Institut Universitaire de Technologie,
Aix-Marseille Université

INTERNSHIP REPORT
Diplôme Universitaire de Technologie
Spécialité Réseaux et Télécommunications

Measurement of the radio range and directional antenna gain of micro:Bits

Quentin MARECHAL

University of the West of Scotland

Company supervisor : Duncan THOMSON

Academic supervisor : Corinne HOUSSAIN

2018

Thanking

First, I would like to thank my academic supervisor Corinne HOUSSAIN who helped me have this internship in Scotland without whom it wouldn't have been possible.

I also thank Joseph HEFFERNAN for the help he gave me, so I could settle down when I arrived, be able to live on the campus and find places to go around Scotland.

A big thank you to my company supervisor Duncan THOMSON for the attention put into my internship, the interesting subject proposed, the advices given in plan of my future and for the friendly welcoming.

Thanks to Steve EAGER, lecturer in Computer Networking, for the tips he gave me and the contribution of a good atmosphere in the office of M. THOMSON.

In general, I would like to thank all the UWS staff members who were friendly and helped in any occasion.

Summary

1 Introduction

As part of my DUT* Réseaux et Télécommunications at the University of Marseille-Luminy, I wanted to do my internship in a company abroad while offering me a further discovery of the world of Networks and Telecommunication.

The Scottish University UWS*, especially in Paisley is known to have an Engineering and Computing school. In the context of my choices of continuation of study, I wanted to integrate their teams to be able to discover the art of their work as well as the constraints to overcome on a daily basis.

Through this report I will show you the different steps I've done with the micro:Bits through the experience acquired during my ten weeks of internship. At first, I will describe the company and how it works by emphasizing the place of the service in which I realized my internship. Subsequently I will present my main and secondary missions before taking stock of my entire internship and my current professional project.

2 University of the West of Scotland

2.1 History of the company

The University of the West of Scotland, formerly the University of Paisley, is a public university with four campuses in south-western Scotland, in the towns of Paisley, Hamilton, Dumfries and Ayr as well as a campus in London.

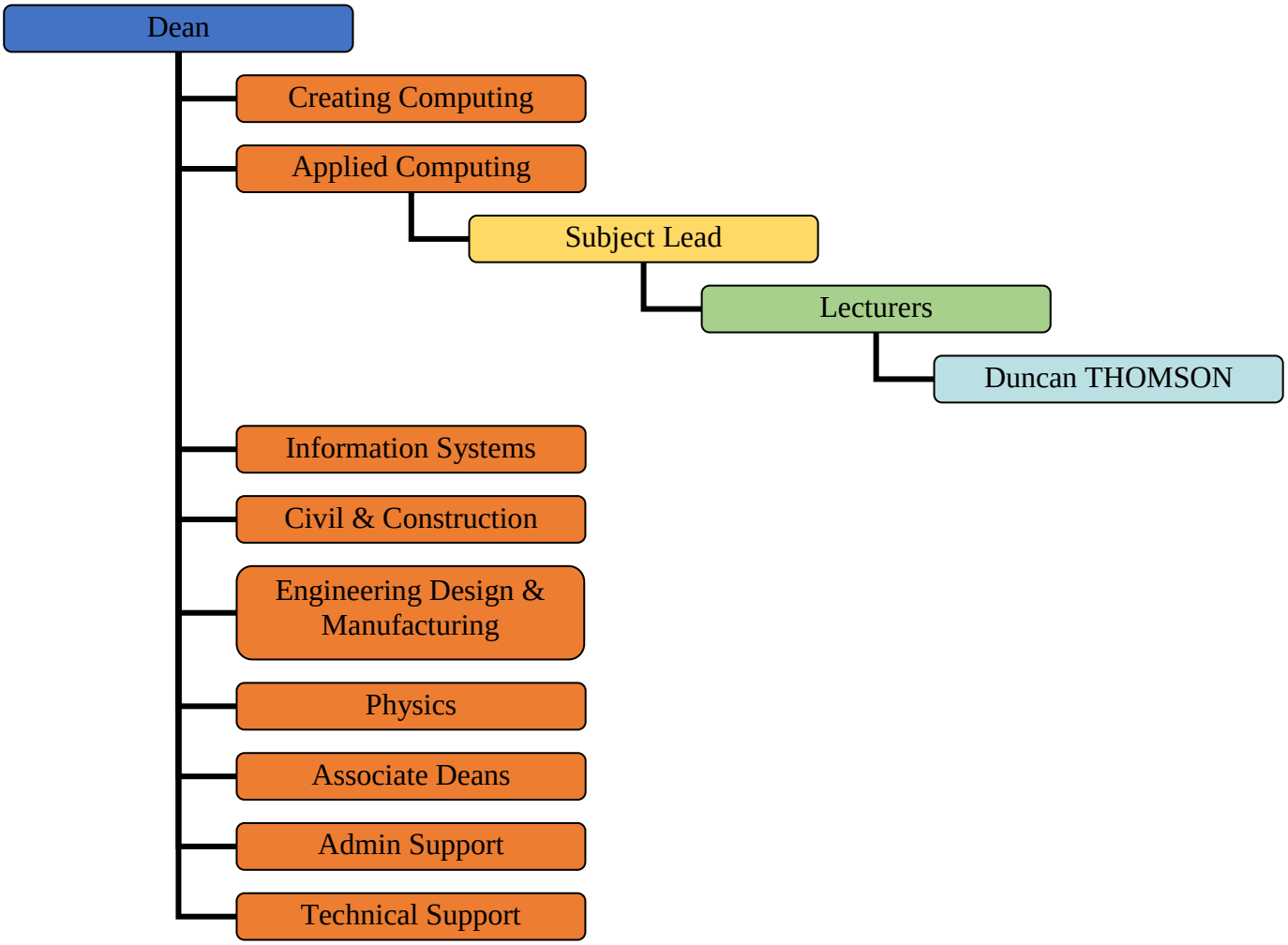
Holding a regional reputation for vocational undergraduate and post-graduate courses the University currently has over 16000 students, with approximately 1300 staff spread across six schools of learning.

The UWS is organized into six academic Schools for learning, teaching and research:

- School of Business and Enterprise
- School of Education
- School of Engineering and Computing
- School of Health, Nursing and Midwifery
- School of Media, Culture and Society
- School of Science and Sport

2.2 Environment of the School of Engineering and Computing

Simplified organizational chart (Picture 1):



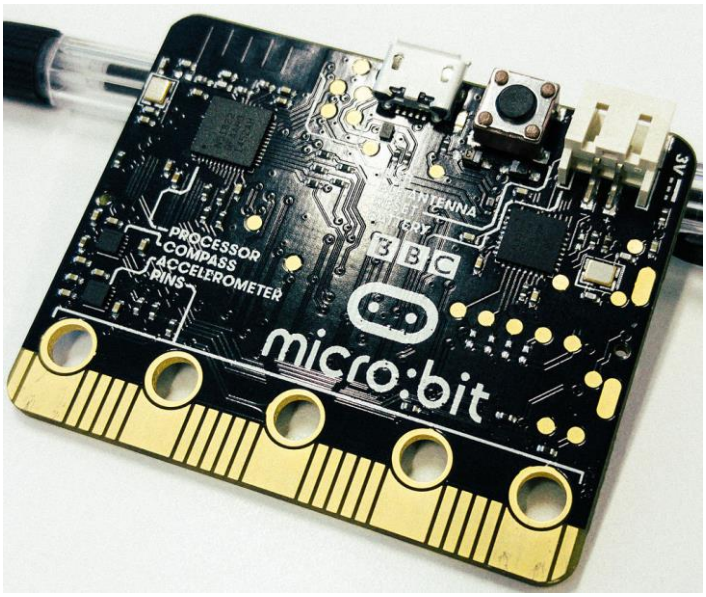
3 General technical framework of the subject

3.1 Internship objectives

During this internship, I had to focus on three main point of the BBC micro;Bits.

3.1.1 Familiarization with the product

The BBC micro:Bits is an embedded system designed by the BBC for use in computer education in the United Kingdom's given to schools all over the country. It was first designed to help schools for learning purposes. The U.K.* wanted young people to be more responsive to programming and therefore let their imagination takes the rest.
I had to search about the micro:bit on internet and try to comprehend the possibilities I had with it. I'll go deeper into what I've learn about it later on.



3.1.2 Radio connection between Micro: Bits

This device can be coded in five different code editors:

- Microsoft MakeCode(formerly Microsoft PXT Editor), using JavaScript and visual blocks, it is also the easiest one to use when you want to start
- MicroPython, using Python
- CodeKingdoms, using JavaScript
- Microsoft Block Editor based on Blockly. Blocky is a client-side JavaScript library for creating visual block programming languages and editors
- Microsoft TouchDevelop

In my internship I had to start off with Microsoft MakeCode and learn Python to use the micro:Bits more effectively. Starting with an easy test to connect them and finish with a full program usable by anyone who wants a radio connection.
These codes would then be used to tests how far certain features of the micro:Bits can go.

3.1.3 Using a MANET* routing protocol

To be able to use multiple micro:Bits without interferences and with the maximum distance possible I needed to do multiple tests.
Knowing what MANET routing protocol exists, how they work and how to use them correctly according the radio range. I was able to discover one type of MANET that could help me, the AODV* protocol. This protocol is both capable of unicast and multicast on mobile networks.
I will explain what MANET protocol is in detail further into the report.

3.2 Specifications

3.2.1 Skills required

The main required skill is mainly to be able to adapt.
With a foreign language, a new programming language and a new routing protocol, being able to put the knowledge I have in similar domains into this internship was a real challenge.

Thanks to my programming skills in Java, C++, I was able to adapt quickly to the programming language Python.
My ability to understand networking also helped me to get used to the MANET protocol and learn how to use it properly.

3.2.2 Technical means

I used BBC Micro:Bits for everything with the exception of a ruler to measure the radio transmission range and a computer to program.

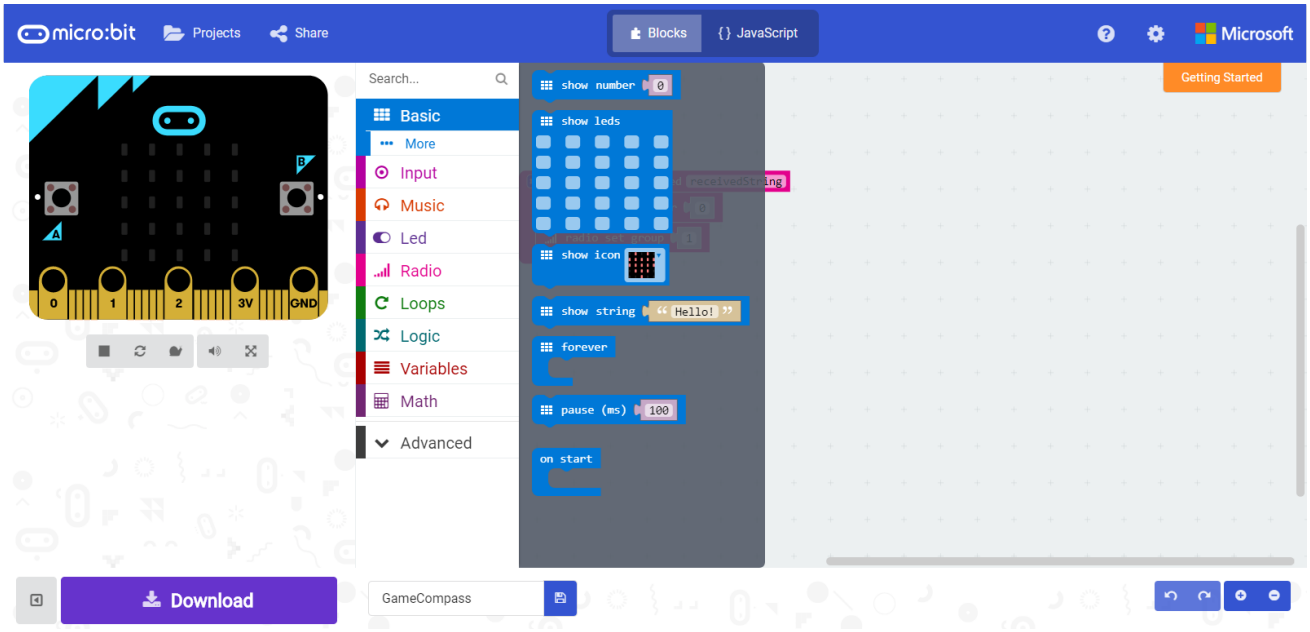
4 Work done

In this forth part, I will detail the work that I have done during this internship

4.1 MakeCode start off

As soon as I arrived in the University, I had to see my supervisor. After a quick review of my objectives for this internship, he gave one micro :Bits at first and told me to go to a website called « http://www.microbit.org ».
His co-worker, working on the same field as him, gave me different documents of projects done with his students, all of them were in JavaScript block editor.
Since then, my first mission was to do those quick projects in the MakeCode editor from the website microbit.org. Each document about those projects were step by step detailed, it was a quick and easy way to get on it.

MakeCode editor (Picture 2):



In the editor we can put blocks with different properties that works exactly like a JavaScript code. We also can directly type in JavaScript by choosing « JavaScript » on top instead of « Blocks » but for a start I would recommend the block because anyone can use it without difficulties.

4.1.1 JavaScript Blocks : Temperature

The first project I've done was about using one of the on-board sensors and a gesture control to display the real world temperature on the micro :Bit.
In the JavaScript Block editor, I selected the Input option from the block menu and find the « On shake » block and just dragged it onto the workspace.
Once on the worksheet the « on » block can be changed to several gestures, such as tilt left or right. But I kept « shake » for this project.
Now the micro :Bit knows when there is a shake gesture, but I needed a variable to store the temperature data.
From the block menu, I selected the Variables option and clicked on « Make a Variable ».
A pop-up appears to name the variable, here it will be « Temp ».
Then I selected the « set item to 0 » and dragged it onto the workspace, from here I was able to replace « item » by the new variable « Temp ».
From the Input option, I used the « temperature » input which will use the micro :bit sensor and then place it instead of the « 0 » on the workspace. So now we have « set Temp to temperature (°C) »

Finally, I wanted to display it, so the « show number » from the Basic option is the way to go.
I dragged this block onto the workspace and change the « 0 » to the variable « Temp »
A picture of the block code is in my annex (**annex 1.**)

4.1.2 JavaScript Blocks : Die Roll

The second project I've done was also about using the gesture input to control the micro :bit by creating randomly generated variable.
It allow the user to shake the micro :bit to show a die face.

Same as the first project I selected the Input option from the block menu and find the « On shake » block and just dragged it onto the workspace.
I selected the « set item to 0 » and dragged it onto the workspace, I renamed « item » to « Roll » by clicking on it and select Rename variable.
I then went to the Math option and selected « pick random 0 to 4 » which will allow me to have a randomly generated number. I put it on the workspace after the « set Roll to » and changed it to « 0 to 5 » (so 6 possibilities, 6 faces of the die)
I needed to display different results depending on the number, the condition « if then else » block from the Logic option is the best to do that.
I added it onto the workspace and by clicking on the gear besides the « if » I added four « else if » and one « else » to the condition block.
From the Logic option on the block menu, I selected the «0 = 0 » block, basically an « equal » block. I put it as the condition and change it to « Roll = 0 » so each time the randomly generated number is 0 , the condition will be true.

To finish, I only need to display it, so I went to the Basic option and chose the « show leds » block. Putting it on the workspace after the « then » and clicking on one square on it to light up a led on the micro :bit when the condition is fulfill.

I duplicated these actions five times, adding 1 to the Roll condition and adding 1 led to the « show led » block each time.
There is the block code in the annex at the end (**annex 2.**)

4.2 Getting into it : Radio communication range in Python

JavaScript blocks are nice but not enough if we want to be more specific into coding the micro :Bit. During my third week of internship, I had a second micro :Bit and I had to do a radio communication in Python between those two and try to see how far 2 micro:bits can be from on to another without losing data.
During my research on the radio module of the micro:bits in general, I found out that it has different "Power level", different "data rate" and each one of them influences the range of the signal.

The power level indicates the strength of signal used when broadcasting a message. The higher the value the stronger the signal, but the more power is consumed by the device. It can go from 0 to 7. The numbering translates to positions in the following list of dBm (decibel milliwatt) values: -30, -20, -16, -12, -8, -4, 0, 4.

Knowing that, I wanted to know how far the signal can go for each Power level and what better way to find out than testing it.
My supervisor told me about an editor called MU which is a simple code editor for beginner programmers, it's written in Python and works on Windows, OSX, Linux and Raspberry Pi.

The first thing to do for a communication is sending a message or number and receive it on the other end.

4.2.1 Micro :bit : The transmitter

Simple send (Picture 3):

```
from microbit import *
import radio
radio.on()

while True:
    if button_a.was_pressed():
        radio.send("a message")
```

First thing to do is to import the library "radio" which contains every piece we need to do a communication.
Starting with "radio.on()" which turns the radio on. This needs to be explicitly called since the radio draws power and takes up memory that I may otherwise need.
And add a condition where each time the button "A" on the micro:Bit is pressed, the transmitter will send a message.

However this message is send in broadcast to every micro:bits around, so how can I limit this message to my 2 micro:bits?
I needed to set a configuration called “channel”. It can be from 0 to 100 and defines an arbitrary “channel” to which the radio is tuned. Messages will be sent via this channel and only messages received via this channel will be put onto the incoming message queue.
This works in exactly the same way as kids’ walkie-talkie radios: everyone tunes into the same channel and everyone hears what everyone else broadcasts via that channel. As with walkie-talkies, if you use adjacent channels there is a slight possibility of interference.

4.2.2 Micro :bit : The receiver

On the receiver I needed a variable where I could stock the message received.
In this case, if the message was equal to “a message”, the receiver would then display thanks to the LEDs, the message received and stocked into the variable.
Simple receive (Figure 3):

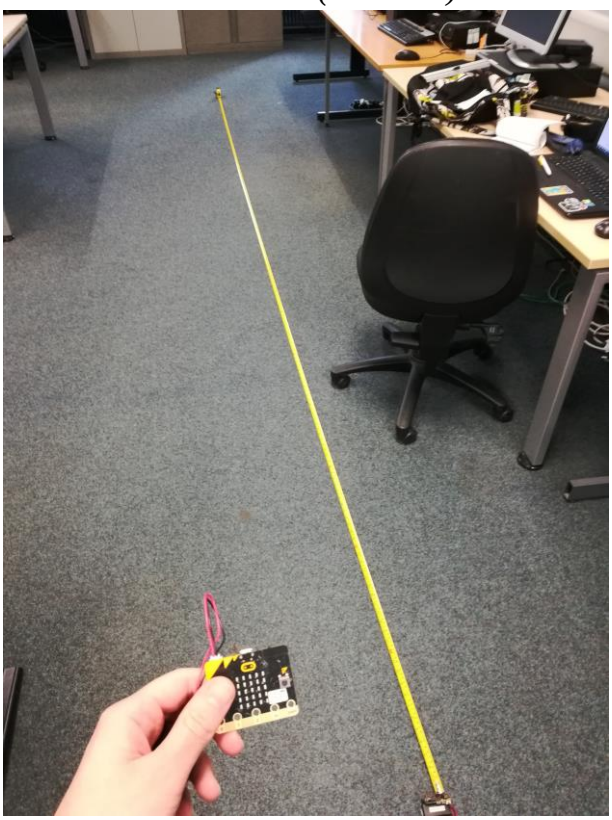
```
from microbit import *
import radio
radio.on()

while True:
    new_message = radio.receive()
    if new_message == "a message":
        sleep(200)
        display.show(new_message, delay=200, wait=False)
```

4.2.3 Range test

Now that I have my codes, I can start testing different range, different power levels and data rates.
I first started with a Power level = 0, to do that I needed to add the code of both receiver and transmitter a line code to config the micro: bit.
It was: radio.config(power=0)
I also wanted to know if the range would vary with data rate.
There is 3 different rate possible for the micro :bit which are 250 KBIT, 1MBIT or 2 MBIT. So I started with the lowest data rate: radio.config(data_rate=radio.RATE_250KBIT)
What I did next was using a tape measure by putting one micro: bit on the floor (in an office I was able to go to work)and try the program every meter until I couldn’t send any more message.

Environment (Picture 4):



I reiterated the operation with a data rate of 1MBIT and a data rate of 2MBIT.
For a power level of 0 and a data rate of 250KBIT, the range of the micro:bit was 6m
For a power level of 0 and a data rate of 1MBIT, the range of the micro:bit was 4m
For a power level of 0 and a data rate of 2MBIT, the range of the micro:bit was 3.5m

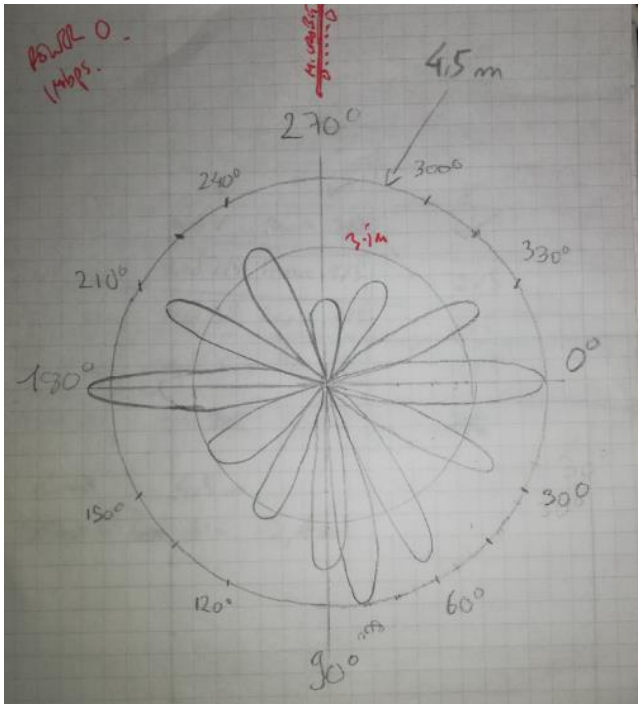
However, sending a message is not enough I need to receive one back.
So, I arrange my code and add a receive variable to the sender code and line to the receiver to send a message when it receives something.
For a data rate of 250KBIT, I started to lose data at 5m
For a data rate of 1MBIT, I started to lose data at 3.6m
For a data rate of 2MBIT, I started to lose data at 2.8m

From those tested I deduced that depending of the data rate, the message could be sent with less data loss. The most viable and consistent data rate was 1MBIT.

4.2.4 Orientation

There was a problem I figured out during the tests. The orientation in which the micro:bit is handled change the range as well. I tried to be as much consistent as I could be during the test (straight micro:bit to straight micro:bit) but now I have to find the maximum range depending on the orientation.
After testing from an angle of 0 to a full 360, I did a polar diagram of the antenna module of the micro:Bit to show the differences according to the angles.

Polar diagram of the antenna module (Picture 5):



I've learned that 180° (back of the micro :Bit to front of the other) is the best angle to have in term of radio range capabilities.

4.3 MANET routing protocol

4.3.1 What is MANET ?

A mobile ad hoc network (MANET), also known as wireless ad hoc network is a continuously self-configuring, infrastructure-less network of mobile devices connected wirelessly.
Each device in a MANET is free to move independently in any direction and will therefore change its links to other devices frequently. Each must forward traffic unrelated to its own use, and therefore be a router. The primary challenge in building a MANET is equipping each device to continuously maintain the information required to properly route traffic. Such networks may operate by themselves or may be connected to the larger Internet. They may contain one or multiple and different transceivers between nodes. This results in a highly dynamic, autonomous topology.

The obvious appeal of MANETs is that the network is decentralised, and nodes/devices are mobile, that is to say there is no fixed infrastructure which provide the possibility for numerous applications in different areas. The early 2000s interest in MANETs has greatly increased which, in part, is due to the fact mobility can improve network capacity, along with the introduction of new technologies.

One main advantage to a decentralised network is that they are typically more robust than centralised networks due to the multi-hop fashion in which information is relayed. For example, in the cellular network setting, a drop-in coverage occurs if a base station stops working, however the chance of a single point of failure in a MANET is reduced significantly since the data can take multiple paths.

4.3.2 AODV protocol

The AODV protocol looked like it could be the best protocol of MANET for my situation.
Ad hoc On-Demand Distance Vector (AODV) Routing is a routing protocol for mobile ad hoc networks (MANETs) and other wireless ad hoc networks.
For multicast groups, AODV builds a tree. This routing protocol is low in energy consumption and does not require large computing power, so it is easy to install on small mobile devices.

I had to readjust my codes to be able to use 3 micro :Bits at the same time.
For example, the three are in a line, the first can talk to the second one but not the third one because he is too far.
So I did a code in which case the first micro: Bit will send a message to the third one through the second one.
The third one would then send a message back to the third one (through the second one again) to tell him he received it.

Here you can see the code of the first one, he just sends a message and wait to receive anything.

Transmitter v.1 (Picture 6):

```
from microbit import *
import radio

radio.on()
radio.config(channel=79)          # N channel
radio.config(power=0)             # Power level
radio.config(data_rate=radio.RATE_1MBIT)     # Rate
msg = "OK"

while True:
    if button_a.was_pressed():
        radio.send(msg)
    ok = radio.receive()
    if ok :
        display.scroll(ok)
        sleep(50)
```

The first receiver which is the 2nd micro:Bits will then send back a message to the first one telling him that his first message was received and sending a message to the third one waiting its answer.

Receiver v.1 (Picture 7):

```
from microbit import *
import radio

radio.on()
radio.config(channel=79)         # N channel
radio.config(power=0)            # Power level
radio.config(data_rate=radio.RATE_1MBIT)    # Rate
msg = "OK"
while True:
    # recu = radio.receive_full()
    if button_a.was_pressed():
        radio.send(msg)
    recu = radio.receive()
    if recu == "OK":
        display.scroll("Ok")
        radio.send("Ok2")
    if recu == "Yes_N2":
        # recu = msg, rssi, timestamp
        display.scroll("2nd")
        radio.send("Yes_N1 and N2")
        sleep(50)
```

When the 3rd micro:Bit receive the message from the second one, he will display an “OK” message to show that he received something and will send back a new message to the 2nd micro:Bit.

2nd Receiver v.1 (Picture 8):

```
from microbit import *
import radio

radio.on()
radio.config(channel=79)         # N channel
radio.config(power=0)            # Power level
radio.config(data_rate=radio.RATE_1MBIT)    # Rate
msg = "OK"

while True:
    # recu = radio.receive_full()
    if button_a.was_pressed():
        radio.send(msg)
    recu = radio.receive()

    if recu == "OK":
        display.scroll("Ok1")

    if recu == "Ok2":
        display.scroll("Ok")
        # recu = msg, rssi, timestamp
        radio.send("Yes_N2")
        sleep(50)
```

If the 2nd micro:Bit receive a message from the 3rd one, so the specific string “Yes_N2”, he need to send back a message that only the 1st micro:Bit will understand so “Yes_N1 and N2”. Proving to the 1st micro:Bit that the message he sent earlier went to the third one.

Now I want to make a code that would work even if I switch the places of the micro:Bits.
To do that I’ll need specific ID for each micro:Bits because they send their message in broadcast so I just need to know who sent what and deliver it to the other one.

I found out that there are 2 settings that could be interesting to me: address and group
Conceptually, “address” is like a house/office address and “group” is like the person at that address to which you want to send your message.
So why not just put specific group to each one of them?
I tried to do that, but I didn’t succeed when it came to know where the message came from

Here’s the transmitter code (1st micro:Bit)

Transmitter v.2 (Picture 9):

```
from microbit import *
import radio

group_id = 1
radio.on()
radio.config(channel=79)         # N channel
radio.config(power=0)            # Power level
radio.config(data_rate=radio.RATE_1MBIT)    # Rate
radio.config(group=group_id)            # Micro:bit "ID"
msg = "OK"

while True:
    # recu = radio.receive_full()
    recu = radio.receive()
    if button_a.was_pressed():
        radio.send(msg)

    if recu:
        if recu.radio.config(group=2):
            display.scroll("Ok2")
            radio.send("Ok1")
            sleep(50)

        if recu.radio.config(group=3):
            display.scroll("Ok3")
            radio.send("Ok1")
            sleep(50)
```

The other 2 codes are the same except for the group ID and the group they want to know about (recu.radio.config(group=….))

I couldn’t do “recu.radio.config(group=2)” neither “recu.group =2” neither “recu.config(group=2).

I also thought of doing a counter to know how much micro:Bits there is after the second one and telling the one where I pressed the button how there is but still, it didn't work. As it is, this program is just going to send broadcasting message, receiving some and then broadcast some other again. So, it's just an infinite loop with a counter increasing to infinite.

Micro:Bit1 (Picture 10):

```
from microbit import *
import radio

radio.on()
radio.config(channel=79)         # N channel
radio.config(power=0)            # Power level
radio.config(data_rate=radio.RATE_1MBIT)    # Rate
msg = "OK1"
counter = 0

while True:
    recu = radio.receive()
    if button_a.was_pressed():
        radio.send(msg)

    if recu == "OK2":
        counter += 1
        display.scroll("Ok2")
        radio.send(msg)
        sleep(50)
        display.scroll(str(counter)+"M", delay=100)
    if recu == "OK3":
        counter += 1
        display.scroll("Ok3")
        radio.send(msg)
        sleep(50)
        display.scroll(str(counter)+"M", delay=100)
```

4.4 Encountered problems and solutions

Regarding my first two missions I didn't really have much problems, it was mainly just testing around. One problem I maybe had was that Python was new to me and I needed to learn it and also the fact that it was hard to find good information about micro:Bits online.
My third mission was surely the one that gave me the most trouble. Is it because I started working on this protocol during the last weeks of my internship, or the fact that I had to learn it.

I managed to find information about the micro:Bit after searching for a long time, but it was harder than anticipated.
Python is an easy language, so I learned the basic very quickly and was able to use it in the beginning of my internship.

4.5 Obtained result

4.5.1 Main missions

In the end I feel like I did mostly what I was asked for, at least for the two first missions.
I still have a regret concerning the last mission that I didn't fulfill in time.

4.5.2 Contribution

My results about the micro: Bits can be used for further research of what it is capable of and I think that's what my supervisor Duncan Thomson wanted.

My code between the micro: Bits didn't work as planned but my ideas and what I tried to do can still be used to develop a code that would work.
I'm confident about the fact that my job was not in vain and that I contributed to M. Thomson and his research.

5 Conclusion

This internship was for me a real opportunity, and if I have to sum up my work done during these ten weeks I would say that I participated in the research and development of the BBC micro:Bit.
First of all by testing range capacity of the radio module from the micro:Bit. Subsequently I would say that I helped in the development of a program allowing three micro:Bits to talk to each other's thanks to a routing protocol.
As I said earlier I'm not happy with my unfinished program on my third mission, but other than that I feel like this internship was interesting.
Doing a work in full autonomy helps with self-management but not having fixed hours was difficult to manage.

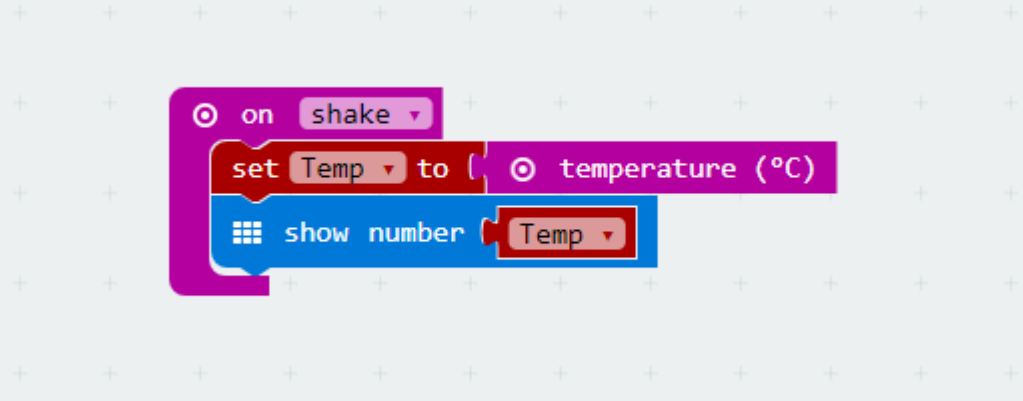
6 Personal review and future

Visiting a whole new country, living and talking someplace where the culture is different was a very good experience to me. It contributed in so many ways to my professional carrier plan, improving in English, having to be more social and open minded.
Since next year I'll be back there for another year doing a licence I think it will be a big plus for me and the opportunities of a job later on.
In the future I would like to work in a company which travel around the world as for example a network technician in Events, so having a diploma from abroad will boost my chances.

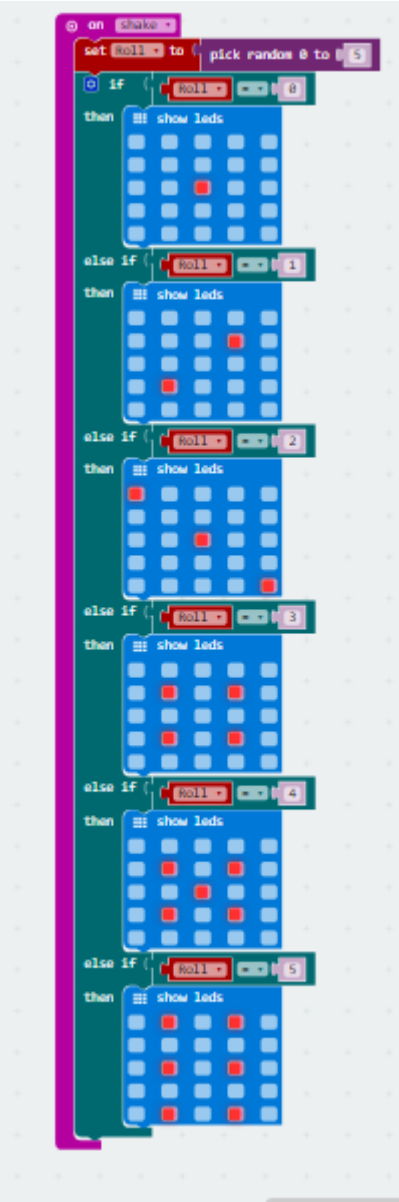
I never thought I would have been able to do an internship abroad or even study abroad! It's a part of my dream but it's just the beginning.

7 Annex

Annex 1 :



Annex 2 :



8 Glossary

DUT(page 7): Diplôme Universitaire de Technologie
UK(page 8): United Kingdom
AODV(page 9): Ad hoc On Demand Distance Vector