

How viable is Software Defined Radio as a wireless sniffing platform?

Rory Bolton

June 5, 2017



In memory of the late
Mr. Aidan Oakley and Mrs. Betty Medhurst
Friend and family, lost while writing.

Contents

1	Introduction	4
2	Literature Review	5
3	Method	8
3.1	Software Design Methodology	8
3.2	Implementation	8
3.2.1	Systems Design	9
3.2.2	SDR Design Factors	10
3.3	Data Collection Algorithm	11
3.3.1	Signal Processing	13
3.3.2	Data Display	16
4	Evaluation/Results	17
5	Discussion	18
6	Reflection on Project Management	19
7	Social, Legal and Ethical Issues	24
8	Feedback From Presentation	24
9	Conclusion	25
10	References	25
11	Appendix	27
11.1	Email from Lime Microsystems	27
11.2	Emails to and from Supervisor	28
11.2.1	30/11/16	28
11.2.2	23/01/17	28
11.2.3	02/02/17	29
11.2.4	09/03/17	29
11.2.5	18/04/17	29
11.2.6	25/04/17	30
11.2.7	Presentation feedback	31
11.3	Detailed Project Proposal Form	32
12	Certificate of Ethical Approval	38

Abstract

This paper shows the design and testing of a mobile protocol identification system using off the shelf Software Defined Radio hardware. SDR is a capable technology that has the capacity to be extremely useful to a security professional thanks to its universal nature. SDR could be used to not only identify protocols in use, but with further work could also be used to collect any data being transmitted on those protocols simultaneously.

1 Introduction

Software Defined Radio (SDR) is an amazingly versatile technology that could have potentially huge ramifications for security due to its rapidly lowering barrier to entry for members of the general public. The security of systems may be at stake thanks to the very reason why SDRs are so versatile. They have an inherent reconfigurability and their wide bandwidth allows them to effectively emulate any other device that broadcasts within SDRs capabilities. An SDR can receive and transmit using any protocol provided its software is instructed how and the hardware is capable thus opening the metaphorical floodgates to broadcasting on restricted frequencies and/or emulating public infrastructure such as GSM and LTE base-stations[13] and video broadcasting. Hardware Defined Radio (HDR) devices are locked to a specific protocol and also have a very limited range of frequencies that can receive or transmit on. SDRs however do not have this same limitation. They have no preference over any protocols or frequency. Anything is possible assuming of course that what is being attempted is within the devices receive and transmit power and bandwidth range. Whereas some HDRs often de-encapsulate the data packets before they exit the silicon, an SDR does no de-encapsulation and communicates every piece of raw data back to the host computer for storage or decoding thus allowing every intricacy of the protocol to be viewed and analysed.

SDRs are extremely flexible in the range of signals that they can receive and interpret. This flexibility allows the same device to simultaneously receive transmissions from devices on multiple channels and (with capable hardware) devices that normally would be completely separate from one-another and transmit on entirely different RF bands.

2 Literature Review

The ability for SDRs to emulate any hardware and transmit using any protocol allows a user to exploit flaws in areas that were not previously accessible by the average non state funded attacker. Thanks to this new technology, attacks are being found in areas that few would ever consider as being at risk, such as the Digital Video Broadcasting - Terrestrial (DVB-T) standard and more specifically, the substandard of Hybrid Broadcast TV (HDTV) that allows a broadcaster to request content from a web page as part of a transmission. This has already been exploited using SDR technology[7] to transmit a signal that makes use of a vulnerability in the televisions in-built web browser to run unsigned code. This is believed to be merely the beginning of what is to come. A surprising many of the currently available IoT devices are vulnerable to simple attacks, with 70% of devices that went under test being found as having no encryption at all[11]. Most IoT exploits are only possible because the manufacturer took a security through obscurity approach to their protection and believed that it was not worth implementing, because nobody would test for it, or nobody would be able to do anything if access was gained. This method of "securing" a product is only functional in the extreme short term and is widely recognised as being ineffective as a method of securing a system "the security of a system should depend on its key, not its design remaining obscure"[1].

SDR has a strong background of military use, first beginning with the DARPA SPEAKEasy project being the first non-prototype design to be widely deployed that has its physical layer components implemented in software[4] for the purposes of ensuring compatibility and integrating existing systems. SDR is currently being used by United States the military to create a range of communications devices known as the Joint Tactical Radio System that was cancelled in 2011 due to being over-ambitious. A small part of the program survived however in the hand-held, manpack and small form radios that are scheduled for full rate production in 2017[17] with the primary goal being to create radios that can have operating frequencies and modes added or updated through a distributed upload while in the field. This such a tool is only possible using software defined radio and functions as an ideal example of using SDR not to exploit security flaws but as a technology that can be used to increase security of a design too.

The first commercial SDR[10] was approved by the American Federal Communications Commission (FCC) in 2004, marking the first time SDR technology became available to anyone without a research fund in the hundreds of millions of dollars and a large development team. The first widely

commercially available single chip RF front end was released in 2009[12] by Lime Microsystems. Since then many more vendors have began offering similar products and as such the price has plummeted while availability has increased. The RF front-end being used for testing in this project is the major successor to that original chip that was released in 2014[5].

The internet consists of billions of connected devices, with that number rapidly expanding as time progresses to the point where 50 billion connected devices are expected by 2020[3] with a rapidly increasing percentage of those devices being a part of the Internet of Things. On many systems; any internal IoT traffic before it reaches the gateway will be un-encrypted[11]. This is often done to conserve power as the gateway is often the most computationally powerful device in the system and therefore more capable of encrypting large quantities of data without any problematic delays. The gateway is usually also the point wherein the internal and external networks meet, external networks may be filtered by the gateway, but as with most routers, the internal traffic is un-monitored and is trusted to not be accessible by an outside attacker. This is not an ideal practice to use, especially for a primarily wireless system where an attacker may not have to be on the premises but can simply place an antenna nearby to receive any signals within range.

SDR technology requires a significantly larger amount of electrical and processing power when compared to a more common hardware defined radio[14] because of its less efficient software defined nature being reliant on a general purpose processor instead of a more optimised Application Specific Integrated Circuit (ASIC) that is designed for the task with efficiency and speed in mind. Although this may affect the hardware's long term portability, it is still possible to perform short term attacks from a nearby dropbox location or laptop for several hours before a battery would need replacing. And although the power consumption may be a limitation, the physical dimensions of SDR hardware is reducing as the technology becomes more integrated meaning that finding a location to hide the device within the premises under attack may not be all too difficult to achieve.

The wireless spectrum is full of different protocols and many of them operate in the 2.4-2.4835GHz band of the EM spectrum. These include Wi-Fi[8], Bluetooth, ZigBee[9] and many proprietary protocols. HDR's are locked to a specific protocol or at most are able to communicate using a select few that reside in the same band because they are constrained by their hardware and are effectively a protocol and frequency locked ASIC. SDRs do not have this limitation of being locked to using a particular protocol or frequency and all that is required to completely switch protocol is a software change which can often be performed in seconds.

When designing a protocol it is common practice to model the system using a number of SDRs to simulate a range of working devices[15]. This enables rapid prototyping of a system before any hardware is obtained and even allows entirely new concepts to be tested if the ASIC hardware does not yet exist.

The choice of SDR hardware is difficult as there are many different products available and they all have their intended use case situated in different areas. This means that no SDR is the same as any other and one may be many times more effective than another.

	HackRF One	LimeSDR	RTL-SDR (R820T2 tuner)
Frequency Range	1MHz - 6GHz[6]	100kHz - 3.8GHz[5]	24MHz - 1.766GHz
ADC Resolution	8 Bit	12 Bit	8 Bit
RF Bandwidth	20 MHz	61.44MHz	3.2MHz
Duplex	Half	Full	Full
FPGA Elements	64 macrocell CPLD	40K	0
Interface	USB 2.0	USB 3.0 or PCI-E	USB 2.0
Price	\$299	\$289	\$20

The project was planned with the hope that the LimeSDR would arrive on time to fully develop the proposed solution. The LimeSDR shows to be the better solution in almost every aspect when compared to the other two products I had available. It has a higher resolution ADC granting a 16x increase in resolution over the other products thanks to its 12 Bit Analogue to Digital Converter (ADC), this also makes it more sensitive to signals meaning that weaker signals can be detected, thus improving the effectiveness of the projects end product.

The only place where the LimeSDR falls short is in its operating range, which is beaten by the HackRF. Although in my intended use-case it is still ideal is I planned to be sniffing protocols that operate within the 2.4GHz spectrum, which is well within the LimeSDR's range.

The LimeSDRs exceptional bandwidth is mainly due to its interface being USB 3.0 which enables much higher data rates than the USB 2.0 interfaces used by the other two solutions. Although the bandwidth used in the product I created is only 2MHz, during further research I would like to replace Soapy_Power with my own solution that directly uses the SoapySDR client API to greatly increase the bandwidth.

The LimeSDR has a 40,000 logic cell FPGA, although this is not used in my project, with further developments this could be used to greatly increase processing speed by utilising it as an ASIC that does a large amount of data processing before it even reaches the host PC.

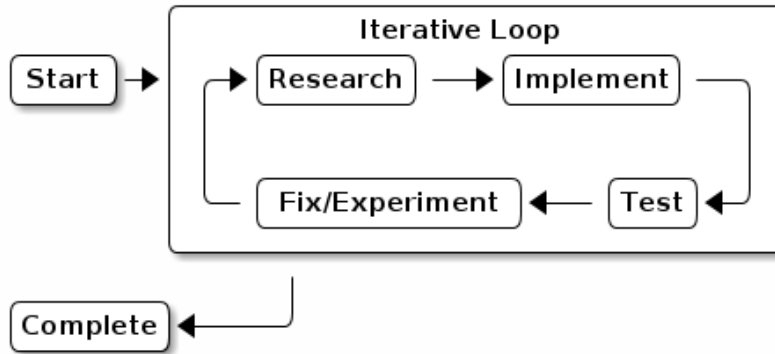
All these points, combined with the lower cost to initially purchase the equipment means that the LimeSDR is clearly the best option from what I had available. The fact the the LimeSDR has all this functionality while maintaining the same or better price as the HackRF (a product it is designed

to directly compete with) is ideal and is an ideal of competition being better for the consumer. And while the support for the device is comparatively low at the time of writing, it is being supported by the manufacturer, who has vast knowledge of SDR technology as they are a leading manufacturer of highly integrated RF Front end ICs for use in SDR equipment, and their current flagship IC is included within the LimeSDR. This means they should be more than capable of ensuring their own product works to its full potential.

3 Method

3.1 Software Design Methodology

The software was designed in an iterative fashion. The required functionality is determined and each necessary function is researched, implemented, tested and then fixed in an iterative cycle before moving on to the next piece of functionality. When an issue cannot be fixed or it may improve the overall outcomes; the code was experimented upon to solve problems I could not overcome with the existing methods.



3.2 Implementation

The primary hardware used for the project is the LimeSDR from Lime Microsystems, a Software Defined Radio that makes use of the LMS7002M fully programmable MIMO transceiver and can reliably cover the entire spectrum of 30MHz to 3GHz. The equipment also has very little clock drift as it contains a Temperature Compensated Crystal Oscillator (TCXO) that helps

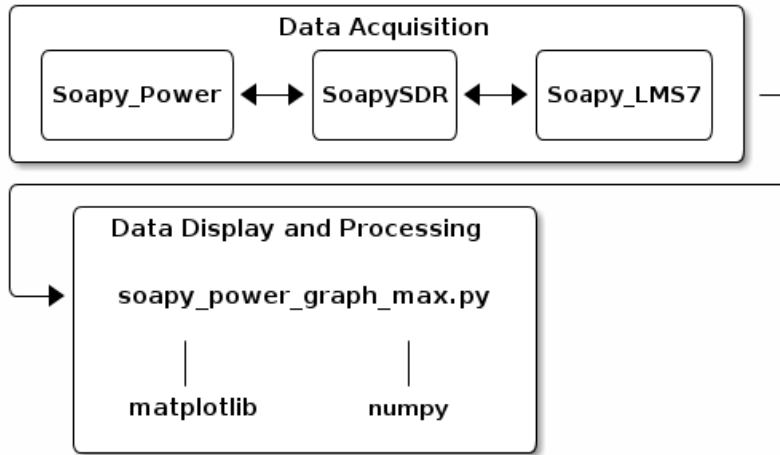
ensure a stable and accurate clock signal for more accurate tuning.

The LimeSDR is controlled and read through the open-source SoapySDR API. SoapySDR was chosen for its relative simplicity, support for many other radios and pre-existing support within many programs used later in the project. The support for other radios should allow the software produced during the project to function not only with the LimeSDR, but also the HackRF, RTL-SDR, all Ettus Research radios and more.

I used python as my language of choice during the development of the project as it is extremely simple and, while it may be slower than compiled languages, it is highly versatile and ideal for a proof of concept implementation where performance may not be a serious concern.

Python also has a few SDR libraries, although my initial plan was to use these libraries, the output they provided proved to be beyond my understanding at the time. With future work these libraries should be used directly instead of the current Soapy_Power implementation which uses these libraries itself but serves as something akin to an abstraction layer.

3.2.1 Systems Design



The system I designed has two functional stages, the data acquisition stage and the data display and processing stage.

The data acquisition stage's task is to receive and store the data in a format that can then be easily interpreted by the data display and processing stage. It does this by reading the SDR through the SoapySDR client

python library, which serves as the abstraction layer and interfaces with the `soapy_LMS7` module that serves as the device specific driver. `Soapy_Power` itself handles converting the received values provided by the library into a list of linear power values that are easily understood and thus can easily be programmed for.

The data display and processing stage stores the 2MHz wide sample received from the loops of the data acquisition stage and performs analysis on the entire specified bandwidth once all the necessary data has been received. The data is displayed in two forms, a plot that shows the peaks detected in the RF spectrum and a statistical readout using the python `curses` library to provide a clean UI showing the protocol, channel and percentage certainty that the reading is accurate.

3.2.2 SDR Design Factors

The LimeSDR has a choice of 3 different antenna connections for each of the 2 RX channels. Currently the `SoapySDR` driver only has support for receiving on channel 0. The antenna connections are as follows:

Input	Use	Noise Figure
LNAL	Low Frequency, tuned for use between 0.1MHz and 2GHz	<2dB
LNAH	High Frequency, tuned for 1.5GHz to 3.8GHz	<3dB
LNAW	Wide-band, non-specific tuning 0.1MHz to 3.8GHz	5-7dB

During testing I have been scanning the range between 2.4GHz and 2.5GHz, so to reduce noise as much as possible I have set the `Soapy_Power` to use the "LNAH" antenna. I could use "LNAW" instead, however the noise figure on this input is noticeably higher, especially since dBm is a logarithmic scale. There is one major use case where the LNAW input may be useful, that is when scanning a wide range of frequencies that overruns the range of a single one of the other inputs in one continuous sweep. This technique would not be able to detect signals as weak as using the other antennas since any weak signals are likely to be buried in the noise floor but would provide a rough outline of what is present in the immediate area.

The physical antennas connected to the ports on the SDR are shown and described below



Figure 1: Image of the actual hardware used for the final implementation.

Input	Antenna	Frequency of antenna
LNAL	820mm Telescopic	Low frequency, variable length, <1GHz
LNAH	ALFA Network Directional	2.4-2.5GHz and 5GHz
LNAW	Generic GSM	800-900MHz

All these antennas were chosen for their sensitivity to commonly used frequencies. Many protocols function at the 433MHz, 800-900MHz and 2.4GHz areas of the spectrum and it is only appropriate to use antennas suited to receiving these signals.

3.3 Data Collection Algorithm

Soapy_Power is an existing program that reads the data from the SDR hardware and prints the power values found at a specific frequency to stdout in the format:

Soapy_Power can automatically sweep over a specified frequency range and continue to output data. This is done by adding the "-c" flag that instructs Soapy_Power to continue running after completing its task whereas without this flag it will exit upon completion. Soapy_Power can sweep a range by supplying the frequency range in the format "startfrequency:endfrequency". The program will then proceed to sweep over this range in 2MHz blocks until it reaches the end, at which point it will either

cleanly exit, or if the "-c" argument is given it will loop back to the starting frequency and begin the loop again.

This is a limitation of the current implementation, the bandwidth is permanently set at 2MHz, there is an option that can be supplied to Soapy_Power to increase the bandwidth, however it does absolutely nothing and even supplying the "--force-bandwidth" flag has no effect. This is a bug with Soapy_Power and it shows no signs of being fixed any time soon. This bug would not exist in any further work as one of the first tasks that needs to be carried out is the replacement of Soapy_Power with a custom implementation that directly interfaces with the SoapySDR libraries.

The data on stdout is piped into the python program. The python program then confirms that the string starts with today's date, indicating that the string is a data message and not a debug message. If that check passes, the comma separated string is split into an array, the current frequency is stored in one array while the 512 power values are retrieved and stored in another. At this point if a debug or error message was printed at the same time as data was, then it may have passed the date test and would cause issues later during processing. In an attempt to remedy this I check every power value to ensure that they are a float, if a single value is not a floating point number, all 512 points of data are discarded. Past this point the data should be valid. The data is added into a larger array while Soapy_Power tunes to the next frequency. Once the new data is received and verified it is added on to the end of the larger array. This array will continue to store all data until the end frequency of any new data is smaller than the previous value. This indicates that Soapy_Power has fully looped and the large array now contains a full sweep of the specified frequencies. To try and differentiate actual signals from noise the overall average power value is found, any power values that sit above that average are flagged for analysis.

The full array of all power values is analysed at the end of every loop of Soapy_Power. The technique I chose to use to identify protocols is to find the number of above-average values that sit within the expected frequency and bandwidth of a list of protocols. The protocols I have included in this proof of concept are 2.4GHz WiFi and ZigBee although more could be added simply by adding them to the 2 dimensional array I have used to store them. The program will return a probability of detection by comparing the number of above-average values found against the number of values that were expected. Calculating the probability was not included in the initial idea, however after writing the code to handle detection I realised it could be an extremely valuable addition and I already had the values stored from earlier in the programs execution.

3.3.1 Signal Processing

When the data is received, it is piped into the python program in the format of 512 power values. The raw signal data these power values show is not directly usable in its received state.

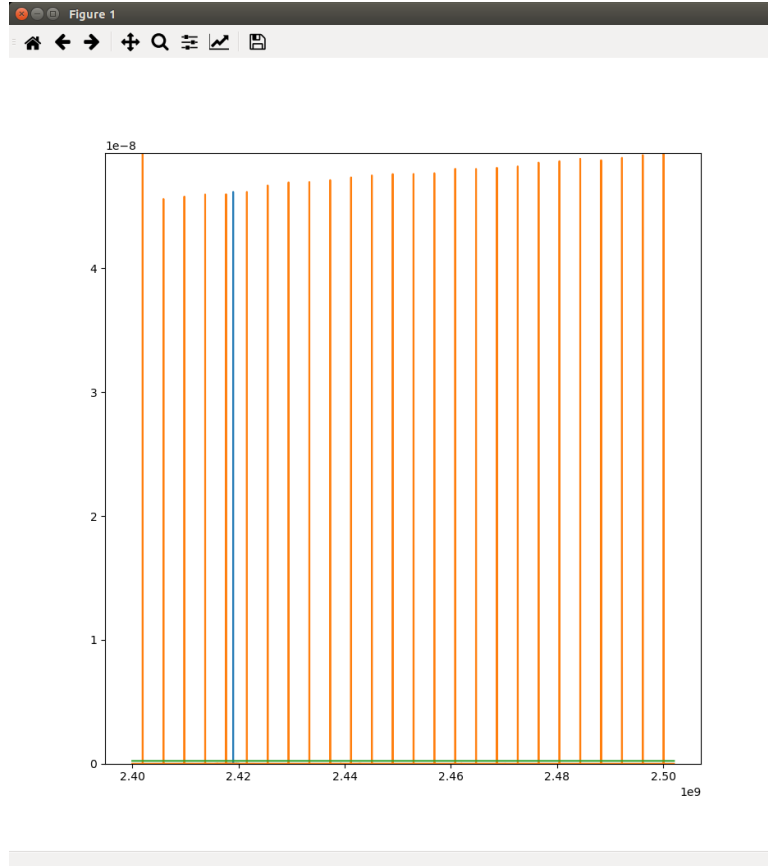


Figure 2: The raw data shown with no signal processing operations, every peak is the IQ peak of each 2MHz wide sample

Thanks to the way SDRs collect and process signals, most SDRs will have a sharp peak perfectly in the centre of their tuned frequency. This is caused by the hardware down-converting the signal received so that the center of the tuned band is located at 0Hz before before it is passed into the SDRs ADC. Because the frequency is 0Hz, it is not oscillating and as such, any DC bias voltage on the input will result in a noticeable and constant

peak. This peak is present in every SDR I have ever seen or tested and if it needs to be removed, it can only be done in software. To remove this peak in my program I remove the middle 20 data points and set their value as the average of their neighbouring real values. While this does remove some data, the bandwidth affected is a mere 78KHz, which is a fraction of the smallest bandwidth protocol being tested for, which is ZigBee at 600KHz wide.

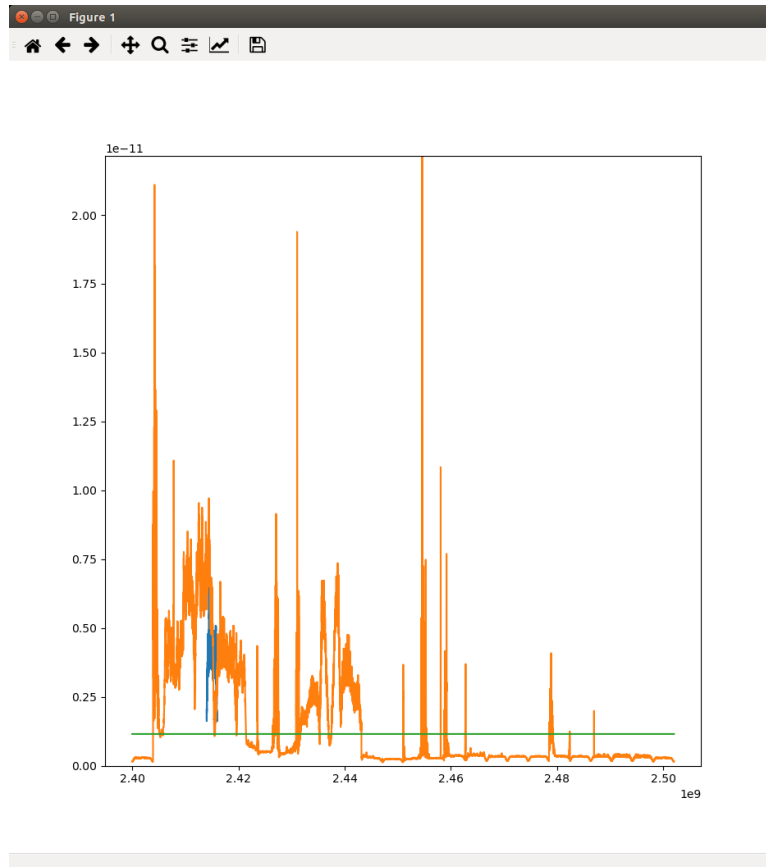


Figure 3: The raw data with the IQ peaks removed

The second issue with the received data is the rapid drop in signal power at the start and end of the scanned area thanks to signal attenuation from the internal band pass filter which is designed to allow through any signals received within the configured range and not allow anything outside that range to pass through. Band-pass filters are not perfect however, as instead of completely removing anything out of their range they attenuate the signal

more the further outside the frequency range it is. These drops would lower the average and increase the chance of having noise being above the average value and being tested as though it were a valid signal. To attempt to remedy or at least reduce the effect this has on the data, I change the value of the first and last 60 data points to the value of the 61st and 451st data points. While, again, this does remove data, the value of the data being lost is far less than any data towards the centre of the spectrum as its power cannot be accurately recorded due to hardware limitations. The affected bandwidth is 234KHz per side, resulting in 469KHz of "lost" data, while that may seem like a large amount of data being lost, the data is effectively useless to begin with.

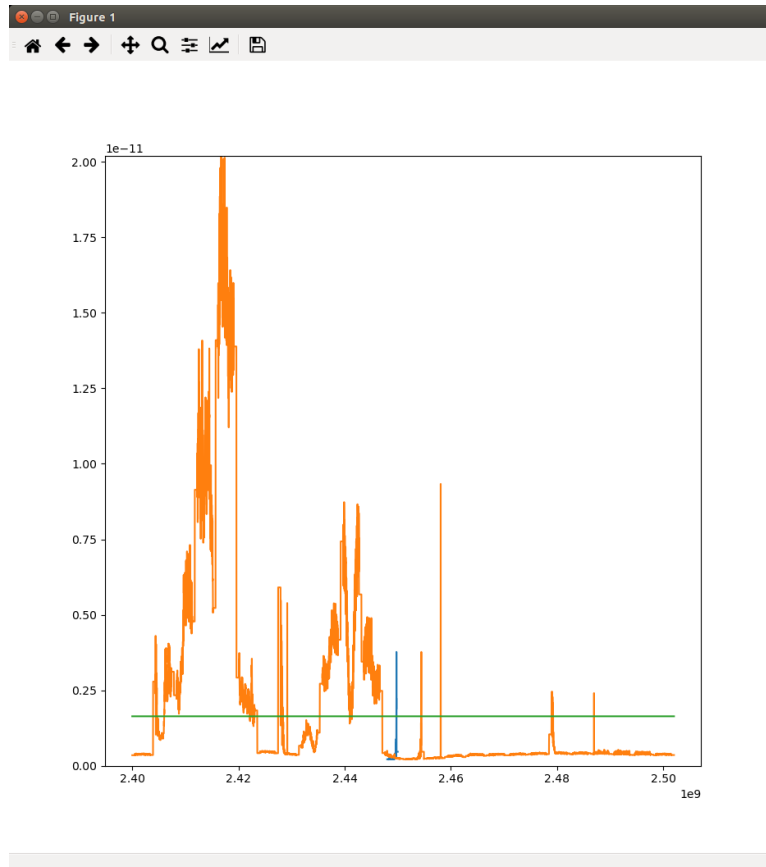
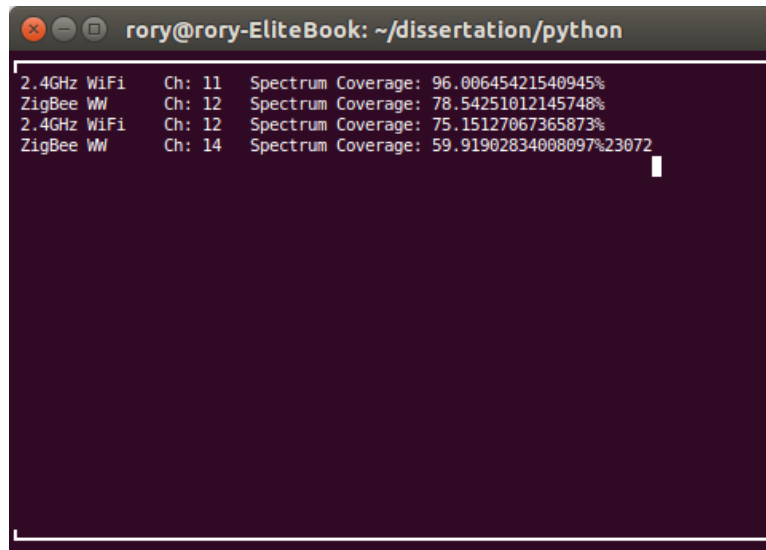


Figure 4: The raw data shown with the IQ peak removed and the edge signals altered

3.3.2 Data Display

Curses is used to provide a list of results to the user in a nicely organised manner. The output is in the form of an ordered list, from highest probability to lowest. Each protocol detected shows the protocol, the channel that was detected and the percentage certainty that the reading is accurate.



```
rory@rory-EliteBook: ~/dissertation/python
2.4GHz WiFi Ch: 11 Spectrum Coverage: 96.00645421540945%
ZigBee WW Ch: 12 Spectrum Coverage: 78.54251012145748%
2.4GHz WiFi Ch: 12 Spectrum Coverage: 75.15127067365873%
ZigBee WW Ch: 14 Spectrum Coverage: 59.91902834008097%
```

Figure 5: The console output of the final system

The data received and collected by the signal processing stage is collected, then graphically plotted using matplotlib with the x axis being the frequency and the y axis being the received signal power on an arbitrary scale. This plot is most useful to confirm that the antenna is connected, working well and that data is being received. It has proven to be extremely useful for debugging.

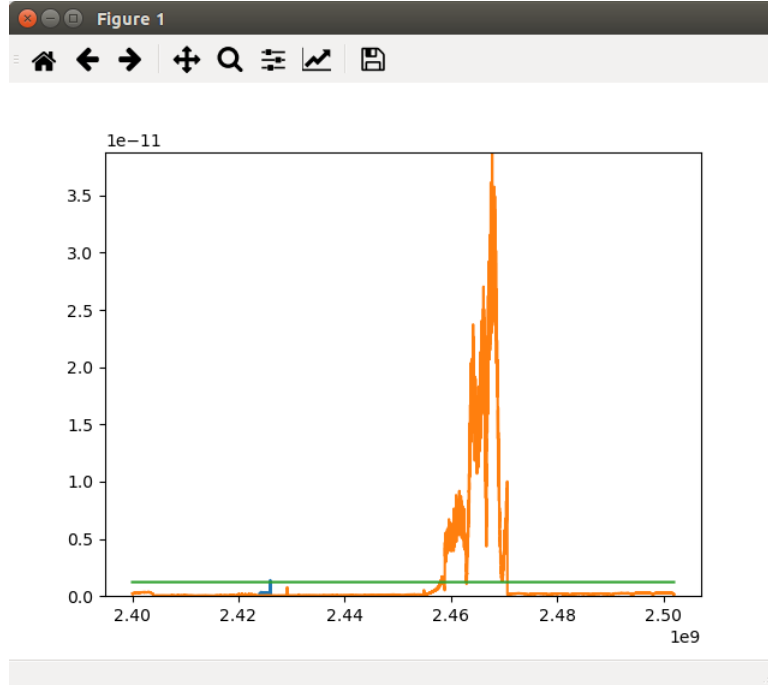


Figure 6: The spectrum plot from matplotlib

4 Evaluation/Results

The system that was created detects power peaks in the RF spectrum and compares the bandwidth of the peaks to a list of known bandwidths and frequencies for a collection of protocols. If the bandwidth and frequency are a close match to the items in the list then it can be assumed that that protocol may be in use on the frequency mentioned. The system outputs a list of protocols it may have detected along with a percentage certainty based on the percentage of the spectrum it has detected.

The system was found to be capable of detecting the protocols currently in use in the area to an accuracy of one channel, showing that the signal fingerprinting requirement has been met. For example, if WiFi channel 11 is in use, it may detect channel 12 or 10 at a slightly higher probability than the actual channel in use. This is a flaw with the system that I simply did not have enough time to attempt to correct without a major rewrite that would take more time to perform than I have available.

Protocols that exist in the same RF band are difficult to differentiate

in the current implementation. The current implementation will detect the traffic in that band and attempt to identify it, however when two protocols overlap the detection will show results for both potentially being present. When a protocol with lower bandwidth operates at the same frequency as one with a higher bandwidth, if a device using the higher bandwidth protocol is transmitting it will result in a false detection of the lower bandwidth protocol because the RF peak will cover the entire range of that lower bandwidth protocol and only most of bandwidth used by actual system that is transmitting. The only possible case included in the current system is ZigBee and WiFi, which results in ZigBee being falsely detected every time a WiFi signal is present. This could be fixed with further research as the peaks would overlap, creating an interference pattern where simultaneous transmissions are occurring that could be used to identify signals inside larger signals. However such a solution is far outside my current knowledge and although the hardware is capable of this, I am not capable of implementing it at time of writing.

5 Discussion

Existing SDR software is fraught with problems for a large percentage of devices that exist under the £700 price bracket. The manufacturer of the hardware does not make enough of a profit to dedicate a development team towards working on support and integration with existing systems. That combined with the recent surge in availability and lack of universal driver support means that a manufacturer often produces their own drivers, many of which function completely differently to similar products from other manufacturers. Host computer software support is often lacking as it is the responsibility of the community to integrate support for the SDR hardware and its drivers into their projects, most of which are open-source and the developers are unpaid. As a result the number of programs that support the hardware is proportional to the number of individuals that purchased it and for newly launched products the support will be extremely limited because potential users realise that little software supports it, often resulting in a feedback loop that ensures the hardware goes unsupported and has a very low adoption rate. A large number of projects are still in their infancy thanks to the recent and sudden increase in SDR availability that has caused people to find uses where software does not already exist and create their own solutions. All these factors combine to create a rapidly evolving environment of inter-operating programs that often gain and lose compatibility

on a daily basis through updates and lack of universal systems that ensure reliable inter-operability. SDR technology cannot reliably be used for channel hopping protocols. The time required to wait until the SDR has tuned to its specified frequency and any noise introduced during tuning has settled is simply too long. I believe that within a few more years an ASIC that is capable of rapid tuning will be created. However, if a channel hopping protocol must be used then the RF bandwidth of the SDR has to cover the entire possible range used by the protocol from start to finish with every sample. This is the only appropriate method for handling channel hopping systems and ensuring no data is missed. The technique often requires SDRs with enormous bandwidths by today's standards, high bandwidths mean that a large amount of data is collected and needs to be transferred over a high speed interface to a well equipped host computer. These and many more factors all contribute to a rapid price increase, far beyond the reach of a consumer or independent professional. The use of SDR at high bandwidths requires an enormous amount of processing power to handle in a useful time-span. Luckily the data can be processed in parallel using general purpose compute in graphics cards like NVIDIA's CUDA. Massively parallel systems are required to process the high throughput of the system, for example the system has a throughput of 320MB/s when receiving at a bandwidth of 60MHz. For any heavy processing other than tasks such as recording the data for storage to a hard disk, a general purpose CPU is not simply not sufficient.

6 Reflection on Project Management

The system initially planned for creation was an automatically decoding sniffer that should be capable of decoding ZigBee and one more to be determined protocol. However there were many setbacks during the planning stages meaning that there was not much purpose behind formulating a detailed plan as the conditions changed week on week. The initial plan quickly fell flat when the hardware I received did not meet the necessary standards of sensitivity required to create the project. The signal the HackRF received was simply too weak and had too little resolution to be of any reasonable use while the HackRF's bandwidth was sufficient for the end product I produced, it was not enough for the design I had hoped to create at the time. These made me realise that it was better to drop the use of the HackRF from the project and search for another solution. The situation was compounded when it was suggested that I use the department's Ettus Research E100 SDR. However upon inspection of the hardware it was quickly discovered

that it was a software defined radio, that was missing its radio components. That path lead nowhere rather quickly. Eventually after a couple of weeks attempting to find an SDR that can operate at 2.4GHz I realised that the 802.15.4 standard, which ZigBee uses as its Layer 2 protocol, allows for some devices to operate at 900MHz as shown in the table below.

IEEE 802.15.4 allows for:

Number of Channels	Frequency Band
16	2.4GHz
10	915MHz
1	868MHz

This would have made it ideal to use a low-cost RTL-SDR dongle if I could source some 915 or 868MHz transceivers. However that proved to be more difficult than expected as very few manufacturers produce transceivers operating on those channels and none that I found offer a breakout board version of their modules, meaning that it would require additional monetary investment and time to design and manufacture circuit boards for testing the modules. In the UK the use of the 900MHz spectrum is partially allowed under European standard EN 300 220-2[16]

Band	Frequency Limits (MHz)	Applications	Relevant Standard
915MHZ	915-921MHz	RFID	EN 302 208
		Non-specific SRDs	EN 300 220
		Assistive listening devices	EN 300 422

However most UK devices operate at 433 MHz instead and I was not able to determine if the devices I would be creating would qualify as an SRD or "Short Range Device". As a result of these it became difficult to source and either illegal or extremely expensive as the hardware had to be shipped from the United States.

The LimeSDR was delivered on 18/03/2017 giving approximately one month to create any sort of useful product after the previous setbacks. Admittedly this was hardly ideal and the HackRF could possibly have been used to build a suitable testing platform as the API I chose to use (SoapySDR) can also be modified to support the HackRF. Doing that may have added a week or so to the development time by using it to ensure that data can be collected correctly, even if it is not sensitive enough to determine what is a valid signal and what is noise. No testing can be carried out without a radio.

Once I had the SDR hardware I had some serious issues getting any useful data from them and various forms of this issue persisted for weeks. The

first attempts at acquiring data from the SDR hardware involved using the Python SoapySDR library to directly pull raw data from the hardware. This began to cause problems almost immediately as the data is presented as an array of complex numbers represented as two 32 bit floating point numbers. I had no previous knowledge of complex numbers or how to use them and after nearly a week of trying to understand it and consulting others I believed to be more knowledgeable on the matter than myself, I was still no closer. My initial starting plan was to plot the spectrums power values on a bar graph to show that I was receiving useful data and have the radio tuned correctly. If the radio was tuned to a frequency where I knew a broadcast is occurring it should show as a peak in the graph, if there is no peak then there is a problem. However graphing these complex numbers proved to be a problem in itself that I could not resolve. Instead of looking to receive my data directly from the radio I had a search for alternate libraries that may already be able to provide the functionality I needed. This was found as another python program called "Soapy_Power" which when run prints a list of 512 power values covering a total of 2MHz of the RF spectrum at a time. To interpret this data I pipe it into a python program that carries out the analysis. Getting this program to work was not the easiest task as it had dependency issues and the program I was writing relied upon python 2.7 whereas Soapy_Power made use of python 3.4 and below. The current version I had installed on my machine was python 3.5. To get this working I had to downgrade the version of python I had installed and also set the default python executable to that version. Just as I finished getting Soapy_Power working my install of Ubuntu 16.04 began failing with approximately a 1/6 chance of successfully booting without becoming completely unresponsive 30 seconds after login. To fix this problem and at the same time make the re-installation of all the dependencies easier, I retrieved my data and code from 16.04 and installed Ubuntu 14.04 instead. Because this is an older version, a lot of the dependency problems were resolved easily as the packages were originally written to support 14.04 from the beginning. Re-installation of the software and drivers only took roughly 6 hours of work. It is also worth noting that the software ran much faster on this version too. I would consider this a positive outcome since it made me realise issues I didn't know I had in the first place, solved those issues and increased my systems overall speed and stability. By the time the radio arrived, and after encountering the problems mentioned previously, I only had roughly 3 weeks to produce a workable proof of concept, which is very little time. I had to settle for a relatively simple design that scans between two frequencies in 2MHz sections and attempts to identify protocols by searching the detected power values

for a peak that correlates with the bandwidth of a protocol.

The project was ambitious from the start, so to keep myself on track I set myself three targets. I hit two of those targets, the ability to fingerprint a protocol or device, and the ability to target more than one platform or protocol at a time. However there are areas that could have been improved. This project required an enormous amount of reading and research before I could even begin to understand how to achieve my goal. I believe that with hindsight, perhaps a project better fitting closely with my existing knowledge would have been a sensible choice since it should have allowed me more time to create a design instead of simply studying how the components should work. I understand however that the success of the project is not the end goal of the module, I understand that the purpose of the module is to create new knowledge and effectively document it. I am happy that I have pushed myself, believe I have met the requirements of the module and do not regret my choice. Python was my language of choice, I mainly chose this because of its wide range of existing libraries and dynamic typing. The dynamic typing is particularly helpful as the data I handle consists of several different data-types and I regularly have to convert between them. These traits make python ideal for creating proof of concept implementations, however one of python's largest shortcomings is its speed. Python is a rather slow language because it is interpreted, thus adding additional steps to the code's execution and decreasing the number of operations that can be performed per unit of time. This caused problems during the coding process as the analysis performed after every successful cycle has hundreds of thousands of tests and operations to perform while new data is already being piped in. The enormous number of operations performed causes the program to appear to freeze for a few seconds while it processes the information. I had to write the program in a manner which tries to perform as little operations as possible but the unresponsiveness still cannot be completely prevented. It should go without saying that this is hardly ideal. In the future I would like to port the project to a compiled language, enabling the use of "bare metal" hardware instructions directly and as such can perform operations significantly faster. The SoapySDR API has hooks available for python, C and C++ languages[2]. C++ would have been a strong language choice because it's fairly simple to use, has hooks available for the API I had to use and is a compiled language which should provide the needed speed-boost. It took months to acquire suitable hardware for testing. The initial HackRF was not sensitive enough to reliably differentiate signal and noise and the only other 2.4GHz capable piece of hardware was the Ettus Research E100 which was missing its radio module, after searching it could not be found.

The only other option at the time was to investigate using an RTL-SDR dongle which could reach a maximum frequency of 1.7GHz and meant that alternative protocols had to be investigated because ZigBee, Bluetooth and WiFi all operate at 2.4GHz. Luckily after close to a week of searching for a solution and having difficulties implementing that solution, the hardware I had originally planned to use in the project arrived and hardware was no longer an issue. With hindsight, I really should have ensured suitable hardware was available before carrying out a project that is heavily reliant on said hardware. The LimeSDR arrived so late because of worldwide component shortages and excessive shipping times during production. These were out of my control but meant that the product, originally due for shipping in early November, was shipped on the 9th of March and received on the 18th. Over three months behind schedule. In the time before it was finally shipped, I did contact the manufacturer to enquire if there was any chance of getting a unit early as I knew they had already shipped some earlier units (see Appendix 11.1). I clearly stated my position, mentioning that I was an undergraduate student and wished to use their product for my dissertation, I even made sure to send the E-mail from my university email address to give it some credibility. A couple of weeks later I received a reply from a Lime Microsystems employee stating that they had noted my order number and would attempt to ship the product in January but that it may not be possible. January soon came and went without the product being shipped.

The project time-line was planned out, however the plan quickly fell apart as the issues began to appear. The initial plan was as follows:

Task	Date Start	Date End	Done
Determine the project idea	20/01/17	03/02/17	X
Initial research	04/02/17	03/03/17	X
Radio selection	04/03/17	11/03/17	X
Implementation and testing	12/03/17	13/04/17	X
Writeup	14/04/17	02/05/17	X

As mentioned previously, this was not the plan that was eventually followed. A revised plan is shown below:

Task	Date Start	Date End	Done
Determine the project idea	20/01/17	03/02/17	X
Initial research	04/02/17	11/03/17	X
Radio selection	04/02/17	11/03/17	X
Implementation and testing	12/03/17	14/04/17	X
Writeup	15/04/17	02/05/17	X

This project requires an enormous amount of research and understanding before anything remotely useful can be created, most of my time was spent attempting to find solutions to problems that require in depth knowledge at both the hardware and software level which considering the end product does not show in this report. The documentation for many of the techniques used exists, but is often hidden as much of it falls under the RF engineering umbrella which requires many years of training to understand well.

7 Social, Legal and Ethical Issues

Sniffing by nature means that anything that operates on the frequencies being scanned is technically collected and much of it could be without permission. In my use case none of it is stored and no data transfers are recorded, just the RF power of those transmissions. Technically this classifies as RF meta-data and could be used to potentially locate a user or broadcasting device, but not determine the content of the transmission itself. It is illegal to sniff any communications, including GSM, LTE or WiFi networks without clear authorisation[18]. Although my product does not store the data it may technically be classified as a sniffer and as with any sniffing hardware there is a potential for misuse. In my implementations current implementation the risk of misuse is low as it would need to be heavily modified to collect actual packets the current functionality can be achieved on a protocol-by-protocol basis using far more inexpensive hardware such as a USB WiFi adaptor.

The hardware necessary is still fairly large and carrying around a block coated in antennas in public is perhaps not going to without some disapproving or curious looks from members of the public. However I suspect that in a couple of years this will be largely solved as radios continue to shrink in size. However I can say through experience it is hard to avoid being noticed when using an SDR in a position where it will receive a strong signal.

8 Feedback From Presentation

- evaluation mention further research
- mention further work (hard work, took a lot of understanding, not shown in end product)
- problems like identifying zigbee where it isnt present (wifi)
- explain algorithm and reasoning behind it

- like covering whole bandwidth in 2MHz blocks
- antenna choice
- in an ideal world type solution.
- block diagram of implementation

Also see appendix 11.2.7 for direct feedback and other email conversations for proof of working through recommendations.

9 Conclusion

SDR has its limitations, mainly that it can not carry out frequency hopping at the speed required by most protocols. This is often overcome by simply increasing the overall RF bandwidth being received and ensuring that it covers the full spectrum in use by the protocol being targeted. Even with these limitations SDR is still extremely under utilised and has enormous potential for not just security analysis, but the wireless ecosystem as a whole. I have outlined an easily expandable system that is capable of detecting and identifying non-channel hopping wireless protocols using a simple method that shows how easy making use of such hardware can be once the underlying concepts are understood.

10 References

References

- [1] Ross Anderson. *Security engineering: a guide to building dependable distributed systems*. 1st ed. John Wiley and sons, 2001, p. 240.
- [2] Josh Blum. *SoapySDR Wiki*. June 23, 2016. URL: <https://github.com/pothosware/SoapySDR/wiki>.
- [3] Dave Evans. *Cisco Internet business solutions group (IBSG) the Internet of things how the next evolution of the Internet is changing everything*. Apr. 2011. URL: http://www.cisco.com/c/dam/en_us/about/ac79/docs/innov/IoT_IBSG_0411FINAL.pdf (visited on 02/02/2017).
- [4] Bruce Alan. Fette. *Cognitive radio technology*. Elsevier ; Academic press, 2009, p. 3.

- [5] *FPRF MIMO Transciever IC With Integrated Microcontroller*. LMS7002M. Rev. 2.2.0. Lime Microsystems. Aug. 2014.
- [6] great scott gadgets. *HackRF One*. 2009. URL: <https://greatscottgadgets.com/hackrf/> (visited on 02/28/2017).
- [7] Dan Goodin. *Smart TV hack embeds attack code into broadcast signal—no access required*. Mar. 2017. URL: <https://arstechnica.com/security/2017/03/smart-tv-hack-embeds-attack-code-into-broadcast-signal-no-access-required/>.
- [8] “IEEE Standard for Information technology–Telecommunications and information exchange between systems Local and metropolitan area networks”. In: *IEEE Std. 802.11-2012* (2012).
- [9] “IEEE Standard for Low-Rate Wireless Networks”. In: *IEEE Std. 802.15.4-2015* (2015).
- [10] VANU Inc. *Vanu, Inc. One Cambridge Center Cambridge MA 02142 www.vanu.com Copyright © 2006 Vanu, Inc. The Vanu Anywave™ Base Station Subsystem*. Apr. 2006. URL: <http://www.vanu.com/wp-content/uploads/vanu-anywave-2006-05.pdf>.
- [11] *Internet of Things Research Study*. Sept. 2014. URL: http://go.saas.hpe.com/1/28912/2015-07-21/32bhy3/28912/69168/IoT_Report.pdf.
- [12] *Multi-band multi-standard transceiver*. LMS6002. Lime Microsystems. 2009.
- [13] Nuand. *Setting up Yate and YateBTS with the bladeRF*. May 2016. URL: <https://github.com/Nuand/bladeRF/wiki/Setting-up-Yate-and-YateBTS-with-the-bladeRF>.
- [14] Yongtae Park et al. “Enabling Sensor Network to Smartphone Interaction Using Software Radios”. In: *ACM Transactions on Sensor Networks (TOSN)* 13.1 (2016), p. 2.
- [15] Sam Shearman and James Kimery. “Software Defined Radio Prototyping Platforms Enable a Flexible Approach to Design [Application Notes]”. In: *IEEE Microwave Magazine* 13.5 (July 12, 2012), pp. 76–80.
- [16] *Short Range Devices (SRD) operating in the frequency range 25 MHz to 1 000 MHz*. EN 300 220-2. European Telecommunicatios Standards Institute. 2016.

- [17] Defense Industry Daily staff. *Soldier Battle JTRS: The HMS Radio Set SANR*. June 2015. URL: <http://www.defenseindustrydaily.com/soldier-battle-jtrs-the-hms-radio-set-07536/>.
- [18] *The Privacy and Electronic Communications (EC Directive) Regulations 2003*. 2003 No.2426. <http://www.legislation.gov.uk/uksi/2003/2426/introduction/made>. United Kingdom Parliament. Sept. 2003.

11 Appendix

11.1 Email from Lime Microsystems

Hello Rory,

Thank you for your support on LimeSDR, appreciated.
Your order ID is noted, we will try our best to ship LimeSDR to you in January, please bear with us though.

Have a peaceful Christmas.

Cheers,
Jimmy

Hello,

I am an undergraduate student at Coventry University studying Ethical Hacking in my third year. I am also a backer of your limeSDR project and backed in the hope i can use it for my dissertation project to create a universal wireless sniffing platform.

I am writing to ask if there is any possibility that i may receive my parts before or soon after the 30th of January since that is the point at which the project must be started.


I believe I was a second flock backer. Order number 30465.

If there is anything that can be done that would be great.
Thank you so much for your time,
Rory Bolton

11.2 Emails to and from Supervisor

11.2.1 30/11/16

Project Supervisor

 Daniel Goldsmith <aa9863@coventry.ac.uk>
Wed 30/11/2016, 15:34
Daniel Goldsmith; Sabina Begum <begums63@coventry.ac.uk>; Rory Bolton <boltonr3@coventry.ac.uk>; Toby Marshall <marsha88@coventry.ac.uk>; +4 more ↗

 Action Items

Hi all,

I know I have spoken to some of you individually, but lets try to get one (or two) group meeting in so I can talk about the project.

Need some Idea of your availability.

Can you let me know your:


Availability on Tuesdays
Availability Weds 1-4
Availability Thurs 1-4
Availability Fri 1-4

Will herd some cats and get time set aside.

11.2.2 23/01/17

Project meeting

 Daniel Goldsmith
Mon 23/01, 14:03
Rory Bolton ↗
Required: Daniel Goldsmith; Rory Bolton ↗

 **When:** Fri 27/01/2017 10:00 - 11:00
Where:

 Accept  Tentative  Decline  Propose new time

This invitation was updated after this message was sent. [Open the update](#) or [open the item on the calendar](#).

11.2.3 02/02/17

RB

Rory Bolton

Fri 02/02, 13:30

Thanks for the feedback, making changes now

DG

Daniel Goldsmith

Thu 02/02, 20:23

It looks fine to me.

Major points are SPAG.

1.1 Is it possible to detect 2.4Ghz (What) etc.

1.4 Home users may also benefit.

1.5

Dont forget the Review / Recommend stages.

- Based on results of study, review detection on common 2.4Ghz Radios

Quite like the lit review, hits the points.

RB

Rory Bolton

Thu 02/02, 04:36

Daniel Goldsmith

303COM Detailed Proje...

27 KB

Download Save to OneDrive Coventry University

This is by no means finished, and I'm no where near done but if you get the chance between when you read this and Friday to give some pointers I'd appreciate it.

Also if it could be done would you be available for meeting like we did last week (10-11 on friday) if necessary?

Thanks

11.2.4 09/03/17

SDR Hardware

DG

Daniel Goldsmith

Thu 09/03, 22:24

[Teaching most of tomorrow, drop in to eh from 10-12 and I will find time.](#)

Get [Outlook for Android](#)

RB

Rory Bolton

Thu 09/03, 17:03

Daniel Goldsmith

I'm having problems with the HackRF again. To the point where i can get a 9x stronger signal out of a RTL-SDR dongle. Does the uni have any other SDR equipment? or will i have to try and source a down-converter for an RTL-SDR. I can meet up tomorrow at 10-11AM or any time past 1PM if you want to talk about it.

11.2.5 18/04/17

Presentation dates?

DG

Daniel Goldsmith

Wed 16/04, 13:27

Rory Bolton

Cool, see you then.

RB

Rory Bolton <boltonr3@uni.coventry.ac.uk>

Wed 16/04, 14:24

Can do

DG

Daniel Goldsmith

Wed 16/04, 08:43

Thursday, about 1?

RB

Rory Bolton

Tue 16/04, 14:44

I'm in uni today if you have the time/want to catch up on progress and sort a date for the presentation. I should be coming in every day this week

11.2.6 25/04/17

Problems

RB

Rory Bolton <boltonr3@uni.coventry.ac.uk>
Thu 27/04, 12:48
Daniel Goldsmith

Just on my way now, might be roughly 10 mins late

RB

Rory Bolton <boltonr3@uni.coventry.ac.uk>
Thu 27/04, 12:49

Just on my way now, migt e roughly 10

DG

Daniel Goldsmith
Thu 27/04, 12:50

writeup-dg.org
39 KB

Download Save to OneDrive - Coventry University

Feedback (we can talk about this)

RB

Rory Bolton <boltonr3@uni.coventry.ac.uk>
Wed 26/04, 15:14

1300 sounds good. I'll see you there then. I'll either be in the cisco lab or on the sofas I'm usually at if you're not already around.

DG

Daniel Goldsmith
Wed 26/04, 15:09

Yes Rory,
Shall we pencil in 13.00. If you prefer another time let me know.
Regards.
Dan.

RB

Rory Bolton <boltonr3@uni.coventry.ac.uk>
Wed 26/04, 14:50

I'll be going in tomorrow. Is there any chance i can have a chat with you in person about this? Preferably some time after 12.

DG

Daniel Goldsmith
Tue 25/04, 10:22

I heard about that myself today. It's so sad.
There are a few things that you can do:
1) Speak to Registry, if something is extenuating circumstances this is. You may be able to get an extension which will give you a couple of weeks to think about things before you have to hand in.
2)The University has a counselling service in the SU / Hub. If you want someone to talk to about it, then they can help.
If there is anything else I can do let me know.

RB

Rory Bolton
Tue 25/04, 09:56

I still want to finish my dissertation work but ive just been told of someones death earlier today so i might not be coming in to uni tomorrow or the day after. That is unless i need to come in. I'll come in for any comments on my dissertation work because i need them, but im still in that odd state of mind where i can't think quite straight.
If you could reply when you read this then that'd be great. I really want to know anything you have to say.

11.2.7 Presentation feedback

Presentation Feedback



Daniel Goldsmith

Thu 20/04, 13:40

Rory Bolton · 5

🔔 ⚙️ Reply all | ▾

* Overview

- Project scans for traffic on various frequencies
- Feeds into python script with array of frequencies and channels
- Curses GUI
- Deals with the input format and works our issues with the pipes.

** In Report

- Be sure to talk about the signal processing that you have done. And describe what this means for the project. How it affects the output.
- Filtering the peaks
- Getting rid of our band data
- How do you deal with false positives. (Mention the Zigbee / Wifi issues, also channel for wifi)
- Looking at the multiple blocks / repeated scanning and why you implemented this. As it should include accuracy.
- How that signal processing worked. Look for average values, and compare peak signal to this
- Looks like a great opportunity to get some Maths in the report.
- Talk about antenna choice and the difference it makes
- Wide band being too noisy, but does it have other advantages.

So we get a percentage certainty that the signals we are looking at are the ones identified.

Given issues with the SDR ou still have a very strong project. While you didnt get the fingerprint tasks complete, you are able to identify that devices are within given wireless spectrum's and channels. This means that the hard part of the project has been done, and its the tweaks to the signal proccssing that are required. Given that signal processing is an extremely specialised area, this future work could be achieved, by collaborating with a signal processing expert, to drive this forward.

11.3 Detailed Project Proposal Form

300/303COM Detailed project proposal

The objective of the detail project proposal is to help you refine your general research question down to a well-focused and achievable piece of practical research work.

The first section: "Defining your research project" focuses on your research question and the plan for conducting your primary method. The second section: "Abstract and Literature Review" is to help you identify current academic sources of literature that are highly relevant to your project and to help you get a head-start in producing your literature review.

Your detailed project proposal will be graded in the second semester – however, it is highly recommended that you submit it by the end of the first semester (04/01/2016) in order to obtain detailed supervisor feedback on your project.

There is no suggested word length for the detailed proposal – although 2000-2500 words would be in order.

The Detailed Project Proposal is worth 20% of the project mark.

300/303COM Detailed Project Proposal

First Name:	Rory
Last Name:	Bolton
Student Number:	5497032
Supervisor:	Dr. Daniel Goldsmith

SECTION ONE: DEFINING YOUR RESEARCH PROJECT

1.1 Detailed research question

Help: Your detailed research question is the statement of a problem within the computing domain which you will address in your project. Refining the research question involves narrowing down an initial question until it is answerable using a primary research method(s) that you will conduct during the time of your project. The refined research question must not be so general that it is answerable with a yes or no answer. It must not be so broad that you would be unable to achieve a solution during your project. The key to this is BEING SPECIFIC: Narrow down the method or technology you will use, narrow down the group that the question refers to (localize a general question) If the project is still 'too big', can you think of a way to work on a part of the problem? Avoid using words that cannot be measured, by you, without a huge research budget e.g. 'effects on society', 'effects on business'. *Example:* The initial question "Does cloud computing effect business" needs narrowing down (*for a start the answer is yes*) What is meant by cloud computing? Or 'effect'? Or 'business', in this question? Refining this first question will involve narrowing it down to something you, personally, can measure. A refined version of this question might be: "Does implementing a cloud based voting system improve the speed of decision making in a small company in Coventry?" This refined question is implementable: You can now identify a small company to work with, document their current decision making processes, implement a cloud based voting system, compare decision making speeds over a limited time period (say 1 month) and evaluate your findings. *A small piece of genuinely new knowledge is produced.*

SDN has a benefit over a hardware locked device in that it is massively more configurable and can be tailored to interact an enormous number of different systems in a way that might not be easily achievable with a conventional system. This is far better when compared to a traditional solution in which the hardware is usually limited to one or a few specific protocols across a very small frequency range.

This project aims to examine "How viable is Software Defined Radio as a wireless sniffing platform?"

This can be answered using the following.

- Is it possible to detect a device operating at 2.4 Ghz using SDR hardware?
- Can devices / protocols be "fingerprinted" by their using either a protocol specific method or through inconsistencies during transmission.
- Can the reconfigurable nature of SDN be used to target multiple wireless platforms / protocols using a single hardware unit.

1.2 Keywords

Help: Include up to 6 keywords separated by a semi-colon; what keywords are appropriate to describe your project in an online database like Google Scholar? Keywords should include the general research area and the specific technologies you will be working with. *Example.* A project that proposes a novel way of visualizing large amounts of twitter feed data may have the

keywords: Data visualization; twitter; hashtags; database design; graphics libraries. For further help, take a look at the ACM keywords list <http://www.computer.org/portal/web/publications/acmtaxonomy>

Software Defined Radio, Identification, Sniffing, Security, Penetration Testing.

1.3 Project title

Help: The project title is a statement based on your detailed research question. For example, the research question *'to what extent does a mobile application reduce the number of errors made in class registers at Coventry University in comparison to current paper based registers'* may be stated in the project title: *"A Wi-Fi driven mobile application for large group registers using iBeacons"*.

Viability of Software Defined Radio as a multi-protocol wireless sniffing platform.

1.4 Client, Audience and Motivation:

Help: Why is this project important? To whom is this project important? A research project must address a research question that generates a small piece of new knowledge. This new knowledge must be important to a named group or to a specific client (such as a company, an academic audience, policy makers, people with disabilities) to make it worthwhile carrying out. This is the **motivation** for your project. In this section you should address who will benefit from your findings and how they will benefit. Example: If you intend to demonstrate that a mobile application that automates class registers at Coventry University will be more efficient than paper based registers - the group who would be interested in knowing/applying these findings would be both academic and administrative staff at Coventry University and they would benefit by time saved and a reduction in their administrative workload. If you are making a business case for an organization explain how the organization will benefit from your findings.

The project would benefit the security professional community by providing a framework that can be used to build a universal sniffing platform. The solution could be far less costly than any other existing solution while being open to modifications and adding additional functionality. The reduction of cost could also eventually make the solution viable as a device monitoring platform within organizations that have a BYOD policy and also for use within consumers' homes inside smart home hubs where multiple wireless protocols are being utilized at once and security may have been a side objective by the designers.

1.5 Primary Research Plan

Help: This is the plan as to how you will go about answering your detailed research question - It must include a primary research method (an extended literature review is not an acceptable primary method). Think and plan logically. Primary methods may include experiments, applications or software demonstrators, process models, surveys, analysis of generated data ...

Example: In the class register example above "to what extent does a mobile application reduce the number of errors made in class registers at Coventry University in comparison to current paper based registers" - the research plan may involve: 1) Collecting and analyzing paper based registers in a given class on five occasions. 2) Identifying the error rate average on these occasions 3) Designing and implementing a mobile application that automatically records attendance in class. 4) Deploying the application in the class on five occasions. 5) Identifying the error rate average of the mobile application on these occasions. 6) Comparison of data and summary of findings.

The primary research method is iterative design of a proof of concept SDR sniffing platform that is

capable of sniffing traffic from a small, purpose built, network of several ZigBee radios using the ZigBee protocol and comparing it against a solution using a hardware defined ZigBee radio.

The initial research would be a literature review to establish any research that has already been done in the field followed by paper based design of a generic sniffing system.

The next step is an iterative design process where an implementation is created based on the IoT devices available at the time, a testbed of devices is built using available equipment. The test devices are then analyzed using a hardware defined radio alongside a software defined radio to help provide a baseline for effective comparison. This process will loop until an effective solution is created and good data is obtained.

Items compared would be:

- Sensitivity
- Cost
- Packets lost
- Error rate
- Ease of use

The final step is reviewing and analyzing the data obtained and drawing appropriate conclusions from it. Then making recommendations for use and also further research.

This is the end of section one.

SECTION TWO: ABSTRACT AND LITERATURE REVIEW (1500 WORDS SUGGESTED)

2.1 Abstract

Help: An abstract is a short summary of a research project that enables other researchers to know if your report or research paper is relevant to them without reading the whole report. It is usually written retrospectively so that it can include findings and results. It is fully expected that you will rewrite your abstract when you come to write your final paper. For now, you should write an abstract of about 250 words that define the project described in section one. Before writing your abstract you MUST read some abstracts from conference or journal papers on *Google Scholar* or from *portal.acm.org* (to understand their style) and then provide your own abstract that outlines what your question is and what you 'did' to answer it.

Software defined radio implements the demodulation and decoding methods in software instead of hardware as is usually used. This reprogrammable design could prove immensely valuable to the community as it adds support for any protocol within its frequency range. With no hardware modifications required to add more protocols; the use of software defined radio could enable rapid security analysis at a lower price point and in a less complex package than using a different device and different software for each available technology.

To achieve this goal a modular software design was created that enables future addition of protocols and other functionality. This paper presents a cost effective solution to multi-protocol wireless reconnaissance for use during a penetration test or general company digital security assessment. The main purpose of the tool is to condense the many devices required to carry out a deep wireless penetration test in the IoT age into a single tool.

2.2 Initial/Mini Literature Review (500 words - 750 words)

Help: A literature review is a select analysis of current existing research which is relevant to your topic, showing how it relates to your investigation. It explains and justifies how your investigation may help answer some of the questions or gaps in this area of research. A literature review is not a straightforward summary of everything you have read on the topic and it is not a chronological description of what was discovered in your field. Use your literature review to:

- compare and contrast different authors' views on an issue
- criticize aspects of methodology, note areas in which authors are in disagreement
- highlight exemplary studies
- highlight gaps in research
- show how your study relates to previous studies

The world is filling with new Internet of Things devices at an alarming rate, there is predicted to be approximately 50 Billion Devices by 2020 (Evans, 2011) and many of these devices will be using various different wireless protocols on many different frequency bands. This makes wireless security analysis increasingly difficult and costly to carry out as the number of devices increases. It also means that any tools that are created will require constant revisions to support the latest revision of a protocol. SDR however is perfectly suited for use in this situation. SDR technology uses a "radio in which some or all of the physical layer functions are software defined" (Shearman and Kimery, 2012) meaning that much of the parts can simply be re-written in software instead of replacing hardware and dynamically reconfigured in real time if necessary.

Researchers have already used SDR to intercept Bluetooth hop synchronization packets, this was shown to be extremely easy thanks in part to the low level access to the hardware that SDR technology requires to function. They state that

it is not only possible to synchronize with the channel hopping carried out between two Bluetooth devices with a >95% probability of success (Tabassam and Heiss, 2008) but that SDR hardware is ideal for the task. However, another paper disagrees and states that an SDR based approach is not ideal when a system uses channel hopping to secure its communications as it requires low response times, which current SDR hardware simply doesn't possess (Picod, Lebrun, and Demay, 2014).

SDR has been used in a penetration testing context to intercept and inject packets between devices operating on the Z-Wave protocol using a slightly modified version of the Scapy framework and two USRP B210 SDR boards (Picod, Lebrun, and Demay, 2014).

2.3 Bibliography (key texts for your literature review)

Help: Please provide references, in correct Harvard style, for at least three key texts that have informed your literature review. If you are implementing an application, select texts which demonstrate how other researchers have tackled similar implementations? The references should be recent and sufficiently technical or academic. Your markers will be looking for you to identify technical reports, conference papers, journal papers, and recent text books. Avoid *Wikipedia* entries, newspaper reports that do not cite sources, and general or introductory texts.

Evans, D. (2011) 'Cisco Internet business solutions group (IBSG) the Internet of things how the next evolution of the Internet is changing everything', Available at: http://www.cisco.com/c/dam/en_us/about/ac79/docs/innov/IoT_IBSG_0411FINAL.pdf (Accessed: 2 February 2017).

Picod, J.-M., Lebrun, A. and Demay, J.-C. (2014) 'Bringing software defined radio to the penetration testing community', *Black Hat 2014*. Available at: <https://www.blackhat.com/docs/us-14/materials/us-14-Picod-Bringing-Software-Defined-Radio-To-The-Penetration-Testing-Community.pdf> (Accessed: 2 February 2017).

Shearman, S. and Kimery, J. (2012) 'Software Defined Radio Prototyping Platforms Enable a Flexible Approach to Design [Application Notes]', *IEEE Microwave Magazine*, 13(5), pp. 76-80.

Tabassam, A.A. and Heiss, S. (2008) 'Bluetooth Clock Recovery and Hop Sequence Synchronization Using Software Defined Radios', *Region 5 Conference, 2008 IEEE*. IEEE. Available at: <http://ieeexplore.ieee.org/document/4562737/> (Accessed: 2 February 2017).

This is the end of section two.

12 Certificate of Ethical Approval



Certificate of Ethical Approval

Applicant:

Rory Bolton

Project Title:

Viability of Software Defined Radio as a multi-protocol wireless sniffing platform.

This is to certify that the above named applicant has completed the Coventry University Ethical Approval process and their project has been confirmed and approved as Low Risk

Date of approval:

27 February 2017

Project Reference Number:

P49201