-in-1-column-mode-error.

Mark Duncombe. n.d. “Is UART Flow Control (RTS/CTS) Necessary for Proper Operation in Wireless Modules?” *Laird Connect*. Accessed November 5, 2019. https://www.lairdconnect.com/resources/blog/uart-flow-control-rtscts-necessary-proper-operation-wireless-modules.

Martyn Currey. 2017. “HM-10 Bluetooth 4 BLE Modules.” https://webcache.googleusercontent.com/search?q=cache:3SL-aKhV5CkJ:www.martyncurrey.com/hm-10-bluetooth-4ble-modules/+&cd=1&hl=en&ct=clnk&gl=uk&client=ubuntu.

Mayes, Keith E., and Carlos Cid. 2010. "The MIFARE Classic Story." *Information Security Technical Report*, Protocols and Cryptography, 15 (1): 8–12. https://doi.org/10.1016/j.istr.2010.10.009.

Milosevich, Milena. 2006. "AT Commands Reference Guide," August, 614. https://www.sparkfun.com/datasheets/Cellular%20Modules/AT_Commands_Reference_Guide_r0.pdf.

Mozilla. n.d. "Web Bluetooth API." *MDN Web Docs*. Accessed October 28, 2019. https://developer.mozilla.org/en-US/docs/Web/API/Web_Bluetooth_API.

National, and Institute of Standards and Technology. 2001. "FIPS 197, Advanced Encryption Standard (AES)." *Processing Standards Publication*, no. 197 (November): 51.

"Nfc-Tools/Mfcuk." 2019. nfc-tools. https://github.com/nfc-tools/mfcuk.

Nikitin, P.V., and K.V.S. Rao. 2006. "Performance Limitations of Passive UHF RFID Systems." In *2006 IEEE Antennas and Propagation Society International Symposium*, 1011–4. https://doi.org/10.1109/APS.2006.1710704.

Nohl, Karsten, David Evans, and Henryk Plotz. n.d. "Reverse-Engineering a Cryptographic RFID Tag," 9.

Nohl, Karsten, and Henryk Plötz. n.d. "Mifare." Accessed October 31, 2019. /v/24c3-2378-en-mifare_security.

Silicon Labs. 2017. "AN0059.0: UART Flow Control." https://www.silabs.com/documents/public/application-notes/an0059.0-uart-flow-control.pdf.

"The Pi4J Project – Pin Numbering - Raspberry Pi 2 Model B." n.d. Accessed November 5, 2019. https://pi4j.com/1.2/pins/model-2b-rev1.html.

"The Raspberry Pi UARTs - Raspberry Pi Documentation." n.d. Accessed November 28, 2019. https://www.raspberrypi.org/documentation/configuration/uart.md.

Tom Pollard, Marvin Reimer, David San, Arco Mul, Matthew Gwynfryn Thomas, Jakub Nowosad, Dennis Weissmann, and W. Caleb McDaniel. 2016. "Template for Writing a PhD Thesis in Markdown." Zenodo. https://doi.org/10.5281/zenodo.58490.

Townsend, Kevin, Carles Cufí, Akiba, and Robert Davidson. 2014. *Getting Started with Bluetooth Low Energy: Tools and Techniques for Low-Power Networking*. O'Reilly.

Vince Deater. n.d. "Raspberry Pi Hardware Flow Control." Accessed November 5, 2019. https://www.deater.net/weave/vmwprod/hardware/pi-rts/.

"Web Bluetooth API - Chrome Platform Status." n.d. Accessed November 28, 2019. https://www.chromestatus.com/feature/5264933985976320.

WebBluetoothCG. 2019. "Web Bluetooth." https://webbluetoothcg.github.io/web-bluetooth/.

Appendix

Appendix 1: student ID full tag output

```
** TagInfo scan (version 4.24.5) 2019-11-27 17:44:39 **
Report Type: External

-- IC INFO --------------------------------

# IC manufacturer:
NXP Semiconductors

# IC type:
MIFARE Classic EV1 (MF1S50)

-- NDEF --------------------------------

# No NDEF data storage present:
Maximum NDEF storage size after format: 716 bytes

-- EXTRA --------------------------------

# Memory size:
1 kB
* 16 sectors, with 4 blocks per sector
* 64 blocks, with 16 bytes per block

# IC detailed information:
Full product name: MF1S507xX/V1

# Originality check:
Signature verified with NXP public key

-- FULL SCAN --------------------------------

# Technologies supported:
MIFARE Classic compatible
ISO/IEC 14443-3 (Type A) compatible
ISO/IEC 14443-2 (Type A) compatible

# Android technology information:
Tag description:
* TAG: Tech [android.nfc.tech.NfcA, android.nfc.tech.MifareClassic, android.nfc.tech.NdefFormatabl
* Maximum transceive length: 253 bytes
* Default maximum transceive time-out: 618 ms

# Detailed protocol information:
ID: E2:F8:0C:77
ATQA: 0x0400
SAK: 0x08

# Memory content:
Sector 0 (0x00)
[00] r--  E2 F8 0C 77 61 88 04 00 C8 23 00 20 00 00 00 18 |...wa....#. ....|
[01] rwi  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |................|
[02] rwi  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |................|
```

```
[03] wxx  FF:FF:FF:FF:FF:FF FF:07:80 69 FF:FF:FF:FF:FF:FF
          Factory default key           Factory default key (readable)

Sector 1 (0x01)
[04] rwi  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |................|
[05] rwi  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |................|
[06] rwi  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |................|
[07] wxx  FF:FF:FF:FF:FF:FF FF:07:80 69 FF:FF:FF:FF:FF:FF
          Factory default key           Factory default key (readable)

Sector 2 (0x02)
[08] rwi  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |................|
[09] rwi  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |................|
[0A] rwi  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |................|
[0B] wxx  FF:FF:FF:FF:FF:FF FF:07:80 69 FF:FF:FF:FF:FF:FF
          Factory default key           Factory default key (readable)

Sector 3 (0x03)
[0C] rwi  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |................|
[0D] rwi  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |................|
[0E] rwi  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |................|
[0F] wxx  FF:FF:FF:FF:FF:FF FF:07:80 69 FF:FF:FF:FF:FF:FF
          Factory default key           Factory default key (readable)

Sector 4 (0x04)
[10] rwi  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |................|
[11] rwi  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |................|
[12] rwi  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |................|
[13] wxx  FF:FF:FF:FF:FF:FF FF:07:80 69 FF:FF:FF:FF:FF:FF
          Factory default key           Factory default key (readable)

Sector 5 (0x05)
[14] rwi  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |................|
[15] rwi  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |................|
[16] rwi  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |................|
[17] wxx  FF:FF:FF:FF:FF:FF FF:07:80 69 FF:FF:FF:FF:FF:FF
          Factory default key           Factory default key (readable)

Sector 6 (0x06)
[18] ???  -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
[19] ???  -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
[1A] ???  -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
[1B] ???  XX:XX:XX:XX:XX:XX --:--:-- -- XX:XX:XX:XX:XX:XX
          (unknown key)                 (unknown key)

Sector 7 (0x07)
[1C] rwi  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |................|
[1D] rwi  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |................|
[1E] rwi  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |................|
[1F] wxx  FF:FF:FF:FF:FF:FF FF:07:80 69 FF:FF:FF:FF:FF:FF
          Factory default key           Factory default key (readable)

Sector 8 (0x08)
[20] rwi  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |................|
[21] rwi  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |................|
[22] rwi  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |................|
```

```
[23] wxx  FF:FF:FF:FF:FF:FF FF:07:80 69 FF:FF:FF:FF:FF:FF
          Factory default key           Factory default key (readable)

Sector 9 (0x09)
[24] rwi  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |................|
[25] rwi  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |................|
[26] rwi  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |................|
[27] wxx  FF:FF:FF:FF:FF:FF FF:07:80 69 FF:FF:FF:FF:FF:FF
          Factory default key           Factory default key (readable)

Sector 10 (0x0A)
[28] rwi  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |................|
[29] rwi  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |................|
[2A] rwi  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |................|
[2B] wxx  FF:FF:FF:FF:FF:FF FF:07:80 69 FF:FF:FF:FF:FF:FF
          Factory default key           Factory default key (readable)

Sector 11 (0x0B)
[2C] rwi  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |................|
[2D] rwi  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |................|
[2E] rwi  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |................|
[2F] wxx  FF:FF:FF:FF:FF:FF FF:07:80 69 FF:FF:FF:FF:FF:FF
          Factory default key           Factory default key (readable)

Sector 12 (0x0C)
[30] rwi  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |................|
[31] rwi  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |................|
[32] rwi  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |................|
[33] wxx  FF:FF:FF:FF:FF:FF FF:07:80 69 FF:FF:FF:FF:FF:FF
          Factory default key           Factory default key (readable)

Sector 13 (0x0D)
[34] rwi  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |................|
[35] rwi  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |................|
[36] rwi  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |................|
[37] wxx  FF:FF:FF:FF:FF:FF FF:07:80 69 FF:FF:FF:FF:FF:FF
          Factory default key           Factory default key (readable)

Sector 14 (0x0E)
[38] rwi  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |................|
[39] rwi  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |................|
[3A] rwi  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |................|
[3B] wxx  FF:FF:FF:FF:FF:FF FF:07:80 69 FF:FF:FF:FF:FF:FF
          Factory default key           Factory default key (readable)

Sector 15 (0x0F)
[3C] rwi  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |................|
[3D] rwi  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |................|
[3E] rwi  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |................|
[3F] wxx  FF:FF:FF:FF:FF:FF FF:07:80 69 FF:FF:FF:FF:FF:FF
          Factory default key           Factory default key (readable)

r/R=read, w/W=write, i/I=increment,
d=decr/transfer/restore, x=r+w, X=R+W
data block: r/w/i/d:key A|B, R/W/I:key B only,
  I/i implies d, *=value block
```

```
trailer (order: key A, AC, key B): r/w:key A,
  W:key B, R:key A|B, (r)=readable key
AC: W implies R+r, R implies r

--------------------------------------
```

Appendix 2: Minicom output

```
--- Miniterm on /dev/ttyAMA0  9600,8,N,1 ---
--- Quit: Ctrl+] | Menu: Ctrl+T | Help: Ctrl+T followed by Ctrl+H ---

--- pySerial (3.4) - miniterm - help
---
--- Ctrl+]   Exit program
--- Ctrl+T   Menu escape key, followed by:
--- Menu keys:
---    Ctrl+T  Send the menu character itself to remote
---    Ctrl+]  Send the exit character itself to remote
---    Ctrl+I  Show info
---    Ctrl+U  Upload file (prompt will be shown)
---    Ctrl+A  encoding
---    Ctrl+F  edit filters
--- Toggles:
---    Ctrl+R  RTS   Ctrl+D  DTR   Ctrl+B  BREAK
---    Ctrl+E  echo  Ctrl+L  EOL
---
--- Port settings (Ctrl+T followed by the following):
---    p          change port
---    7 8        set data bits
---    N E O S M  change parity (None, Even, Odd, Space, Mark)
---    1 2 3      set stop bits (1, 2, 1.5)
---    b          change baud rate
---    x X        disable/enable software flow control
---    r R        disable/enable hardware flow control

--- Settings: /dev/ttyAMA0  9600,8,N,1
--- RTS: active    DTR: active    BREAK: inactive
--- CTS: active    DSR: inactive  RI: inactive  CD: inactive
--- software flow control: inactive
--- hardware flow control: inactive
--- serial input encoding: UTF-8
--- serial output encoding: UTF-8
--- EOL: CRLF
--- filters: default
--- local echo active ---
AT
ABCABCABCABCABCABCABCABCABCABCABCABCABCABCABCABCABCABCABCABCABCABCABCABCABCABCABCABCABCABCABC

--- Settings: /dev/ttyAMA0  9600,8,N,1
--- RTS: active    DTR: active    BREAK: inactive
--- CTS: active    DSR: inactive  RI: inactive  CD: inactive
--- software flow control: inactive
--- hardware flow control: inactive
--- serial input encoding: UTF-8
--- serial output encoding: UTF-8
--- EOL: CRLF
```

```
--- filters: default

--- Settings: /dev/ttyAMA0  9600,8,N,1
--- RTS: active    DTR: active    BREAK: inactive
--- CTS: active    DSR: inactive  RI: inactive  CD: inactive
--- software flow control: inactive
--- hardware flow control: inactive
--- serial input encoding: UTF-8
--- serial output encoding: UTF-8
--- EOL: CRLF
--- filters: default
```

Appendix 3: HM-10 Test Script

```
import serial, time

ser = serial.Serial(
        port='/dev/ttyUSB0',
        baudrate=9600,
        parity=serial.PARITY_NONE,
        stopbits=serial.STOPBITS_ONE,
        bytesize=serial.EIGHTBITS,
        timeout=1
)

print("Serial is open: " + str(ser.isOpen()))

ser.write(b'ABCABCABCABCABCABCABCABCABCABCABCABCABCABCABCABCABCABCABCABCABCABCABCABCABCABCABCABCABCABCAB
print(str(ser.read(100)))

time.sleep(2)

ser.write(b'AT')
print(str(ser.read(100)))

ser.write(b'AT+ADDR?')
print(str(ser.read(100)))

ser.close()
```

Appendix 4: Web Bluetooth API test

```
<html>

<head>
    <title>Web Bluetooth</title>
</head>

<body>
    <button id="button" onclick="makeBLE()" type="button">Make BLE</button>
</body>

<script>
    function makeBLE() {
        console.log("Connecting");
        navigator.bluetooth.requestDevice({
```

```
                filters: [{
                    services: [0x484D536F6674]
                }]
            })
            .then(device => {
                // Human-readable name of the device.
                console.log(device.name);

                // Attempts to connect to remote GATT Server.
                return device.gatt.connect();
            })
            .catch(error => {
                console.log(error);
            });
    };
</script>

</html>
```

A proposed technical solution to attendance monitoring at Royal Holloway

Hugh Wells - 864564

20th November 2019

Abstract

The current systems at Royal Holloway to monitor student attendance at lectures and other mandatory events have been discussed at length previously. This report aims to propose a new solution using modern technology which is more secure, trustworthy and less of an administrative burden. Functional requirements will be discussed and the overall solution including technical choices explained. Additional reports investigating technical details and justifying those in more detail will be available.

Problem Statement

Royal Holloway keeps track of attendance in lectures both to ensure that students are regularly attending and also to satisfy legal requirements regarding the visas of overseas students (Home Office (UK Government) 2019). It is essential that this data is gathered and analysed efficiently and accurately.

In the past attendance has been tracked using signatures on registers. More recently, due to the lack of scalability of this former approach, a system of clickers has been employed (Royal Holloway Department of Computer Science 2018). This latter system is non-optimal and insecurities have been discussed in another report.

In the context of a lecture there are several key problems faced:

- **Circulation of a register**: In classes that range from 50 to 200 students it is infeasible to expect a paper register to have made a complete circuit of the room by the end of a class. Adding another register doesn't always solve this, as the paths of the two registers will often intersect and students are less interested in the optimal pathing of the register than the lecture content. There is often a "scrum" at the end of a lecture between those who did not sign to register their attendance and leave in time for their next class.
- **Data processing**: When discussing paper registers, it is obvious that significant administrative effort is required to process the written records and translate them to a digital medium for easy statistical analysis. This becomes more complex when there are multiple registers for a class, or when changes to the document in question are made. For instance, sometimes students have not been included on the sheet due to an oversight and they will then append their name to the end of the document to register their attendance in a non-standard fashion. Whilst with the digital clickers there is no need more manual transcription, the data from the Turning Point software is in a proprietary format and requires conversion to a CSV and then further processing to be handled by the custom software used in the department for attendance monitoring.
- **Forgery**: Both the clicker and paper register systems are vulnerable to attack. The paper register can simply be signed by another person on behalf of an absentee. The way to prevent this is to carry out signature checking against a sample on record and random ID checks at lectures where forgery is suspected. The signature comparison is labour intensive and

only catches bad forgeries. Student signatures also change over time and can be "gamed" to provide trivial samples which are easily forged. The attack on the clicker system has been described in more detail in another report.

- **Intrusiveness**: The paper register is relatively intrusive in a lecture, however more so to the students than to the lecturer. The clicker system, anecdotally, tends to cause more issues as the set up process is non-trivial cutting into lecture time and lecturer patience. There is no real feedback for a student that their attendance has indeed been counted in the same way as there is on a paper register. The clicker system also requires you to remember a device that must be in your possession to verify your identity - a device is easy to forget, a signature less so.

User Stories

A number of user stories have been written to illustrate the user journey and clarify the key focus of a solution. These are included in Appendix 1.

General Description

The proposed solution is to have a Bluetooth Lite device that a lecturer will configure via a website prior to the lecture, and will be plugged in to a power source for the duration of the lecture in the lecture theatre. No networking or connection to a PC will be required for this device. Students will log in to a website and request to register for the lecture. The website will, using the Bluetooth API, connect to the Bluetooth device in the lecture and will request that the device signs a some data signed and provided by the server, as a challenge. This will provide proof that the student was in, at least the vicinity, of the lecture.

Issues with the solution

- There is nothing to stop someone signing in and then simply walking out; or indeed standing just inside the range of Bluetooth Light but outside the lecture theatre. This is not a risk mitigated by either the register or the clickers. It could be possible to have the student device poll for signatures throughout the lecture, however this would potentially have technical constraints around student device battery life, the capacity of the Bluetooth signing device and running background processes in a web browser. Ultimately, Bluetooth Light is not designed for these kind of constant connections so Bluetooth Classic (which is more difficult to implement) would be a more appropriate communication medium. An alternative would be to randomly poll devices, however the Web Bluetooth implementation requires the user to knowingly acknowledge a connection each time (WebBluetoothCG 2019) and the proposed implementation is designed to accommodate that, so this potential solution is beyond the scope of this project.
- It is conceivable that students might attempt a relay attack by generating the data to be signed remotely and then passing it to another student to transmit it for signing, before they submit the signed data back to the server without ever having been near the lecture. There are two ways this could be solved:
  - Firstly, including the student Bluetooth MAC address in the data signed by the server and validating it with the source MAC address of the transmission on the Bluetooth device would allow you to ensure the device that sent the message also generated the message. This then requires the Bluetooth device to carry a certificate used by the server to validate the message it has sent - this isn't a bad idea regardless, although having the Bluetooth device blindly sign messages regardless isn't a huge security concern as a brute force attack is infeasible. Technically, Bluetooth devices are supposed to have a fixed and unique MAC address which will not change - plus a configurable MAC that does. However, device manufacturers have begun to randomize even the supposedly fixed MAC addresses for privacy reasons, so these may not be a reliable identifier. (Kalantar, Mohammadi, and Sadrieh 2018)

  - Secondly, implementing timing based controls to detect the round trip time of a message to ensure it does not exceed a certain value. This is the method used in EMV payment cards (EMVCo 2016), known in the specifications as "Relay Resistance Protocol"). This is vulnerable to attack should a rogue card reader not carry out the timing checks correctly. (Chothia, Boureanu, and Chen, n.d.) Our security model here is different, as we make the assumption that both the server and the Bluetooth device are not compromised and as any modification of the data would be detected (as the signatures would no longer match) it would not be possible for the student device to alter the timestamp to carry out such an attack. This is the solution that will be implemented, subject to time constraints.
- The proposed implementation will use simple username/password authentication and if any user/password management is provided it will be basic. This is not within the scope of the project and it is likely that if this were ever to be used it would be necessary to integrate this with existing user accounts; such as via the LDAP (Lightweight Directory Access Protocol).

Functional Requirements

1. Student
   a) A website for students to access to manage their attendance
   b) An average percentage attendance score for students for their currently ongoing classes
   c) An individual percentage attendance score for students for a given class
   d) Data exports of attendance by class
   e) Capability to initiate registration process for a class to mark a student as having attended
      1. Website to initiate wireless connection with device in lecture theatre
      2. Website to send data to be signed by wireless device, and receive response
      3. Website to record correctly signed responses as attendance
   f) Ability to make absence request for a given time period
      1. Ability to view absence requests and their status
      2. Ability to withdraw an absence request
2. Lecturer
   a) Lecturers to be able to view their classes on a website
   b) Lecturers to be able to see overall percentage attendance for their class
   c) Lecturers to be able to see breakdown of percentage attendance by student per class
   d) Lecturers to be able to see breakdown attendance records by student per class
   e) Lecturers to be able to provision wireless device for an upcoming class
      1. This involves providing a configuration file and certificate to the device
3. Administrator
   a) Administrators to be able to view classes by department on a website
   b) Administrators to be able to view same breakdown of data as lecturers
   c) Administrators to be able to view student overall attendance across all classes
   d) Administrators to be able to view students whose attendance falls below required minimum
   e) Administrators to be able to view students whose days of absence is above a required threshold
   f) Administrators to be able to manage absence requests
      1. Ability to review outstanding requests
      2. Ability to approve or deny requests
      3. Ability to view previous requests per class or per student
   g) Administrators to be able to view browser fingerprinting data

    1. Ability to see when a fingerprint is used across multiple students
    2. Ability to see fingerprint consistency per student
h) Arbitrary data export as CSV for all previous views
    1. This in particular concerns a format the College can process - student ID against a boolean attendance value for each class in a CSV.

Constraints

Design constraints

1) The website will be hosted on a Python Flask application backed by a MySQL database
2) The frontend will be based on the Bootstrap framework
3) The wireless device will be a Bluetooth Low Energy device compatible with the Bluetooth 4.1 specification
    a) This is currently planned to be a HM-10; further details are available in other reports
    b) The processing capability will be provided by a Raspberry Pi
    c) The provisioning of the device will be carried out by copying a file to the SD card of the device

UML Sequence Diagram

Included in Appendix 2

Acknowledgements

Thanks to Tom Pollard et al. (2016) for the front cover template which I have adapted, Marco Torchiano (2015) for the Pandoc table preamble and Cohen and Matos (2013) for the Final Year Project guide and suggested layouts.

Bibiography

Chothia, Tom, Ioana Boureanu, and Liqun Chen. n.d. “Making Contactless EMV Robust Against Rogue Readers Colluding with Relay Attackers,” 10.

Cohen, Dave, and Carlos Matos. 2013. “Third Year Projects – Rules and Guidelines.” Royal Holloway, University of London.

EMVCo. 2016. “EMV Contactless Specifications for Payment Systems,” 589.

Home Office (UK Government). 2019. “Tier4 of the Points Based System: Guidance for Sponsors Document 2: Sponsorship Duties.” https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/824003/Tier_4_Sponsor_Guidance_-_Doc_2_-_Sponsorship_Duties_2019-08_1.1.pdf.

Kalantar, Golnar, Arash Mohammadi, and S. Nima Sadrieh. 2018. “Analyzing the Effect of Bluetooth Low Energy (BLE) with Randomized MAC Addresses in IoT Applications.” In *2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, 27–34. https://doi.org/10.1109/Cybermatics_2018.2018.00039.

Marco Torchiano. 2015. “How to Solve Longtable Is Not in 1-Column Mode Error?” *TeX - LaTeX Stack Exchange*. https://tex.stackexchange.com/questions/161431/how-to-solve-longtable-is-not-in-1-column-mode-error.

Royal Holloway Department of Computer Science. 2018. “DEPARTMENT OF COMPUTER SCIENCE Undergraduatestudent HANDBOOK.” https://intranet.royalholloway.ac.uk/computerscience/documents/pdf/currentug/ug-student-handbook-2018-19.pdf.

Tom Pollard, Marvin Reimer, David San, Arco Mul, Matthew Gwynfryn Thomas, Jakub Nowosad, Dennis Weissmann, and W. Caleb McDaniel. 2016. “Template for Writing a PhD Thesis in Markdown.” Zenodo. https://doi.org/10.5281/zenodo.58490.

WebBluetoothCG. 2019. “Web Bluetooth.” https://webbluetoothcg.github.io/web-bluetooth/.

Appendix

Appendix 1

User Stories

Student

1) "As a student I want to register my attendance so I can prove I was at a lecture"
2) "As a student I want to register my attendance without distraction from the lecture content"
3) "As a student I want to be able to know my attendance has been counted"
4) "As a student I want to be able to see my attendance history"
5) "As a student I want to register with the minimum of additional hassle"
6) "As a student I don't want to have to remember extra items to register"

Lecturer

1) "As a lecturer I want to not have to think about lecture registration"
2) "As a lecturer I want to have the minimum of distractions and the minimum of fuss, in registration"
3) "As a lecturer I don't want to be involved in the registration process"

Administrator

1) "As an administrator I want to have all the registration data in a machine readable format"
2) "As an administrator I want to be able to easily identify students who have not reached minimum attendance levels"
3) "As an administrator I want to easily identify cases of fraud or forgery"
4) "As an administrator I want to generate reports on attendance and export raw data"

Appendix 2

UML Sequence Diagram