



PawPrint

DE0974
Personal Project 02

Luke O'Keeffe
Year 3
Semester 2
Steve Gibson
Interactive Media Design
Northumbria University

CONTENTS

Brief	5	Concept	26
- PawPrint	6	- Tracking collar	27
- Aim	7	- Branding	28
- This Project	8		
Research	9	Components	29
- Competitor Analysis	10	- TinyDuino GPS Car Tracker	30
- FitBark	11	- Existing Code	31
- DOG Tracker Plus	12	- Components	32
- PawTrax S	13	- Processor Board	33
- Chipping	14	- Accelerometer	34
- Current Methods	15	- GPS Shield	35
- What has this shown?	16	- Wifi Shield	36
- Arduino	17	- USB & ICP Board	37
- Cooking Hacks	18	- Components Order	38
- TinyDuino	21		
- Who?	23		
- User Persona	24		

Coding	39	- Testing GPS	58
- Connection Issue	40	- Turning GPS on/off	59
- FTDI	41	- Turning GPS on/off outside	60
- Wifi Code	42		
- Wifi Test	44	What next?	61
- GPS SD Removed	45	- Location Services	62
- GPS Test	46	- Method	63
- GPS Serial Monitor	48	- Tracking Code Explained	64
- GPS Workings	49	- Test 1st Code	66
		- 2nd Code	67
Combining Codes	50	- 3rd Code	68
- Compiling Parts	51	- 3rd Code Test	69
- Combining Code	52	- Spark Core	70
- Buffers	53	- index & update.php	71
- Buffer Test	54	- Map Construction	72
- Testing Wifi	55	- Spark Core Test	73
- Turning Wifi on/off	56	- Finalised Code	74
- Testing both Wifi & GPS	57		

Collar Design	76	Website	93
- Case ideas	77	- Branding	94
- Attachment	78	- Wordpress	95
- Chosen Case	79	- Plugins	96
- Exploded Diagram	80	- Editing Theme	97
		- Choosing Logo	98
Case Construction	81	- Images	99
- Design	82	- Tracker	100
- Creation in Sketchup	83	- Products	101
- Testing .stl file	85		
- 3D Printing	86	User Testing	102
- 3D Hubs	87	- Tracking Collar	103
		- Dog	104
Inner Casing	88		
- Sponge	89	Evaluation	105
- Base Layer	90	- Evaluation	106
- Component Frame	91		
- Sponge Lid	92		

BRIEF

I am continuing the PawPrint brand in order to try and create a fully functioning prototype of the tracking collar, as well as developing the brand image further

PawPrint

I had created the brand PawPrint, in order to allow vets to keep in contact with pet owners. Feedback I received suggested that the app should have been more vet focused, as it requires the connection with an existing vet surgery in order to work effectively. As well as this, the feedback in relation to the design aesthetics suggest its use is for younger audiences. This was seemed to be not fully appropriate, and therefore would have to be altered in some way

Aim

The overall aim of my product is “connecting pets with their vets”. Therefore, in order to do this I am still going to try and maintain the use of my existing interfaces. However, I will be attempting to focus upon improving the overall design aesthetics of the brand image. I will also be trying to create two adverts, one directly focused towards owners, and the other for the vets themselves.

Finally, I will want to produce a working prototype of the animal collar. It should be able to track the animal, as well as record their readings in an easy to understand manner

This Project

My main focus throughout this project will be to be creating the working collar prototype. It will be able to track the animal, and transmit the whereabouts to the vet and the user. On top of this, the accelerometer situated in the collar will record the animals movements and thusly activity. I will also be attempting to improve the overall brand image and advertisements for PawPrint. This will be through a small redesign of the existing interfaces I already have, and well as creating a few extra products (i.e. Dog bowls, toys, etc.)

RESEARCH

I will be evaluating and examining existing products in relation to the collar. As well as this, I will be looking into potential technologies I could use in order to create such a product

Competitor Analysis

There are many animal collars available today. So why does there need to be another one introduced into the market?

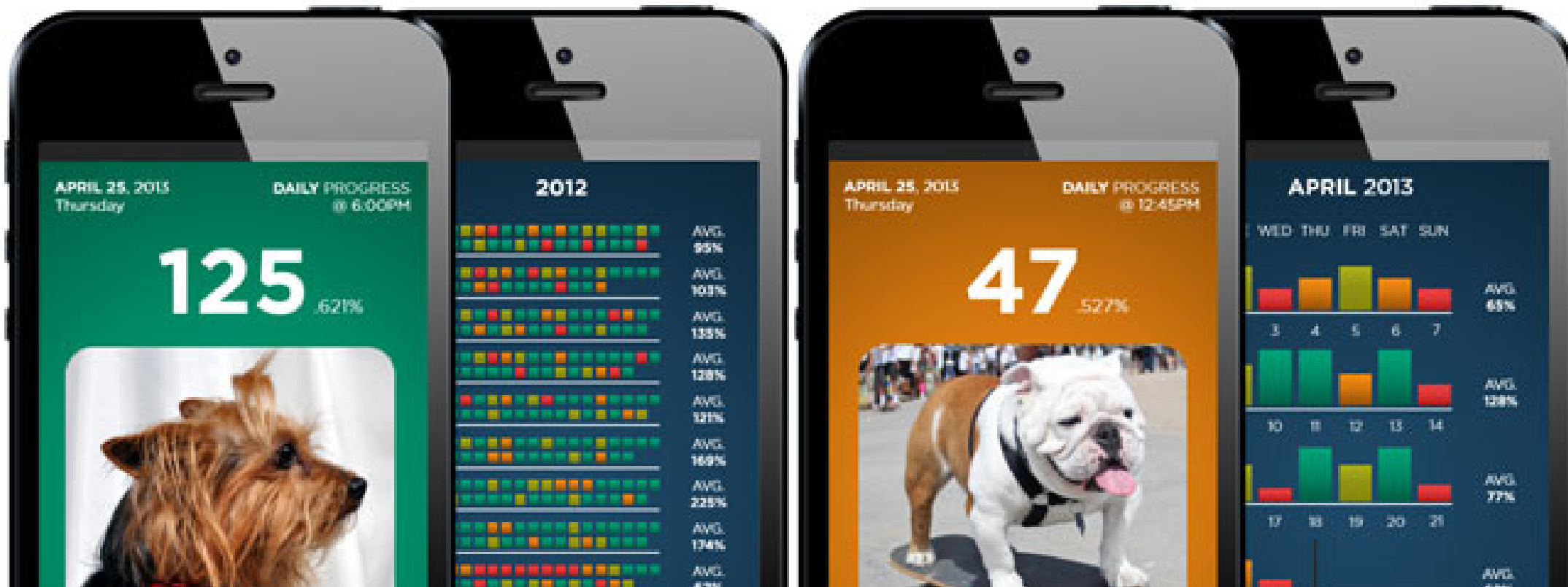
Well the main reason is that they are all designed for one specific area. So why can't they all be combined together at a cheaper price?

This is the issue I am trying to tackle, and see whether or not it would be possible to complete. Therefore, in order to see if I would be able to make something like this happen, I would have to initially analyse certain existing competitors in the market

FitBark

The FitBark product uses a small activity monitor situated in the collar of the dog. It records the movement/behaviour of the dog and then displays this in graphs on an app based interface. The aim of the product is to basically monitor what the dog is up to through its movement as well as its rest. The information recorded is mainly around the dog's pulse, as well as the use of an accelerometer for their activity.

The user can then use real time updates in order to see what the dog is up to. They can also set targets (i.e. amount of "play time" and walking distances) in order to earn 'bark points'. They can also upload photos for people to see, and use a blog in order to comment on their dog's progress (i.e. its fitness or how it is playing)





Safe and secure

Never lose your dog with the world's
most advanced collar and app

DOG Tracker Plus

DOG Tracker Plus, again as with the FitBark, is attached onto the collar of the dog, and has a rechargeable battery (2-5 days life). The collar uses real time updates in order to record and display the information.

It tracks the location of the dog (through its gps tracker), as well as recording the speed and direction the dog is going. A

second useful feature that can be found on the device, is a geofence setting that can be applied. This is a radius in which your dog is allowed to travel safely, and once left, an alert will be sent

The main issue found with the product however, is the cost. It costs around 55p a day (which comes to £200.75 a year). the average life span of a dog is around 16

years, therefore this would cost the owner around £3,212 in order to keep the tracker at all times

PawTrax S

PawTrax S is another cheaper dog tracking collar, however, this brand takes a completely different approach. The collar comes at a one of price of £85 which is overall a lot cheaper in the long run. The way in which the collar tracks the dog, is through the SIM card it has attached to it. The owner registers their phone to that SIM card, and then simply texts if they want to find the whereabouts of their dog

The location of the dog is then sent back through a text message. Even though this is a much simpler method of tracking the dog, it is still not an amazing process. The information about the location of the animal is only sent once when the owner contacts the device. By the time the owner has received this information, the dog could have moved a couple of hundred metres, and the location would be pointless.

As well as this, the overall cost of the collar would still add up, as the owner would be paying for texts in order to gain the information they need

[Sign In](#)
| [Favorites](#)

PawTrax S - Dog

Category: Store > For DOGS



In stock

was £105.00
£85.00
 Save £20.00

f Like
Share
< 22

Colour

- ☒ Black
- ☐ Yellow
- ☐ Grey
- ☐ Blue

SIM

- ☒ with GiffGaff sim (UK)
- ☐ I will supply my own 2G compatible sim

♡ 29

Detailed Info & Buy
For DOGS
For CATS
For KIDS
For PEOPLE
Web Platform - (optional)
Misc/Spares/Collars
Setup Service

Introductory price - £85

Android and iOS app. for setting up and controlling the S now available

Chipping

Today, when any animal is bought they have to be chipped. The owners details (i.e. address, contact number and the name of the pet) are stored onto the chip. This is because, if the pet manages to run away, the owner can be easily contacted to inform them that their animal has been found

The chips cannot be used as a gps tracker for the animal, because that would require a constant power source. Therefore, the chip is only effective if the animal has been found by the right authorities. If not, the pet cannot be tracked and therefore cannot be identified

Current Methods

Excluding the existing apps which have been introduced, there is no real other way of being able to track a lost pet. For majority of people, the main way in which they would be able to find their pet if it ran away, would be through the creation of posters and then contacting their local rescue centre

Information displayed on the Manchester and Cheshire Dogs' Home really displays how difficult it can be in order to retrieve a lost animal:

"To increase the chances of finding your LOST DOG please ensure that you come to Manchester Dogs' Home on a regular basis to view our stray dogs.

A telephone call to us is not enough as collars & tags can fall off, descriptions may vary and occasionally a microchip CANNOT be found"

What has this shown?

Well, looking back at the competitor analysis, the best way in order to track an animal, would be the use of a gps tracker. However, owners (as well as the vets), would love to make an easier method of seeing the details of their pet (like FitBark), in easy to understand graphs that can be accessed.

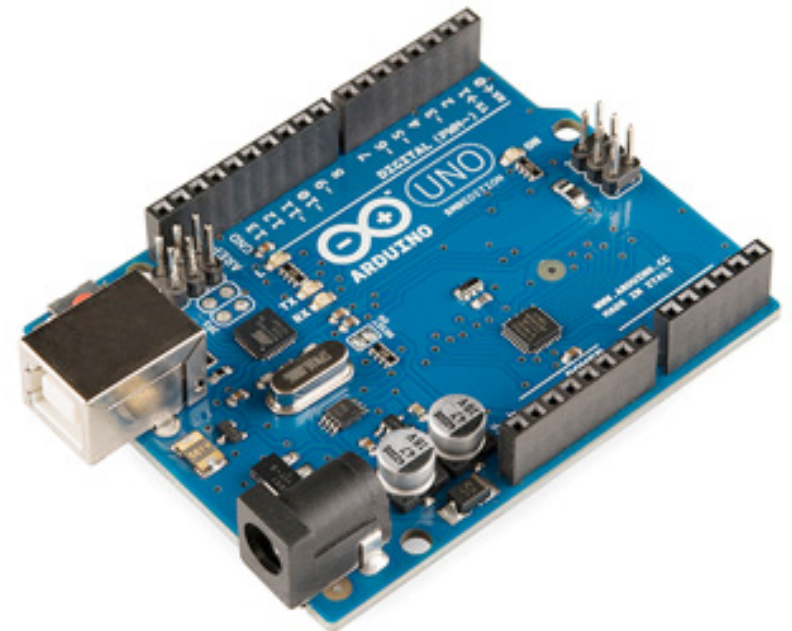
However, owners don't want to have to pay a ridiculous amount of money in order to make sure that they can track the location of their animal. A one off payment is more preferred, which means that the components of the collar would have to be relatively cheap or easy to maintain

From here, I will have to look into the correct hardware in which I could use in order to create a prototype

Arduino

After working with Arduino last year, I felt that this would be a great start in order to focus on the creation of the collar. The main components which are needed throughout the collars construction, would be much cheaper to obtain if I was to use Arduino. As well as this, being able to personally code each of the individual components so that they would be relevant, would also be a bonus.

Finally, there are a large number of examples which are out on the internet, that I could follow in the construction of the collar. This would make the making process of the prototype so much easier. However, the only issue that I do have with using Arduino, is that the size of the components may be too large to attach to an animal's collar. This could therefore be a very large issue in relation to the aesthetics and design of the collar.





Cooking Hacks

I decided to look into whether or not it would be possible to create an Arduino based GPS tracker. After searching the internet, I stumbled across Cooking Hacks and their version of a GPS+GPRS vehicle tracker.

This was a great starting point in order to find, as it would be able to give me an idea of whether or not the GPS would be

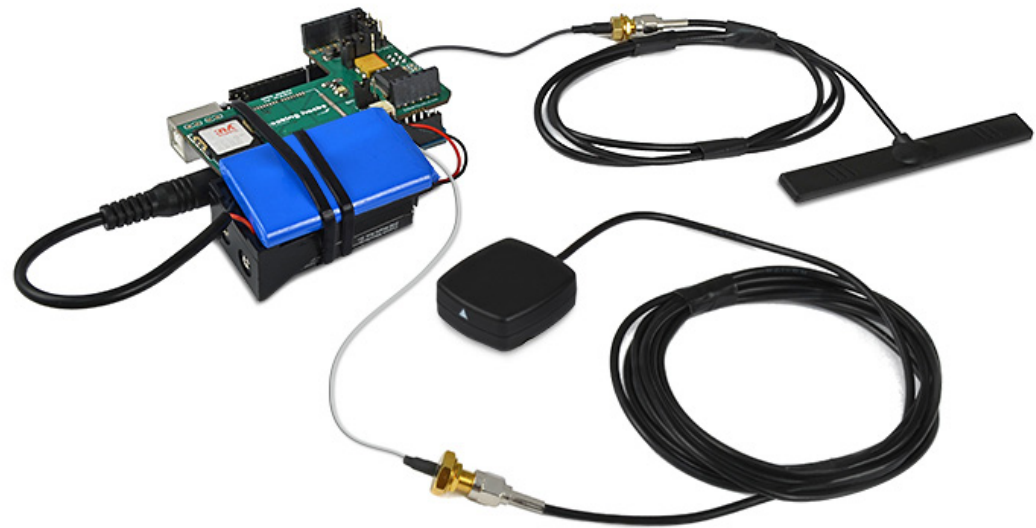
possible to make. As with the PawTrax S. The way in which this GPS tracker would work, would be through the use of an SMS in order to receive the longitude and latitude location of the car. The owner can then access google maps in order to see the journey in which the car has taken. As well as this, the information will only be sent to the correct mobile number (the one in which you have registered to the Arduino)

Cooking Hacks

In order to create the GPS tracker, a large number of components would have to be needed. They would be as follows:

- Arduino Uno
- Geolocation Tracker (GPS+GPRS) with SIM908
- External GPS Antenna
- External GPRS-GSM Antenna
- 2300mA/h Rechargeable Battery
- 9V Alkaline Battery
- 9V Battery Holder

After seeing the assembled product, I can already tell that this would not be possible to attach to a collar as the size of all of the components together is clearly too large. Also, I don't want to cause any upset to the animal



Cooking Hacks

However, even though the components are too large, the coding could become very useful for what I wish to use. I set about analysing the code, to see which areas could be more relevant to me. As well as this I discovered that the code was for free distribution, which it would be fine for me to incorporate into my work

I would mainly focus on using the GPS coding, as this would help in working out

how I would actually be able to track the animal. I also felt that a great bit of code for me to use, would be how to use the google maps feature in order to display where the animal has been. This would be using the php script in order to plug the google maps into my website, for the display of the animals movements to be clearly seen

```
/*
 * Description: This sketch gets the GPS coordinates and sends them
 * through HTTP if your phone number is correct.
 * Copyright (C) 2013 Libelium Comunicaciones Distribuidas S.L.
 * http://www.libelium.com
 *
 * This program is free software: you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation, either version 3 of the License or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program. If not, see .
 *
 * Version 0.1
 * Author: Marcos Martínez
 */
```

```
<?php
if (!empty($_GET['latitude']) && !empty($_GET['longitude']) &&
    !empty($_GET['time']) && !empty($_GET['satellites']) &&
    !empty($_GET['speedOTG']) && !empty($_GET['course'])) {

    function getParameter($par, $default = null){
        if (isset($_GET[$par]) && strlen($_GET[$par])) return $_GET[$par];
        elseif (isset($_POST[$par]) && strlen($_POST[$par])) {
            return $_POST[$par];
        } else return $default;
    }

    $file = 'gps.txt';
    $lat = getParameter("latitude");
    $lon = getParameter("longitude");
    $time = getParameter("time");
    $sat = getParameter("satellites");
    $speed = getParameter("speedOTG");
    $course = getParameter("course");
    $person = $lat.", ".$lon.", ".$time.", ".$sat.", ".$speed.", ".$course."\n";

    echo "
    DATA:\n
    Latitude: ".$lat."\n
    Longitude: ".$lon."\n
    Time: ".$time."\n
    Satellites: ".$sat."\n
    Speed OTG: ".$speed."\n
    Course: ".$course;

    if (!file_put_contents($file, $person, FILE_APPEND | LOCK_EX))
        echo "\n\t Error saving Data\n";
    else echo "\n\t Data Save\n";
}
else {
    ?>
```

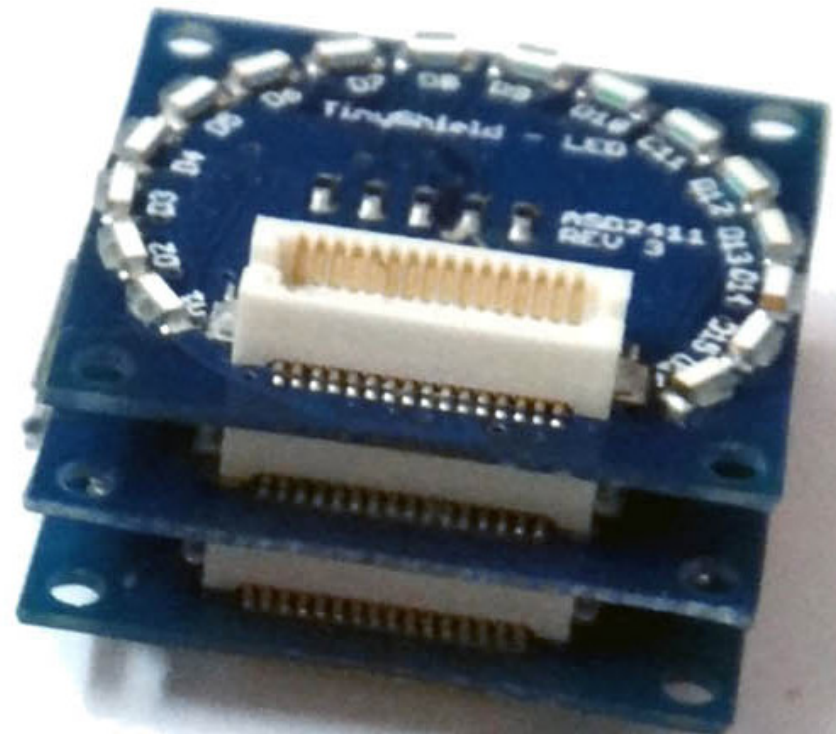
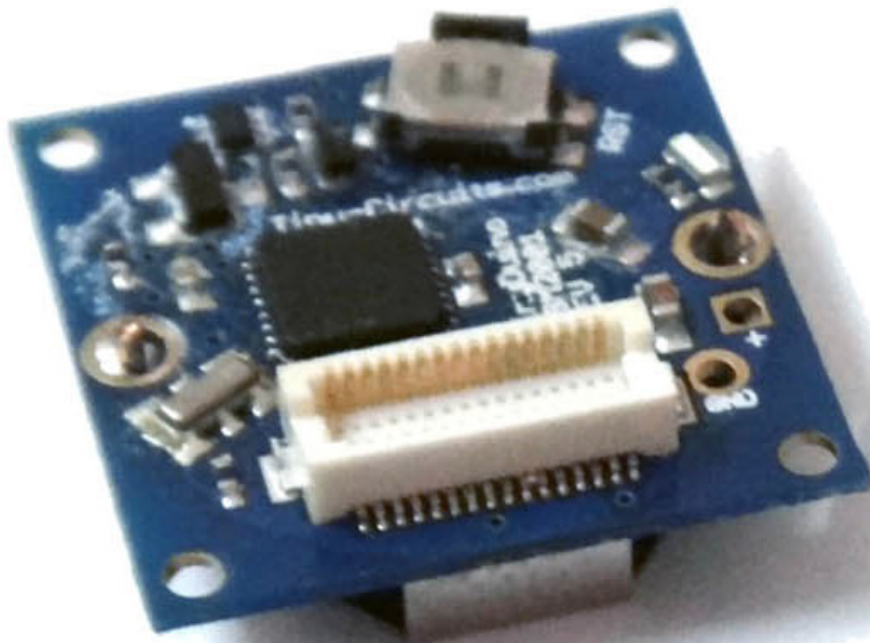
```
void send_HTTP(){
    uint8_t answer=0;
    // Initializes HTTP service
    answer = sendATcommand("AT+HTTPINIT", "OK", 10000);
    if (answer == 1)
    {
        // Sets CID parameter
        answer = sendATcommand("AT+HTTTPARA=\"CID\",1", "OK", 5000);
        if (answer == 1)
        {
            // Sets url
            sprintf(aux_str, "AT+HTTTPARA=\"URL\", \"http://192.168.1.100/demo_sim908.php?\"
            Serial.print(aux_str);
            sprintf(frame, "visor=false&latitude=%s&longitude=%s&altitude=%s&date=%s&satellites=%s&speedOTG=%s&course=%s",
            latitude, longitude, altitude, date, satellites, speedOTG, course);
            Serial.print(frame);
            answer = sendATcommand("", "OK", 5000);
            if (answer == 1)
            {
                // Starts GET action
                answer = sendATcommand("AT+HTTPACTION=0", "HTTPACTION:0,200",
                if (answer == 1)
                {
                    Serial.println(F("Done!"));
                }
                else
                {
                    Serial.println(F("Error getting url"));
                }
            }
            else
            {
                Serial.println(F("Error setting the url"));
            }
        }
        else
        {
            Serial.println(F("Error setting the CID"));
        }
    }
}
```


TinyDuino

After coming to the conclusion that arduino components would be too large for the use of this collar, I stumbled across something called TinyDuino. TinyDuino was a kickstarter project which began in September 2012. The project itself was campaigned by Ken Burns, who had the same issue which I was having. That majority of projects now needed smaller components than those used by Arduino.

Therefore, in order to resolve this issue, Ken decided to create TinyDuino boards.

The boards themselves are no bigger than a quarter (20mm x 20mm), and are fully capable of running Arduino and LilyPad. They also maintain the power and functionality of the normal sized Arduino boards, including shield support.



TinyDuino

The boards themselves also have the advantage of being extremely easy to assemble. Instead of having to connect each individual component with wires, the boards simply 'click' together. This would make the construction of the collar extremely easy as I would simply have to click all of the pieces together.

Due to this, once fully constructed, they maintain their small size and can therefore be more successful being used in the collar. This reason alone, makes me feel that the TinyDuino components would be best suited for the creation of the dog collar



Who?

So who will be using the tracking collar?

Well, as with my original concept, I created PawPrint for the owners and vets to keep in touch easily for the benefit of the animals. Therefore, through the use of the updated brand designs (app and website), both the vet and the owner will be able to receive the information sent

Why will they be using it? It will allow pet owners and vets to keep in contact with one another in order to help the animal out as much as possible. As well as this, there are no other apps that focus on both the vet and owner, and will therefore be very beneficial for both parties

User Persona

Steven Briggs (67)
Taxi Driver

"I'm getting on quite a bit now, and I want to make sure my little guy is best looked after"

Personal Information

Steven is a 67 year old widow, who's main companion is his faithful 12 year old golden retriever. After his wife passed away around 5 years ago, Steven likes to stay active (which is why he is still working), and takes his dog for long walks

Issues

As Steven is getting rather old now, he cannot always keep up with his dog when they go for their walks. He is afraid that if his dog runs too far down the path, he will not be able to catch up

How will the collar help?

By having his dog wear the tracking collar, Steven will be able to keep an eye of where his dog is whilst on the long walks. As well as this, if his dog happens to run away, he will be able to locate the whereabouts of his dog nearly straight away



User Persona

Lisa Beesley (32)
Vet

"I've always loved animals, and want to do everything I can to help them out"

Personal Information

Lisa is a 32 year old vet, who is an owner of a dog herself. She can understand how hard it is for pet owners to keep track of their animals. She wants to make sure that all pets are well looked after, and owners aren't having any trouble caring for them

Issues

Sometimes when pet owners come into the veterinary, they don't fully remember where their pet has been. This can cause issues if tics have been found on the animal

How will the collar help?

If pet owners have their animals wear the tracking collar, the vets (Lisa) will be able to see where their animal has been. This can therefore assist in determining where the animal could have picked up certain illnesses/tics



CONCEPT

Initially, I will be focusing more towards the construction of the prototype dog collar, before any developments to the brand design are made

Tracking Collar

The tracking collar itself will have three main purposes throughout its design. Obviously, the first and most important would be to be able to use a GPS tracking system in order to keep track of the animal. This will be through the incorporation of a TinyDuino GPS shield in the collar

Secondly will be how the activity will be recorded. An accelerometer would have to be added to the main structure of the components, in order to record this information. This is to make sure that the collar coincides with the previous designs from the PawPrint brand

Finally, the information recorded will have to be sent directly to a computer wirelessly. This can be done through the adding of a Wifi shield in the collar's components. This will then be displayed through the use of Google maps onto the vet website, and then hopefully onto the app

Branding

After gaining my feedback from the PawPrint branding last year, I will have to make changes to mainly the design of the PawPrint vet website. This is to make sure that the design appears to be much more professional and relates to the specific audience it is for (i.e. the vets)

As well as this, I will try and make sure that the website is fully functioning in order for the information recorded from the collar to be displayed. Other branding in relation to the collar will be completed towards the end of the project (after the main components have been assembled and are all functioning correctly).

This will include the main case of the product and any extra promotional products for the brand (i.e. water bowl and food tray and possibly a lead for dogs)

COMPONENTS

After looking into the different technologies I could use, I had to determine which components I would need

TinyDuino GPS Car Tracker

Before I began the main designs of the tracking collar, I had another look for any inspiration towards the components that I would potentially have to use. I came across an example GPS car tracker that proved to be very useful. One Arduino tutorial was a TinyDuino based GPS Car Tracker with SD card.

What this did, was use the GPS shield in order to track the locations of the car, and record the co-ordinates onto an SD card. This essentially is the idea of the tracking collar, however, I would need for the information recorded to be transmitted directly to the computer/website

I would therefore have to look further into a method in which I would be able to transmit this information. An initial thought was towards using a Wifi shield to try and resolve this problem

Existing Code

Even though the incorporation of the SD card would not be relevant towards my project, the use of the GPS shield is near enough vital. Therefore, because of this reason, I felt that it could prove to be rather useful if I was to examine the coding. This would be much more appropriate than the code in which I had taken from the Cooking Hacks tracker.

I would have to try and take the existing code from this example, and attempt to apply the GPS tracking to the requirements of my tracking collar. This could be done through the removal of the SD card section of the code, and using the Wifi shield in order to transfer the data

```
/* This Arduino sketch will log GPS NMEA data to a SD card */

#include
#include

// The Arduino pins used by the GPS module
static const int GPS_ONOFFPin = A3;
static const int GPS_SYSONPin = A2;
static const int GPS_RXPin = A1;
static const int GPS_TXPin = A0;
static const int GPSBaud = 9600;
static const int chipSelect = 10;

// The GPS connection is attached with a software serial
SoftwareSerial Gps_serial(GPS_RXPin, GPS_TXPin);

void setup()
{
    // Init the GPS Module to wake mode
    pinMode(GPS_SYSONPin, INPUT);
    pinMode(GPS_ONOFFPin, OUTPUT);
    digitalWrite( GPS_ONOFFPin, LOW );
    delay(5);
    if( digitalRead( GPS_SYSONPin ) == LOW )
    {
        // Need to wake the module
        digitalWrite( GPS_ONOFFPin, HIGH );
        delay(5);
        digitalWrite( GPS_ONOFFPin, LOW );
    }

    // Open the debug serial port at 9600
    Serial.begin(9600);
}
```

Components

After analysing the TinyDuino GPS car tracker, I constructed a list of potential components that I would need in order to construct the tracking collar. Obviously, as my concept is on the same lines as the GPS car tracker, I would need near enough all of the same components (except the SD card). I would also need a few extra features in order to make sure that the collar would do what I would require it to.:

- Processor board with Battery holder
- Accelerometer
- GPS Shield
- Wifi Shield
- USB & ICP Board (in order to program the components)

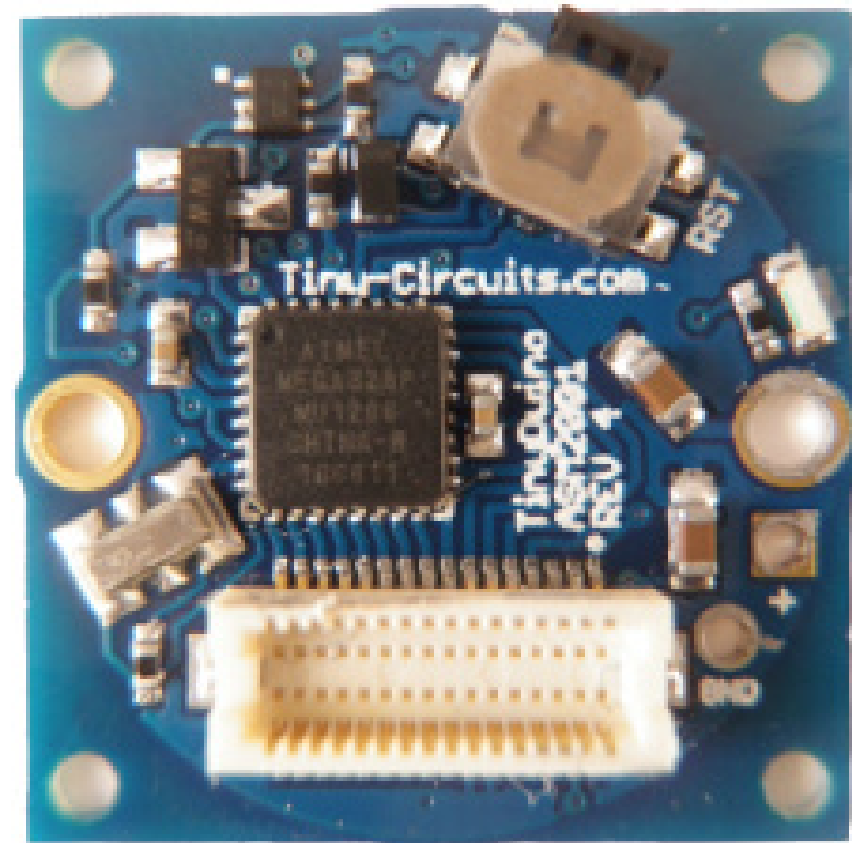
I then found a company called iCSAT, who distributed TinyDuino to the UK. They are Yorkshire based and I felt that this would be easier to order from (as I would receive the products quicker, than if they came from America)

Processor Board

I decided to do some more research into each of the individual components to see whether or not they had any specific requirements. The processor board itself was pretty basic, in the fact that it ran the same as an Arduino Uno Board.

However, the main thing I would have to focus upon, is that the board does not include a voltage regulator. This means that I would have to make sure that I don't supply more than +5.5V to VBATT or the +5V signal, or would cause permanent damage to the board.

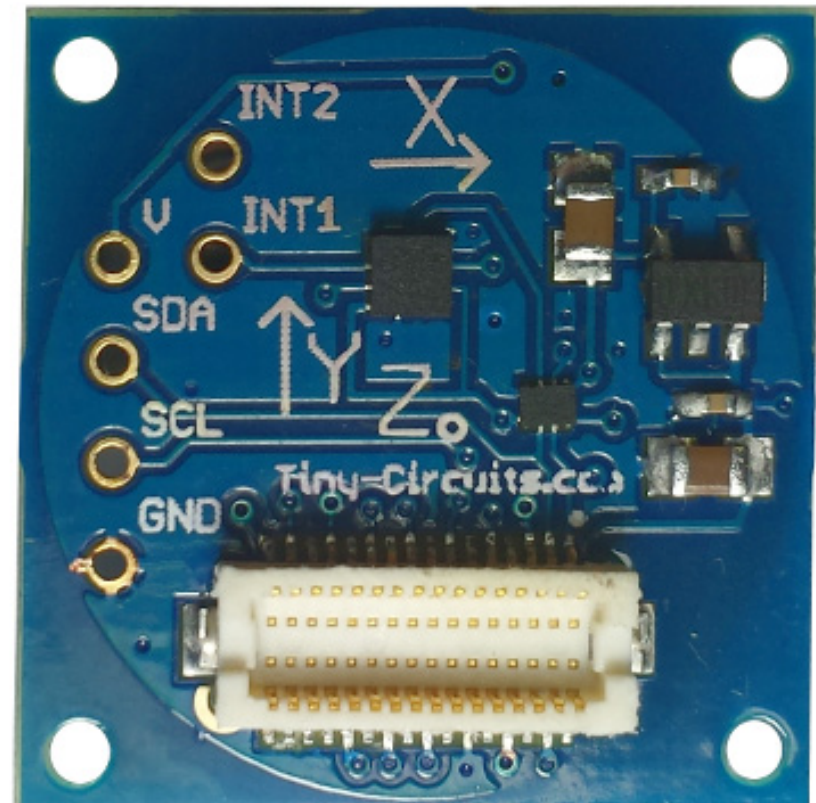
I would also have to make sure, that when uploading code to an Arduino Pro Mini requirement



Accelerometer

The TinyDuino Accelerometer contains a high performance and low power Bosch BMA250 3-axis accelerometer. This allows the measurement of accelerations in 3 perpendicular axes and therefore will allow the measurement of tilt, motion, shock and vibration.

It is designed to run for 1.8V, but with the inclusion of level shifters and a local power supply, the accelerometer can operate safely up to the 5V maximum of the TinyDuino Processor. An example code in order to test the accelerometer is given on the TinyCircuits website (main designers and manufacturers of TinyDuino)

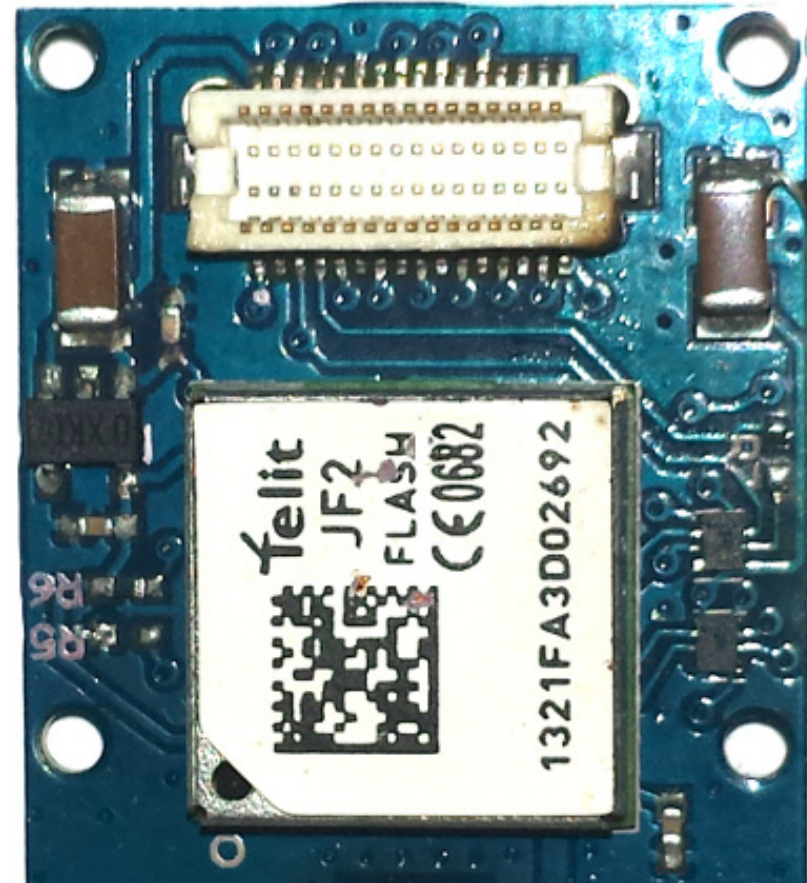


GPS Shield

The GPS shield is based upon the Telit JF2 GPS Module, which uses the popular SiRFstar IV chipset. The board itself is integrated with a GPS antenna to make sure that the size of the board remains small (no external antenna needed).

However, with the antenna being so small, the GPS will only be picked up outdoors. It has stated that in order to make sure the best results are found, I will have to make sure that I keep all metal away from the antenna, and make sure that it is used in an outdoor environment.

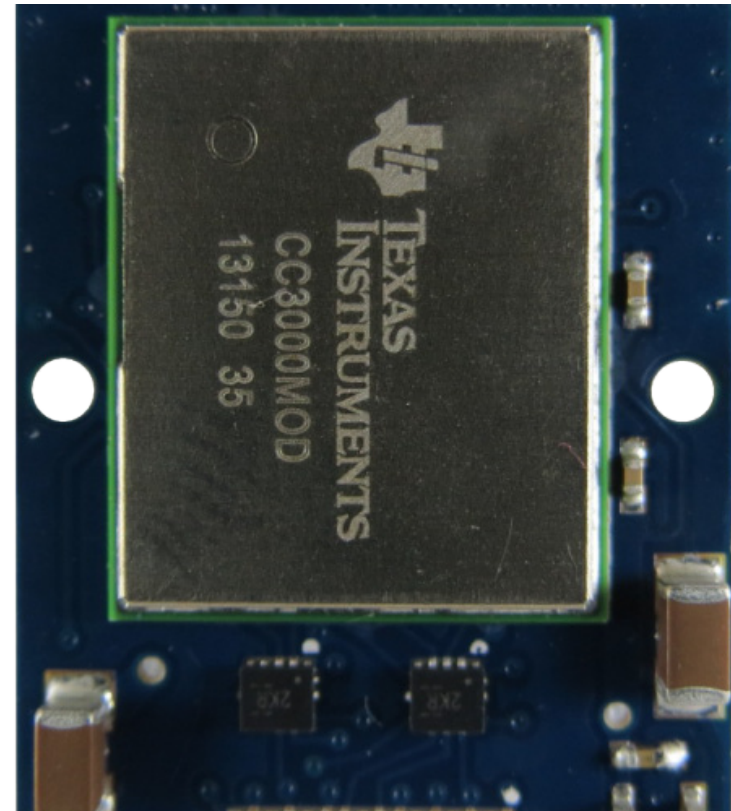
There is also example coding available in the GPS car tracker to ease my use of the product



Wifi Shield

The Wifi Shield is based upon the TI CC3000 Wifi Module, which supports 802.11b/g, different security modes (None, WEP, WPA and WPA2) and has a built in TCP/IP Stack that supports up to 4 concurrent sockets. Due to the amount of features the shield has, it draws up to 275mA which means a coin cell battery isn't strong enough. This means that I will have to attach an external battery pack, in order to power all of the components.

There is a large amount of example code available in order to test out the shield, which can come extremely useful in order to code the collar. This includes the pin numbers in which I will have to change for it to work

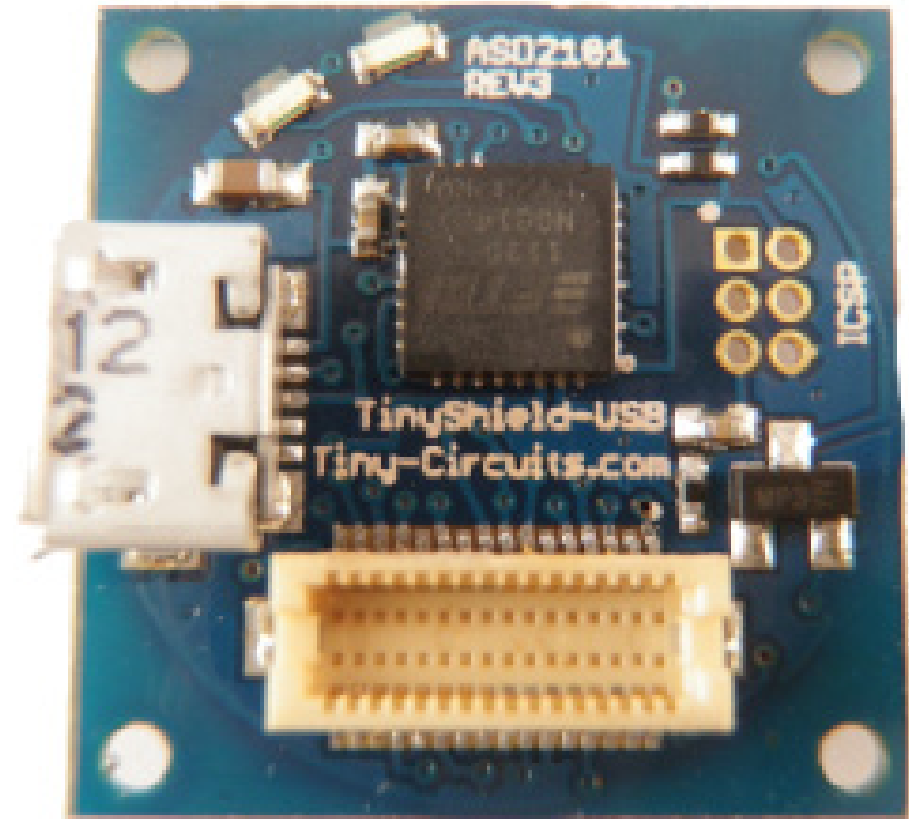


USB & ICP Board

The USB & ICP Board is a very simple component for me to use. This is basically an extra attachment which will enable me to program all of the other components (as this will be how I connect them to the computer).

The cable needed to connect to the USB & ICP Board is the same that is used most Samsung phones. Luckily I was already in the possession of this, so that would reduce anymore potential costs to the project.

The other factor that I had to make sure I remembered, was that the Processor Board would not operate by itself with the USB & ICP Board attached. Therefore, when running on the collar, this would be removed



Components Order

However, after ordering the components, I encountered an issue with the payments. This halted my production of the prototype as the main payment (£122) had been taken from my account, but the products had not been processed.

I contacted the main director (Brian Smith) in order to resolve the issue. After a few discussions over email and phone, we

managed to sort out the billing problem. As an apology for the delay and problems caused, Brian very kindly sent the items up through special delivery so I could try and catch up on time lost

4	✓	1 piece(s)	TD0021	TinyShield WIFI	£38.00	0%	£38.00
5	✓	1 piece(s)	TD0004	TinyShield USB & ICP	£12.50	0%	£12.50
6		Subtotal					£114.00
7		Delivery method		Standard delivery			£8.00
8		Payment method		PayPal Express Checkout			
9		Tax area		EU country			
10		Total amount					£122.00
11		Total amount (without VAT)					£120.67
12		Including VAT of: 20 %					£1.33

PAID

BL BACS 02/03/15

THANKS

BR

Good luck with your project

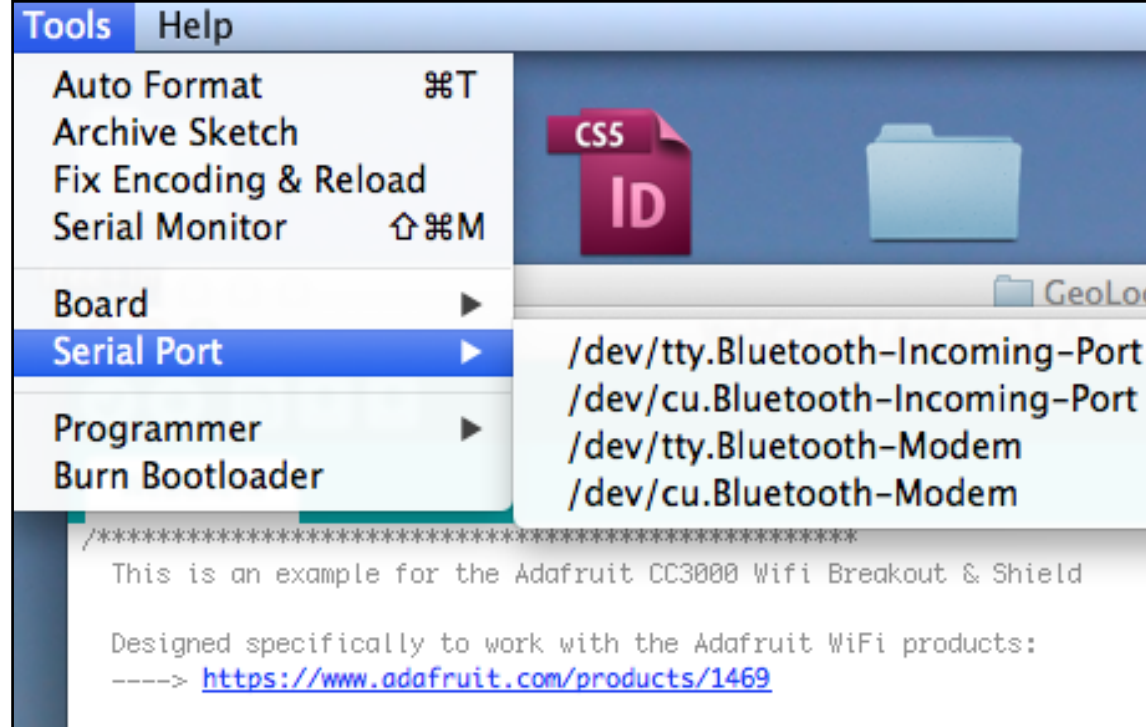
CODING

Before creating the main case for the collar, I will have to make sure that the technical side is completed and fully working

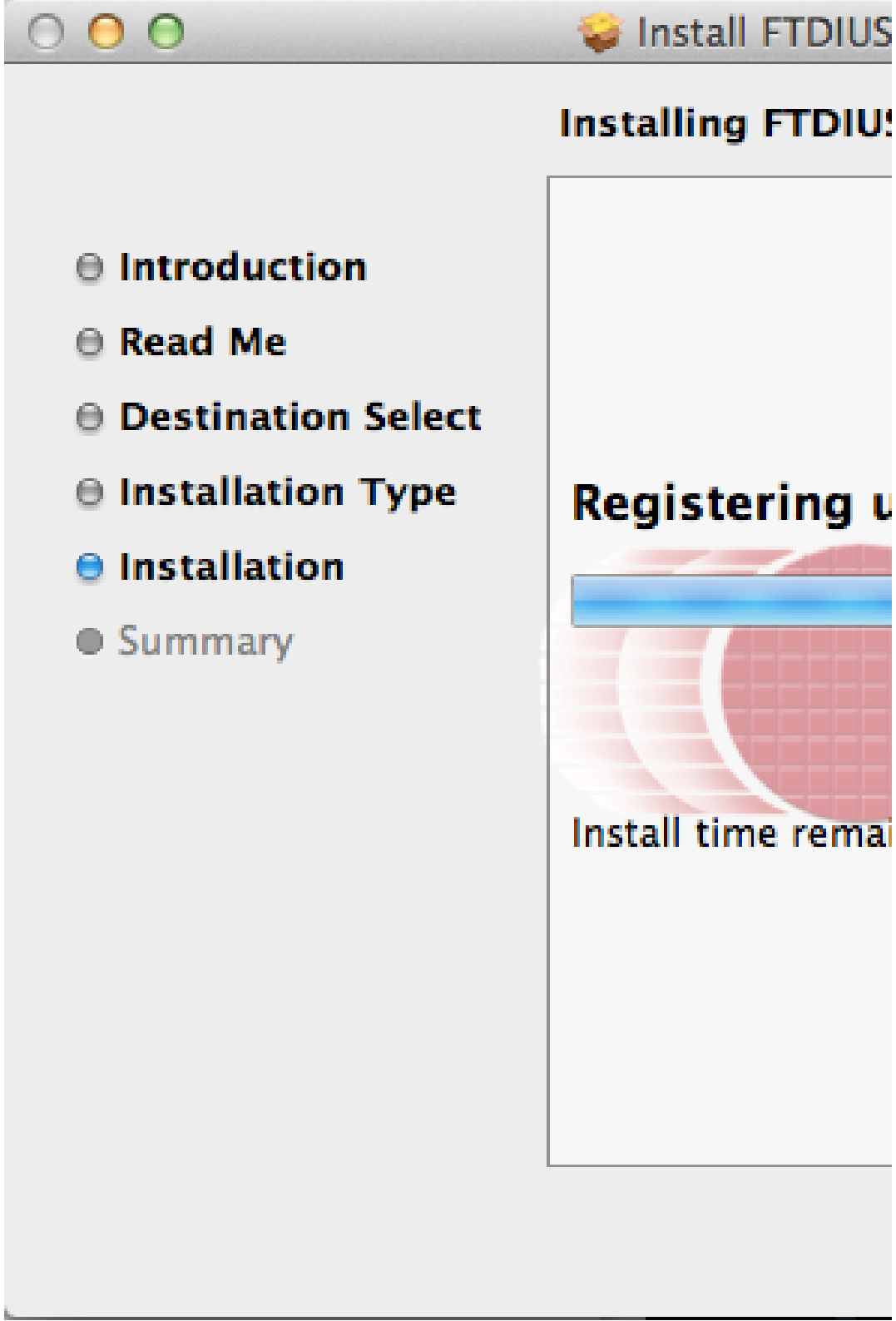
Connection Issue

Initially, when I first attempted to code the components, there was a connection error. Basically the board itself was not registering on my mac. This meant that I was not able to upload any code onto the Processor board meaning it wouldn't do anything.

After referring back to the specifications on the board, I noticed that FTDI was required in order to upload any code onto the board. This required me to download the specified FTDI online which was compatible with my Mac OS X. I decided to go for the updated version of the software in the hope that any previous bugs/errors with the previous had been removed



		Processor Architecture		
Operating System	Release Date	x86 (32-bit)	x64 (64-bit)	PPC
Windows*	2014-09-29	Available as setup executable Contact support1@ftdichip.com if looking to create customised drivers		-
Linux	2009-05-14	1.5.0	1.5.0	-
Mac OS X	2012-08-10	2.2.18	2.2.18	2.2.18
Windows CE 4.2-5.2**	2012-01-06	1.1.0.20	-	-
Windows CE 6.0/7.0	2012-01-06	1.1.0.20 CE 6.0 CAT CE 7.0 CAT	-	-
Windows CE 2013	2015-03-06	BETA		



FTDI

After downloading the FTDI software, I had to make sure that it was fully installed in order to upload the code onto the Processor Board. The software itself was rather self explanatory, simply stating that it was the format towards which the TinyDuino would be able to understand (and therefore be registered by my computer).

After downloading and installing the software, the serial port became immediately recognisable and I was able to start testing each of the components to see if the example code would work properly

Wifi Code

One of the example codes in which the Wifi Shield used is the WebClient example. This basically allows the Wifi Shield to access a website/page to test the connection. After working alongside Alistair, I found that the main area of code in which I will need to use for the Wifi Shield to work is the highlighted section to the right.

In relation to all of the individual pieces of the code, I have gained an understanding of what each section refers to:

DCHP - asks the network for an IP address and the router will return one to the Wifi Shield

Web Address - displays the information and remembers the PHP



```
WebClient | Arduino 1.0.5

Note: HTTP/1.1 protocol is used to keep the server from closing the
*/
Adafruit_CC3000_Client www = cc3000.connectTCP(ip, 80);
if (www.connected()) {
  www.fastrprint(F("GET "));
  www.fastrprint(WEBPAGE);
  www.fastrprint(F(" HTTP/1.1\r\n"));
  www.fastrprint(F("Host: ")); www.fastrprint(WEBBSITE); www.fastrprint(
  www.fastrprint(F("\r\n"));
  www.println();
} else {
  Serial.println(F("Connection failed"));
  return;
}

Serial.println(F("-----"));

/* Read data until either the connection is closed, or the idle timeout
unsigned long lastRead = millis();
while (www.connected() && (millis() - lastRead < IDLE_TIMEOUT_MS)) {
  while (www.available()) {
    char c = www.read();
    Serial.print(c);
    lastRead = millis();
  }
}
www.close();
```

127 - 151 Arduino Pro or Pro Mini (3.3V, 8 MHz) w/ ATmega328 on /dev/cu.usbmodem

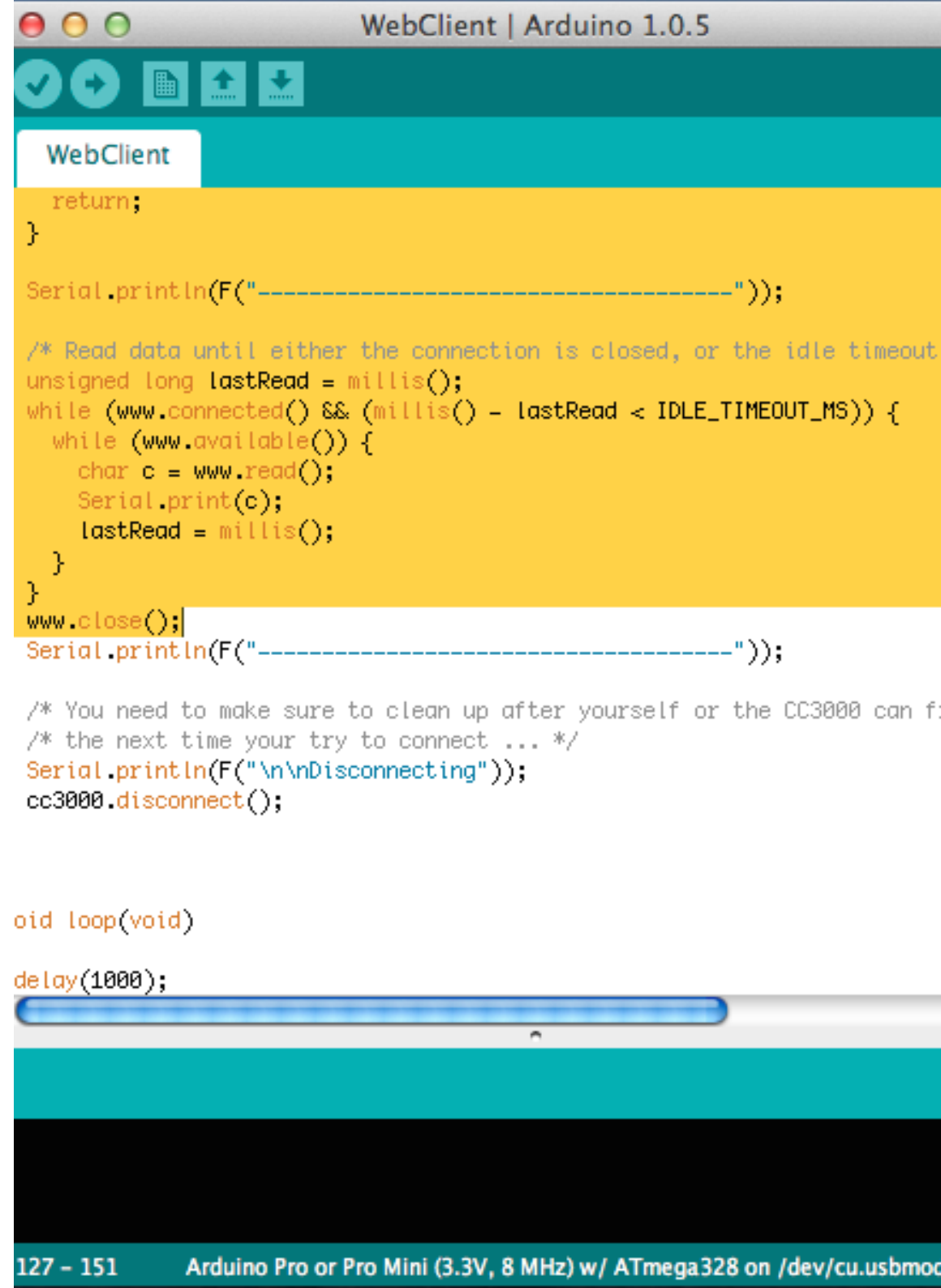
Wifi Code Cont.

The host is found by its name through the IP address and then the DNS server. The IP address doesn't know the page in which the Wifi shield is looking for, and therefore sends the host to confirm if the page is correct.

The "Print" section of the code, is the information that will be displayed on the serial monitor. The "Get" is the website in which the Wifi shield is looking for, and the "Webpage" is the '/example' section of the web address.

In terms of encryption, the Wifi is encrypted to the device itself, whereas the https is encrypted to the server used.

Finally, the 'millis' check the time in which the Wifi shield is receiving something. If the connection times out, the process will stop to save memory. If there remains a connection, it will continue as normal



```
return;
}

Serial.println(F("-----"));

/* Read data until either the connection is closed, or the idle timeout
unsigned long lastRead = millis();
while (www.connected() && (millis() - lastRead < IDLE_TIMEOUT_MS)) {
  while (www.available()) {
    char c = www.read();
    Serial.print(c);
    lastRead = millis();
  }
}
www.close();
Serial.println(F("-----"));

/* You need to make sure to clean up after yourself or the CC3000 can f
/* the next time your try to connect ... */
Serial.println(F("\n\nDisconnecting"));
cc3000.disconnect();

oid loop(void)

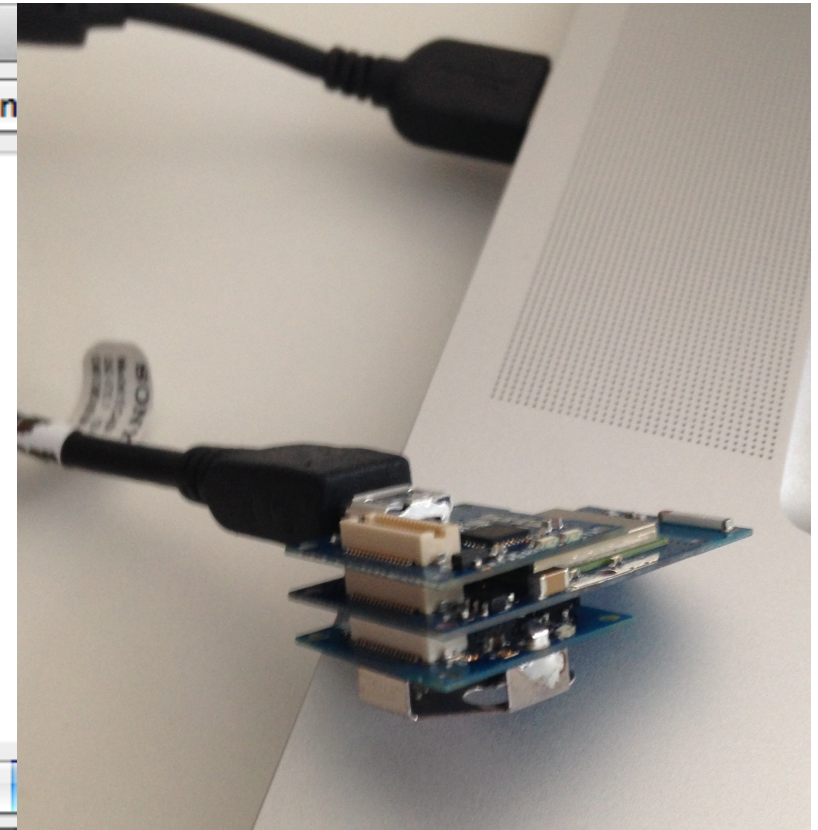
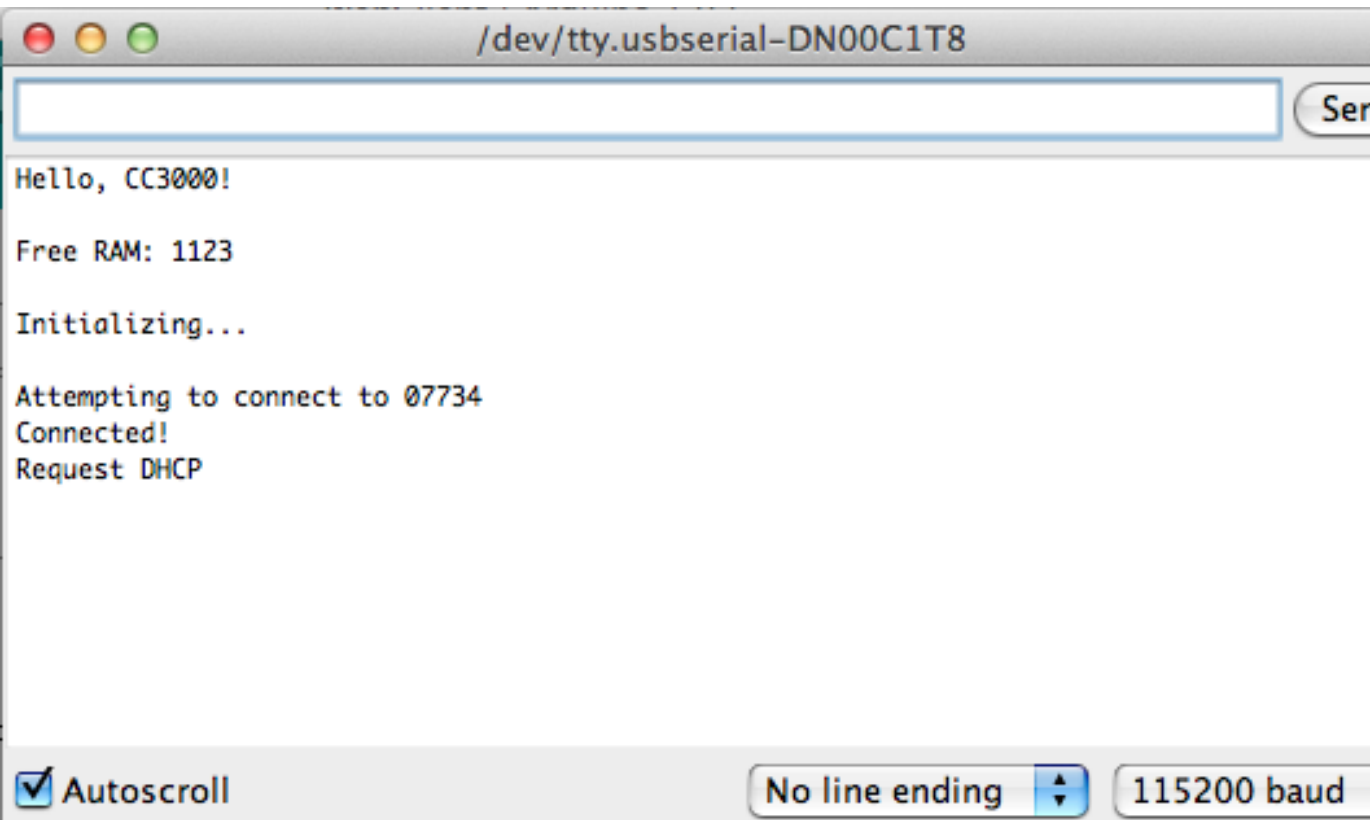
delay(1000);
```

127 - 151 Arduino Pro or Pro Mini (3.3V, 8 MHz) w/ ATmega328 on /dev/cu.usbmodem

Wifi Test

To test whether or not the Wifi Shield was working, I compiled all of the required components. This consisted of the Wifi Shield, Processor Board and USB & ICP Board. Once all were clicked together (making sure that the USB & ICP Board was positioned at the top), I connected them to my mac and uploaded the Arduino code. The way in which I was able to see whether or not the connection had been successful,

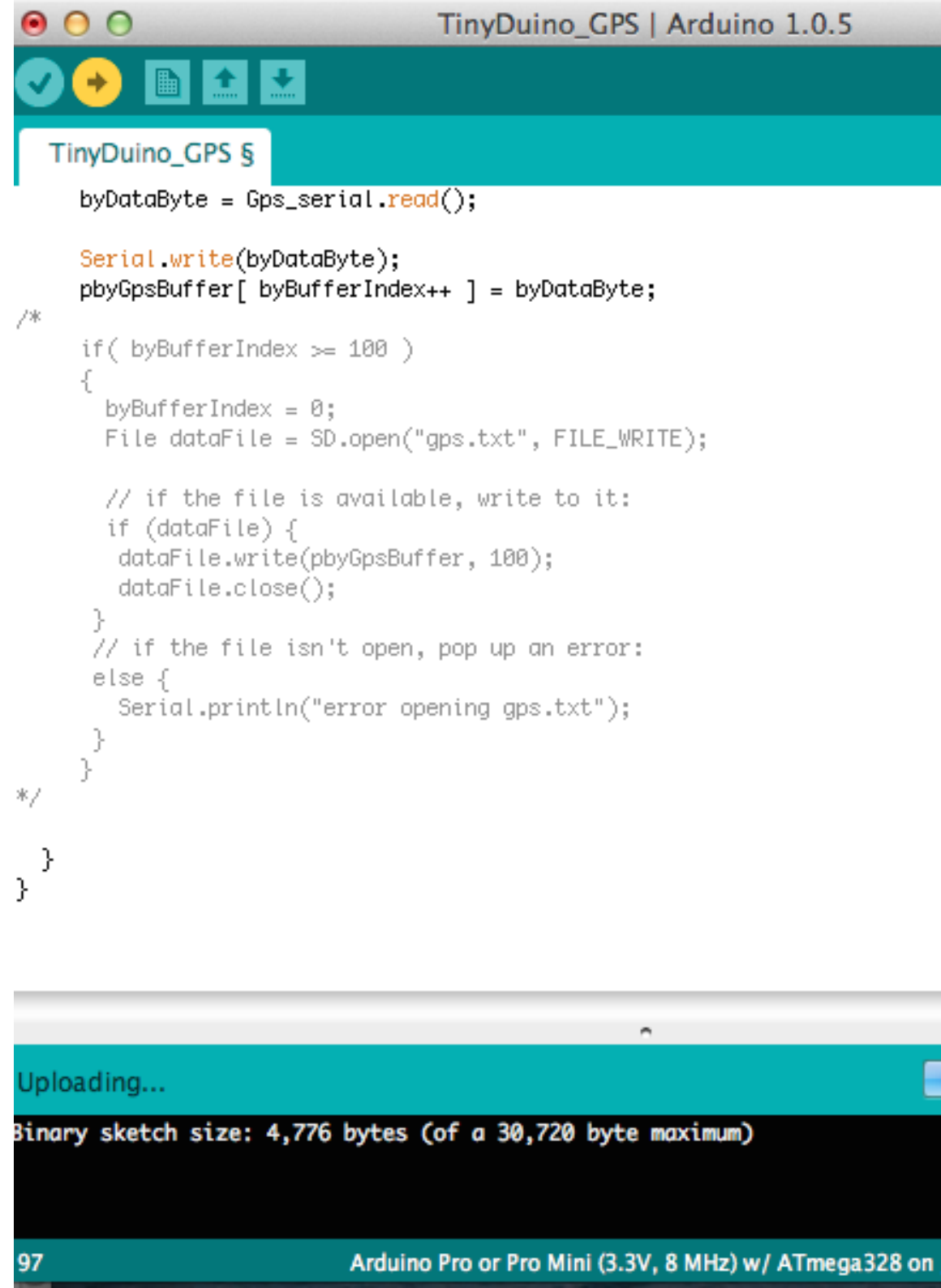
was to click on the option 'serial monitor'. This would display the process in which the Wifi Shield would be completing (in the printed format set in the code), and if it had managed to find the preset webpage. Fortunately, the connection was successful which meant that the Wifi Shield was working correctly. I could now move on to the GPS Shield and see or not that would function



GPS SD Removed

To make sure that the GPS would work correctly, I would have to edit the existing example code in which I had found. I made sure that the removal of the SD components code had been completed (this is shown by changing the code into annotations on the Arduino software).

What this meant was, that instead of all of the co-ordinates (longitude and latitude) onto the SD card, they will just be displayed on the serial monitor. I then had to upload the code onto the GPS Shield and test it to determine if it was functioning properly



```
TinyDuino_GPS §
byDataByte = Gps_serial.read();

Serial.write(byDataByte);
pbyGpsBuffer[ byBufferIndex++ ] = byDataByte;
/*
if( byBufferIndex >= 100 )
{
  byBufferIndex = 0;
  File dataFile = SD.open("gps.txt", FILE_WRITE);

  // if the file is available, write to it:
  if (dataFile) {
    dataFile.write(pbyGpsBuffer, 100);
    dataFile.close();
  }
  // if the file isn't open, pop up an error:
  else {
    Serial.println("error opening gps.txt");
  }
}
*/
}
}
```

Uploading...

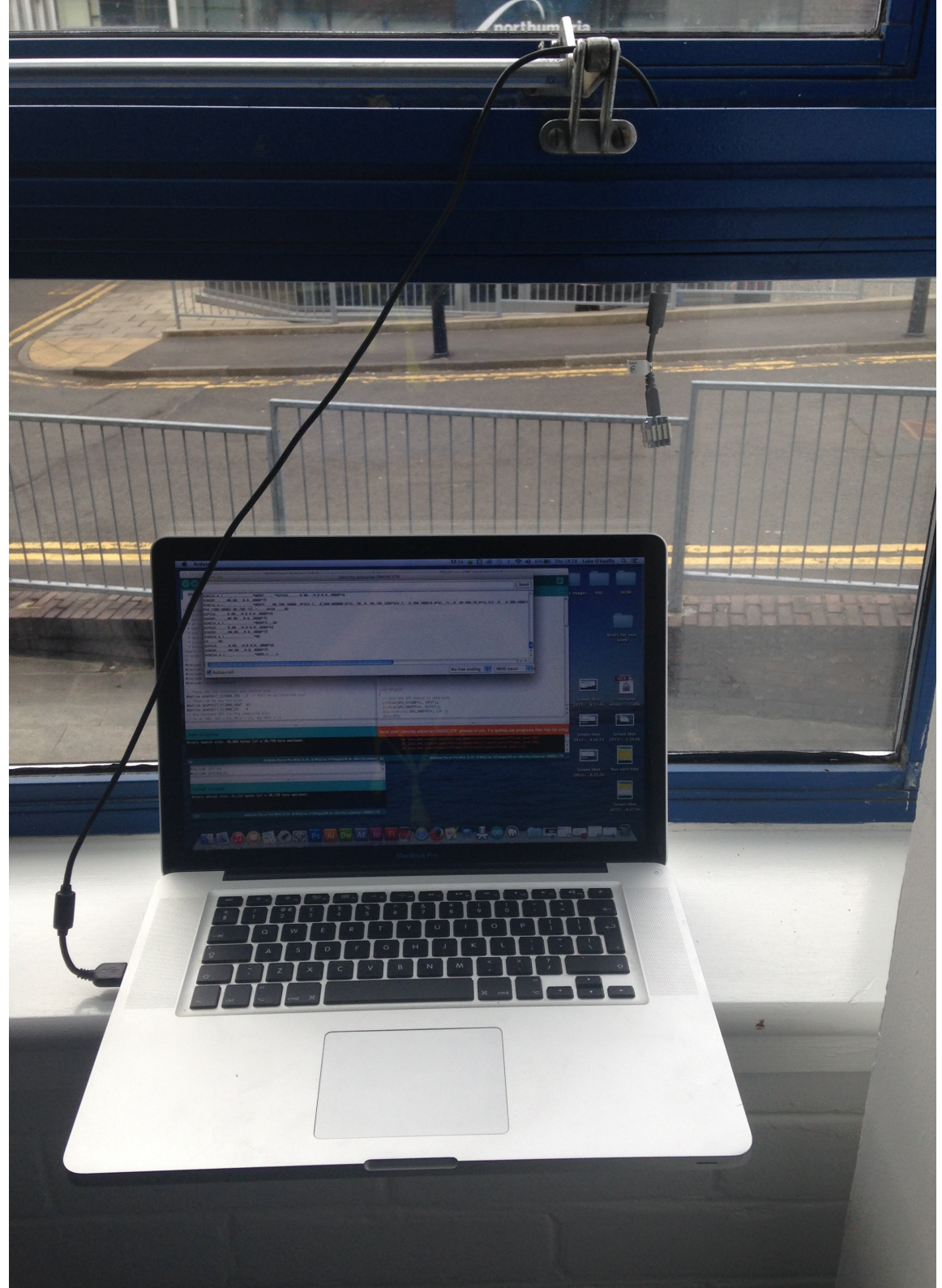
Binary sketch size: 4,776 bytes (of a 30,720 byte maximum)

97 Arduino Pro or Pro Mini (3.3V, 8 MHz) w/ ATmega328 on

GPS Test

After attempting the GPS initially in the design room, we found that there was no connection. This was because, since the GPS antenna on the device is so small, its best place to work would be outside. Due to this, we decided to test what the success rate would be if we simply hung the device out of the window.

This was mainly to make sure that the device could transmit to potential satellites so they could locate it. However, even after attempting this for over half an hour, there was still no reading being shown onto the serial monitor. It was most likely due to the building blocking the signal, so the test would have to be done outside





GPS Test Cont.

After moving outside to a slightly more open environment, we began the test for a second time. As there are less obstacles for the transmissions to pass through, this would hopefully mean that the location would be displayed.

Again, for the GPS to begin working, we had to wait for roughly around 15-20 minutes. However, this time the co-ordinates began to be displayed. To test the accuracy of the GPS, I inputted the co-ordinates (after some working out) into google maps. The readings were very accurate which was great news in terms of how the collar would work in the future

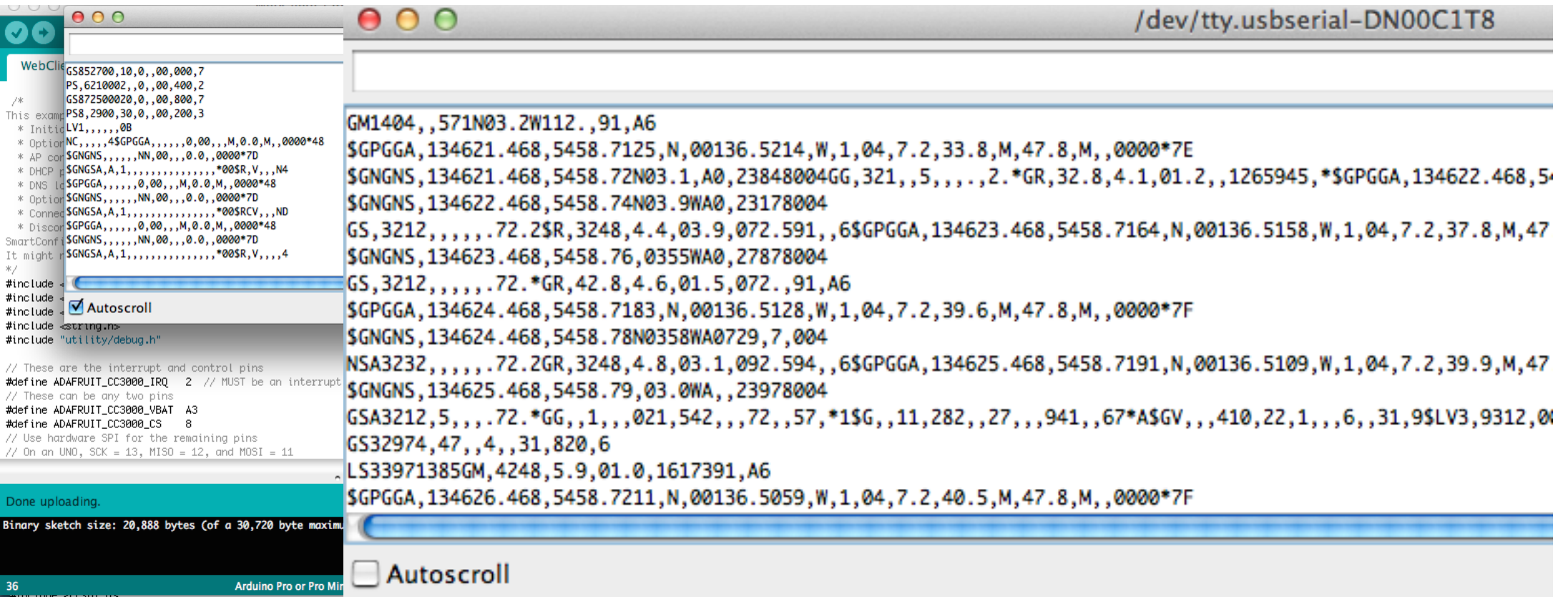
GPS Serial Monitor

The readings that are presented on the serial monitor are rather complicated to understand. This mainly because they are show as random letters and numbers which can be very confusing to what the readings actually mean.

Due to this, I had to conduct some research towards what each individual piece was valuable, and what it actually meant. The

results were as follows:

- The first set of number following \$GPWGA can be worked out as the current time
- The next set of digits represent the latitude and its direction (either north or south of the equator)
- Following this is the longitude and whether it is east or west



GPS Workings

Even though I know what the code now represents in the serial monitor, I still have to conduct a few working in order to gain the exact co-ordinates. The calculation are as follows:

Latitude

1st two characters + (remaining characters / 60).
If the outcome is positive, the direction is north.
Therefore meaning a negative outcome will be south

Longitude

1st three characters + (remaining characters / 60).
If the outcome is positive, the direction is east.
Therefore meaning a negative outcome will be west

The most important NMEA sentences include the GGA which provides

GGA - essential fix data which provide 3D location and accuracy data.

```
$GPGGA,123519,4807.038,N,01131.000,E,1,08,0.9,545.4,M,46.
```

Where:

GGA	Global Positioning System Fix Data
123519	Fix taken at 12:35:19 UTC
4807.038,N	Latitude 48 deg 07.038' N
01131.000,E	Longitude 11 deg 31.000' E
1	Fix quality: 0 = invalid 1 = GPS fix (SPS) 2 = DGPS fix 3 = PPS fix 4 = Real Time Kinematic 5 = Float RTK 6 = estimated (dead reckoning) 7 = Manual input mode 8 = Simulation mode
08	Number of satellites being tracked
0.9	Horizontal dilution of position
545.4,M	Altitude, Meters, above mean sea level
46.9,M	Height of geoid (mean sea level) above WGS84 ellipsoid
(empty field)	time in seconds since last DGPS update
(empty field)	DGPS station ID number
*47	the checksum data, always begins with *

If the height of geoid is missing then the altitude should be suspect. So all. This is the only sentence that reports altitude.

GSA - GPS DOP and active satellites. This sentence provides details on indication of the effect of satellite geometry on the accuracy of the fix. overdetermined solutions it is possible to see numbers below 1.0.

There are differences in the way the PRN's are presented which can effectively be scattered indicating that the almanac would show satellites in the null field all stacked up at the end. This difference accounts for some without regard to their use as part of the solution but this is non-standard

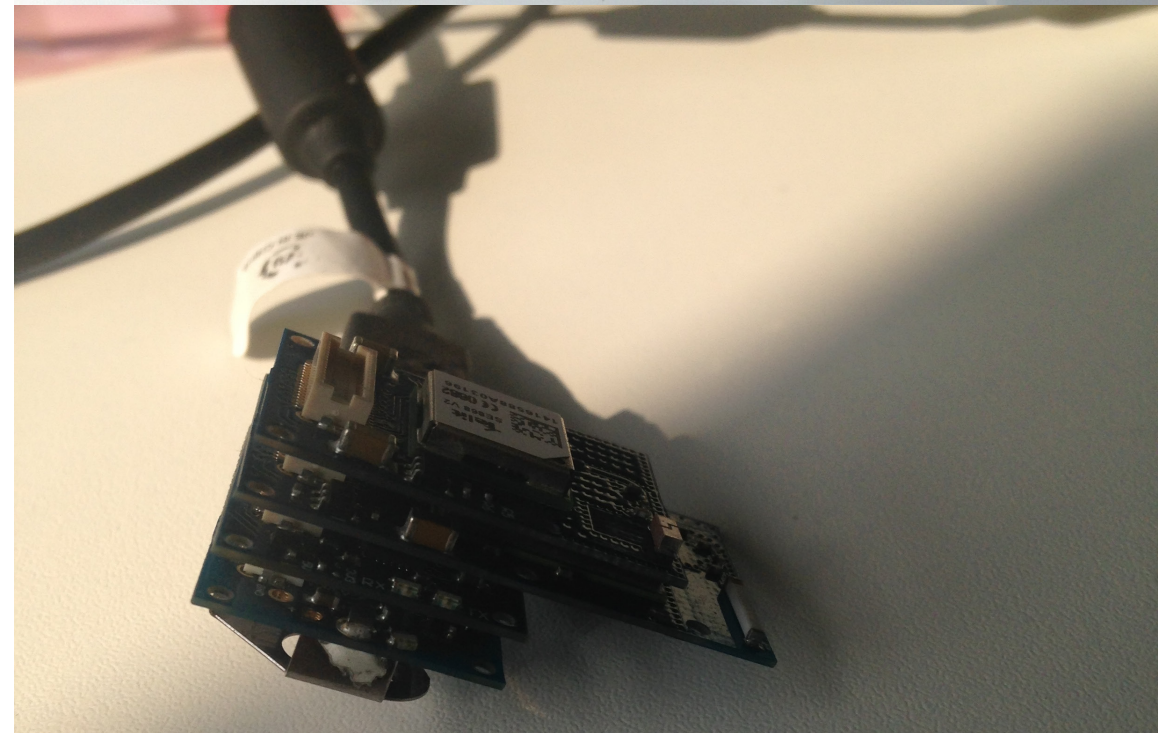
COMBINING CODES

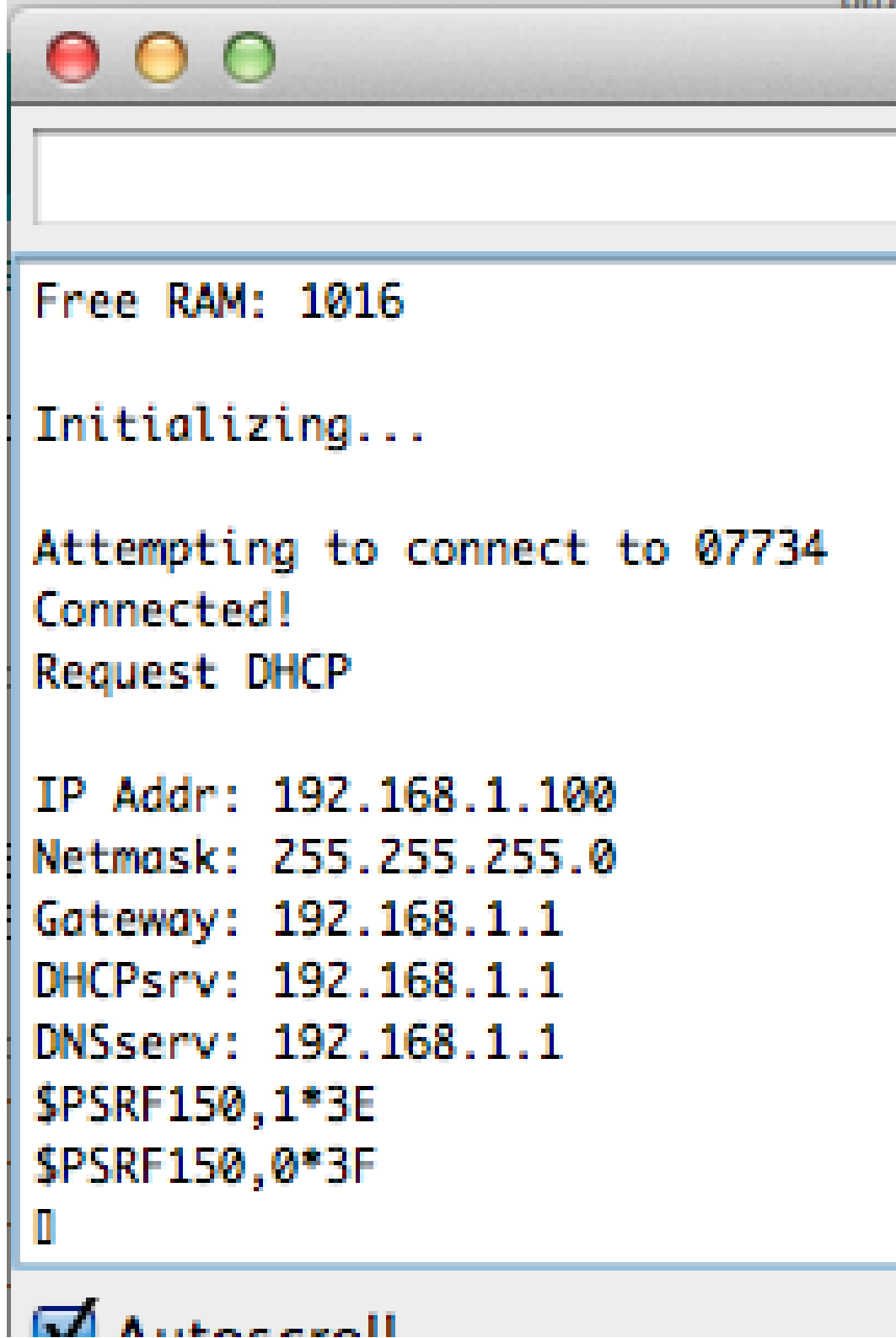
After testing each individual component,
I will now attempt to combine them in
order to create the final working tracking
collar

Compiling Parts

In order to combine the codes together, I would have to firstly make sure the components were correctly assembled. The GPS Shield would have to remain on top of the stack, as the antenna would need to have a clear access point (due to the fact that it isn't that strong because of the boards size).

The Processor board has to remain on the bottom, as this has the Coin battery slot (meaning no other boards can connect). Finally, it doesn't matter where the other two connect, just as long as when the components are not connected to the computer, the USB & ICP Board is removed





```
Free RAM: 1016  
Initializing...  
Attempting to connect to 07734  
Connected!  
Request DHCP  
  
IP Addr: 192.168.1.100  
Netmask: 255.255.255.0  
Gateway: 192.168.1.1  
DHCPsrv: 192.168.1.1  
DNSserv: 192.168.1.1  
$PSRF150,1*3E  
$PSRF150,0*3F  
[]  
  
☒ Autoscroll
```

Combining Code

After compiling all of the components together, I felt that the best way to test the codes would be an initial combination of them both. I simply copied the GPS code into the Wifi code, and made sure that all of the information was in the correct place/order.

To see whether or not the code had worked, I uploaded it onto the TinyDuino Processor board, and viewed its outcome on the serial monitor. The board was being able to connect to the internet, however, as soon as it attempted to gain a GPS signal it crashed. This therefore meant I would have to make some alterations to the code in order to fix this issue

Buffers

To try and improve the functionality of the code, I tested a few different buffers to see if they helped. Initially, I incorporated a buffer, which would assist in managing the information/data and the order of which the TinyDuino would operate (i.e. allowing the GPS to wait for the Wifi to be connected). This proved to be ineffective, and therefore I decided to remove this from the coding (by adding // in front of the code). However,

to make sure that the data was being managed properly, I attempted a String buffer instead.

This would again, help to manage the data, and allow for the information to be delivered to the specified website (in this case Alistair's website) in the correct order. Therefore, in order to test this, I had to upload the new 'improved' code onto the Processor

```
if (Gps_serial.available())
{
    byDataByte = Gps_serial.read();

    Serial.write(byDataByte);
//    pbyGpsBuffer[ byBufferIndex++ ] = byDa
//    dataBuffer += byDataByte;

    if( byBufferIndex >= 100 )
    {
//        byBufferIndex = 0;
//        sendData("Test1");
//        sendData(dataBuffer);
//        dataBuffer = "";
    }
}
```

```
String dataBuffer = "";

void loop(void)
{
    byte byDataByte;

    if (Gps_serial.available())
    {
        byDataByte = Gps_serial.read();

        Serial.write(byDataByte);
    }
}
```

Buffer Test

By viewing the serial monitor, I was able to determine if the code would now work in sending the information to Alistair's website. As can be seen on the code, the GPS did begin searching for a location in order to track the collar. However, it was not able to transfer the information to the website.

After accessing the website address, the board was trying to process too much information. Due to this, it crashed and ceased to operate. This therefore meant I would have to go back to the drawing board, and make some more alterations on the code

Initializing...

Attempting to connect to 07734

Connected!

Request DHCP

IP Addr: 192.168.1.100

Netmask: 255.255.255.0

Gateway: 192.168.1.1

DHCPsrv: 192.168.1.1

DNSserv: 192.168.1.1

\$PSRF150,1*3E

\$PSRF150,0*3F

!\$PSRF150,1*3E

\$GPGGA,,,,,0,00,,M,0.0,M,,0000*48

\$GNGNS,,,,,NN,00www.agm.me.uk ->



Autoscroll

Testing Wifi

Due to the code not working, I decided to test the Wifi shield again (with the GPS shield still attached), to make sure that it would still be functional in the combined set up. Instead of attempting to connect to Alistair's website, I tried google maps (as this would be the style in which the information would be sent to the computer). As shown below, the connection is still accurate and therefore working. This means

that the issue in the code must be with how the Wifi and GPS shields communicate with one another.

For this reason, I decided that it could be possible to turn the Wifi and GPS shields, on and off when the other is not in use



```
/dev/tty.usbserial-DN00C1T8

Set-Cookie: PREF=ID=65e7fd1397737a5a:FF=0:TM=1429105193:LM=1429105193:S=DSSWHMq0lhQ0lhBy; expires=Fri, 14-Apr-2017 13:39:53 GMT; path=/; domain=.google.com;
Set-Cookie: NID=67=JHa8wrMxx2YHe-RRyWhPOaCRb7fLnSaVSHTbF3yOjauZy_gLmuG605nwUNOD5qs6jTpAC1t9TRQuKlSkZiGXf4tZNc245KKr1HHjMeLU9xFWYc0A-QEJmXfXZ; expires=Fri, 14-Apr-2017 13:39:53 GMT; path=/; domain=.google.com;
P3P: CP="This is not a P3P policy! See http://www.google.com/support/accounts/bin/answer.py?hl=en&answer=151657 for more info."
Date: Wed, 15 Apr 2015 13:39:53 GMT
Server: gws
Content-Length: 291
X-XSS-Protection: 1; mode=block
X-Frame-Options: SAMEORIGIN
Alternate-Protocol: 80:quic,p=0.5

<HTML><HEAD><meta http-equiv="content-type" content="text/html; charset=utf-8">
<TITLE>302 Moved</TITLE></HEAD><BODY>
<H1>302 Moved</H1>
The document has moved
<A HREF="http://maps.google.co.uk/maps?ll=54.9780445,-1.6126314&zoom=15&output=classic&dg=ntvb">here</A>.
```

☒ Autoscroll No line ending

Turning Wifi on/off

With the inclusion of turning the Wifi on and off, the connection still remained strong (as well as the connection time being relatively quick). Due to the success of this test, I set about incorporating the GPS code into the new and improved Wifi code.

This was done through simply copying and pasting the needed areas, and placing them in the correct order. This would allow the Wifi shield to turn off when the GPS is in use so that a better connection could hopefully be made

```
// Wifi off, GPS on
digitalWrite( GPS_ONOFFPin, HIGH );
delay(10000);
```

```
// Wifi on, GPS off
digitalWrite( GPS_ONOFFPin, LOW );
delay(5);
```

```
#include <Adafruit_CC3000.h>
#include <ccspi.h>
#include <SPI.h>
#include <string.h>
#include "utility/debug.h"
#include <SoftwareSerial.h>

// These are the interrupt and control pins
#define ADAFRUIT_CC3000_IRQ 2 // MUST be an interrupt pin!
// These can be any two pins
#define ADAFRUIT_CC3000_VBAT A3
#define ADAFRUIT_CC3000_CS 8
// Use hardware SPI for the remaining pins
// On an UNO, SCK = 13, MISO = 12, and MOSI = 11
Adafruit_CC3000 cc3000 = Adafruit_CC3000(ADAFRUIT_CC3000_CS, ADAFRUIT_CC3000_IRQ, ADAFRUIT_CC3000_VBAT,
                                           SPI_CLOCK_DIVIDER); // you can change this clock speed

#define WLAN_SSID "07734" // cannot be longer than 32 characters!
#define WLAN_PASS "07946581732"
// Security can be WLAN_SEC_UNSEC, WLAN_SEC_WEP, WLAN_SEC_WPA or WLAN_SEC_WPA2
#define WLAN_SECURITY WLAN_SEC_WPA2

#define IDLE_TIMEOUT_MS 3000 // Amount of time to wait (in milliseconds) with no data
                             // received before closing the connection. If you know the server
                             // you're accessing is quick to respond, you can reduce this value.

// What page to grab!
#define WEBSITE "www.google.co.uk"
#define WEBPAGE "/maps/@54.9780445,-1.6126314,15z"

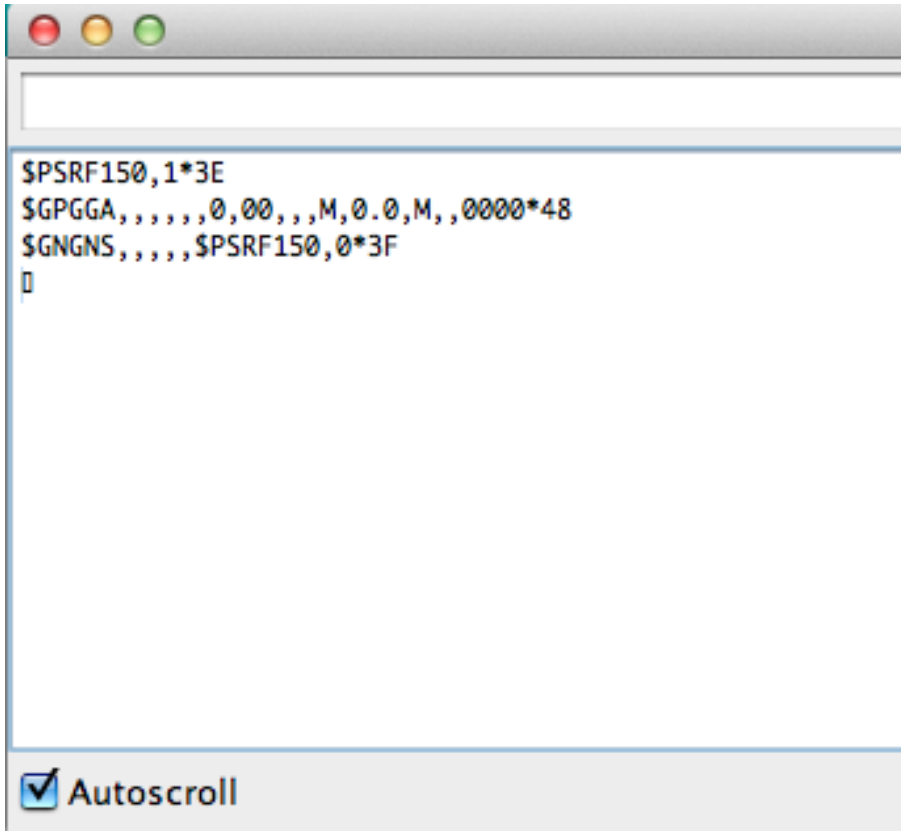
static const int GPS_ONOFFPin = A3;
static const int GPS_SYSONPin = A2;
static const int GPS_RXPin = A1;
static const int GPS_TXPin = A0;
static const int GPSBaud = 9600;
static const int chipSelect = 10;
/*****
```

Testing both Wifi and GPS

With the Wifi shield turning on and off, the same would have to be applied to the GPS shield. I would therefore have to make a delay that can be applied to this component as well. This was very easy to do, so it used the same layout as that of the Wifi shield.

However, on testing the new code out, it appeared that the GPS shield crashed

when it was turned on and off. This caused issues, as the signal was completely lost and therefore an accurate location could not be made. As with before, I wasn't sure whether or not this was due to both the Wifi shield and the GPS shield being connected at the same time. In order to determine this, I had to test the GPS shield by itself and see whether or not it could work.

A screenshot of a serial monitor window. The window has a title bar with red, yellow, and green buttons. Below the title bar is a text input field. The main area displays the following text: \$PSRF150,1*3E, \$GPGGA,,,,,0,00,,M,0.0,M,,0000*48, \$GNGNS,,,,,PSRF150,0*3F, and a cursor. At the bottom, there is a checkbox labeled 'Autoscroll' which is checked.

```
$PSRF150,1*3E
$GPGGA,,,,,0,00,,M,0.0,M,,0000*48
$GNGNS,,,,,PSRF150,0*3F
█
```

☒ Autoscroll

```
// Wifi on, GPS off
digitalWrite( GPS_ONOFFPin, LOW );
delay(10000);

// Wifi off, GPS on
digitalWrite( GPS_ONOFFPin, HIGH );
delay(5);
```

```
// The Arduino pins used by the GPS module
static const int GPS_ONOFFPin = A3;
static const int GPS_SYSONPin = A2;
static const int GPS_RXPin = A1;
static const int GPS_TXPin = A0;
static const int GPSBaud = 9600;
static const int chipSelect = 10;

// The GPS connection is attached with a software serial port
SoftwareSerial Gps_serial(GPS_RXPin, GPS_TXPin);

void setupGPS()
{
    // Init the GPS Module to wake mode
    pinMode(GPS_SYSONPin, INPUT);
    pinMode(GPS_ONOFFPin, OUTPUT);
    digitalWrite( GPS_ONOFFPin, LOW );
    delay(5);
    if( digitalRead( GPS_SYSONPin ) == LOW )
    {
        // Need to wake the module
        digitalWrite( GPS_ONOFFPin, HIGH );
        delay(5);
        digitalWrite( GPS_ONOFFPin, LOW );
    }

    // Open the GPS serial port
    Gps_serial.begin(GPSBaud);
}

void setup()
{
    // Open the debug serial port at 9600
    Serial.begin(115200);

    setupGPS();

    /*
```

Testing GPS

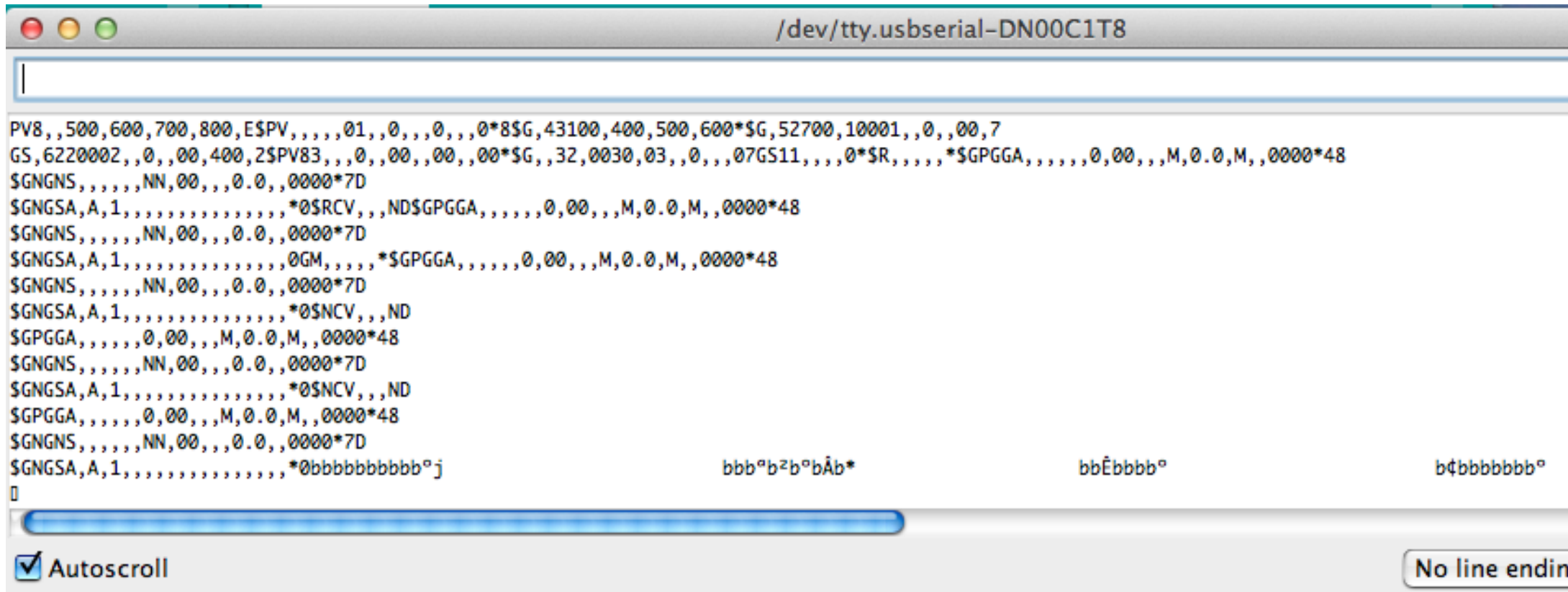
Once testing the GPS shield by itself, it began working as normal. However, there was still the issue of trying to figure out how quickly it would pick up a signal after being turned on and off again. To try and find an accurate time for this, I created a code that would allow an input in the serial monitor to stop the GPS shield for 10s. I would then be able to time how long it would take for the GPS shield to start up and find the location again.

Turning GPS on/off

As stated, the new code allowed me to input any message into the serial monitor (e.g. ff), and this would turn the GPS shield off and on. I would then time how long it would take for the shield to start finding the location again. I first tested this inside (where there would be no connection to the satellites), just to see whether or no the shield would turn off/on.

Initially, there was an error with the scan, and the shield crashed again. I repeated the test again, just to make sure that it wasn't an error simply with the connection to the computer. The second time, the GPS started to work again, and took an overall time of 20s to fully restart. Even though this was slightly longer than expected, It would still be effective as there would not be too much of a delay for the tracking to take

place



The screenshot shows a serial monitor window titled "/dev/tty.usbserial-DN00C1T8". The window contains a list of NMEA sentences from a GPS module, including PV8, GS, \$GNGNS, \$GNGSA, and \$GPGGA. At the bottom, there are three lines of text: "*0bbbbbbbbbbb°j", "bbb°b²b°bÅb*", and "bbÊbbbb°". The window also has a scrollbar and a checkbox for "Autoscroll" which is checked.

```
PV8,,500,600,700,800,E$PV,,,,,01,,0,,,0,,,0*8$G,43100,400,500,600*$G,52700,10001,,0,,00,7
GS,6220002,,0,,00,400,2$PV83,,0,,00,,00,,00*$G,,32,0030,03,,0,,,07GS11,,,0*$R,,,,,$GPGGA,,,,,0,00,,M,0.0,M,,0000*48
$GNGNS,,,,,NN,00,,0.0,,0000*7D
$GNGSA,A,1,,,,,,$$RCV,,ND$GPGGA,,,,,0,00,,M,0.0,M,,0000*48
$GNGNS,,,,,NN,00,,0.0,,0000*7D
$GNGSA,A,1,,,,,,$$GM,,,,,$$GPGGA,,,,,0,00,,M,0.0,M,,0000*48
$GNGNS,,,,,NN,00,,0.0,,0000*7D
$GNGSA,A,1,,,,,,$$NCV,,ND
$GPGGA,,,,,0,00,,M,0.0,M,,0000*48
$GNGNS,,,,,NN,00,,0.0,,0000*7D
$GNGSA,A,1,,,,,,$$NCV,,ND
$GPGGA,,,,,0,00,,M,0.0,M,,0000*48
$GNGNS,,,,,NN,00,,0.0,,0000*7D
$GNGSA,A,1,,,,,,$$0bbbbbbbbbbb°j
bbb°b²b°bÅb*
bbÊbbbb°
b¢bbbbbbbb°
```

☒ Autoscroll No line ending

Turning GPS on/off outside

However, in order for the test to be fully accurate, I would have to attempt to test the GPS shield outside. Doing so caused great issues, as after restarting, it took 5 minutes before the GPS shield managed to recalculate the time. To make matters worse, it took nearly an hour to reconnect to the satellites in order to find a location.

This would not be helpful at all, as each

reading for the computer would take an hour in between. In this time, the animal could have easily travelled a large distance away from the original location. Due to this, I became stuck towards what I would be able to do in order to make the collar actually work

/dev/tty.usbserial-DN00C1T8

```
00,3
00,4$PS8,2600,10001,,0,,00,5
00,6$PV,,,400,600020,02,,0*$G,,,,,00,000,3000,,,0*$GV,,,,,.*$R,50.9V,,262,5$GPGGA,152510.769,,,,,0,00,,M,0.0,M,,0000*5C
,,,NN,00,,,0.0,,0000*69
2179,,,76,N5
,,,0,00,,,M,0.0,M,,0000*5D
,,,NN,00,,,0.0,,0000*68

GA,152512.769,,,,,0,00,,,M,0.0,M,,0000*5E
,,,NN,00,,,0.0,,0000*6B
217,,,76,N5
,,,0,00,,,M,0.0,M,,0000*5F
,,,NN,00,,,0.0,,0000*6A
5179,,,76,N5
```

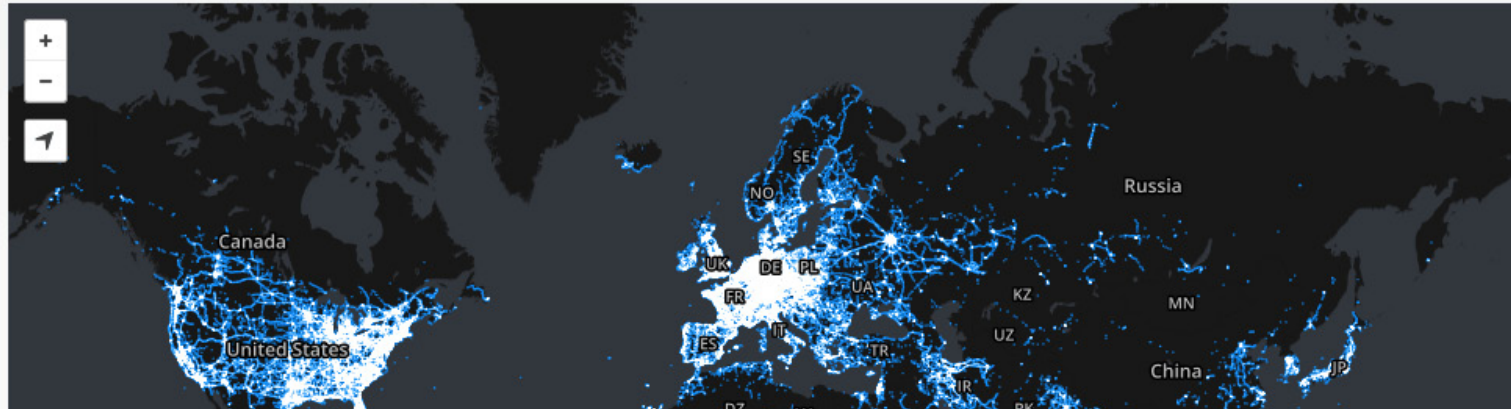

What next?

As none of the current attempts had been successful, I had to go back to the drawing board and take a completely new approach

Map

This map shows areas in which our stumbling community has collected data samples. The map is updated once a day, after midnight in UTC.

The map shows general areas in which we have data samples. It does not show the locations of cell towers or WiFi access points.



Location Services

I looked into the idea of trying to incorporate location services into my design, instead of using the actual GPS shield. Simply due to the issues I was having, the collar would just use the Wifi shield, and use local servers in order to triangulate the location.

Mozilla has already attempted something like this with “Mozilla Location Service”. Basically, any device (mainly phones)

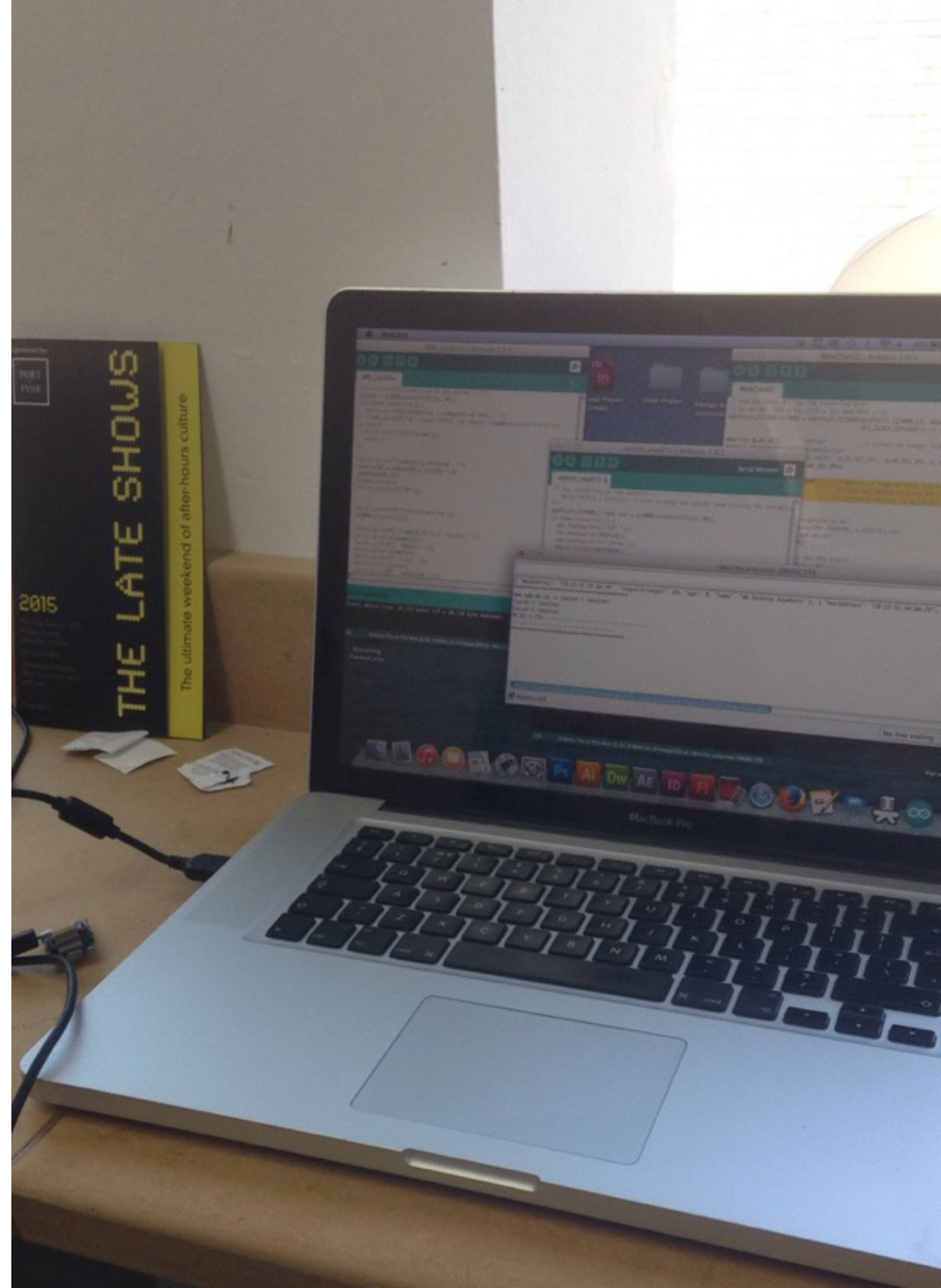
would use other Wifi access points and cell towers and find the strongest signals. It would then calculate where each of the points overlapped, in order to determine the exact location of the device. This would reduce the need for too many components on the collar, but would require a constant Wifi connection. This could be easily resolved through the use of the personal hotspots on mobiles devices which the Wifi

shield could connect to

Method

The way in which I was going to make sure that the code would work (and allow me to triangulate a location), was through amending the existing Wifi code itself.

In the example codes for the CC3000 (which is the Wifi chip), there is already a code available for a Wifi location. However, this simply connects to the local server and finds the location of that. This would not be useful, as it would only use the server which the Wifi shield is connected to, and not give an accurate location



Tracking Code Explained

The initial code in which I had created for the tracking collar was extremely complicated and long. I have tried to explain it to the best of my ability referring to the images on the next page:

1 - The “#include” and first section of “#define” is how the cc3000 is started and recognised by the Arduino software. Following this, the Wifi in which the shield will transfer the data through is labelled. The SSID is the name of the router, and the PASS is the password. A 30s delay (3000) is in position so the shield can calculate and process all of the information being received. I then stated the test website I would be uploading the co-ordinates to as the client

2 - The “Wifi Scan” locates the local servers, and is sent to the “json” which is an online page to store the information. The void set ups refer to what information is being received, and if they match the following characters (0123456789ABCDEF). This is to identify each of the servers and their specific ID codes

3 - The first section formats the servers and SSIDs scanned into a geolocation. It scans 16 times with 2s in between each scan.

Finally, when the scan receives a scan result equal to 0 (meaning it cannot find anymore servers), it will cease to scan for anymore

4 - However, I have programmed it not to include the Wifi in which the cc3000 is connected to (the personal hotspot). This is because the location which the hotspot will deliver will be the mainline it is registered to. This could therefore mess up the overall result, as that location wouldn't change with the other servers. I set the list of servers to 6, so the name and location of servers will only be recorded from the strongest 6 available

5 - A void setup has been included to display an error message if no servers are detected in the scan. The main loop has been included, to make sure that the online database knows when a new set of readings is being delivered, in order to update the results

6 - If the results have been successfully sent, the display on the serial monitor will be “*****”

Finally, the shield will look up the websites IP address in order to send the information to the final page

7 - If connected, the name of the web address we are using will be displayed (Alistair's blog). If not, the following “-----” will appear to display an error. Finally, if there has been no readings for two attempts, the shield will turn itself off in order to save on power and memory

```
WiFiTracking | Arduino 1.0.5

#include <Adafruit_CC3000.h>
#include <Adafruit_CC3000_Server.h>
#include <ccspi.h>
#include <SPI.h>
#include <string.h>
#include <utility/debug.h>
#include <SoftwareSerial.h>

#define ADAFRUIT_CC3000_IRQ 2
#define ADAFRUIT_CC3000_VBAT A3
#define ADAFRUIT_CC3000_CS 8

Adafruit_CC3000 cc3000 = Adafruit_CC3000(ADAFRUIT_CC3000_CS,
ADAFRUIT_CC3000_IRQ, ADAFRUIT_CC3000_VBAT,
SPI_CLOCK_DIVIDER);

#define WLAN_SSID "87734"
#define WLAN_PASS "87946591732"
#define WLAN_SECURITY WLAN_SEC_WPA2

#define IDLE_TIMEOUT_MS 3000 // Amount of time to wait (in milliseconds) with no data
// received before closing the connection. If you know you're accessing is quick to respond, you can reduce this.

Adafruit_CC3000_Client client;

#define WEBSITE "www.ogm.me.uk"
#define WEBPAGE "/unn/dogtrack/update2.php"

uint32_t ip;

// The library code...

typedef struct WifiScanResults {
  uint32_t networks;
  uint32_t status;
  uint8_t rssi;
  uint8_t security;
  uint16_t time;
  uint8_t ssid[32];
}
```

```
WiFiTracking | Arduino 1.0.5

uint8_t bssid[6];
} WifiScanResults_t;

class WifiScan {
private:
  String parseScanResultToJson(WifiScanResults_t result);
public:
  int scanResultsCount;
  String scanResults;
  int startScan();
  bool Next();
};

void wipeStr(char *str, int len) {
  for(int i = 0; i < len; i++) {
    str[i] = 0;
  }
}

void bytesToHex(unsigned char *data, int length, char * buffer) {
  const char * hex = "0123456789ABCDEF";
  int i=0, a=0;

  for(i=0;i<length;i++) {
    unsigned char c = data[i];
    buffer[a] = hex[(c>>4) & 0xF];
    buffer[a+1] = hex[(c << 4) & 0xF];
    buffer[a+2] = ':';
    a+=3;
  }
  buffer[a-1] = 0; //null
  buffer[a] = 0;
}

// Format a SSID Scan result into a GeoLocation API Wifi Object
String WifiScan::parseScanResultToJson(WifiScanResults_t result) {

  uint8_t length = (result.security>>2);
  uint8_t rssi = (result.rssi>>1);
```

```
WiFiTracking | Arduino 1.0.5

char ssidStr[32];
strcpy(ssidStr, (char *)result.ssid, length);
ssidStr[length] = 0;

char bssidStr[24];
bytesToHex((unsigned char *)result.bssid, 6, bssidStr);

//more standard geolocation format
String json = "{" + String(bssidStr) + "\", \"signalStrength\": " +
return json;

int WifiScan::startScan() {
  const unsigned long intervalTime[16] = { 2000, 2000, 2000, 2000, 2000,
    2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000 };

  wlan_ioctl_set_scan_params(4000, 20, 100, 5, 0x7FF, -120, 0, 300,
    (unsigned long *) &intervalTime);

  scanResultsCount = 0;
  scanResults = "";
  return 0;
}

bool WifiScan::Next() {
  WifiScanResults_t scanResult;

  wipeStr((char *)scanResult.ssid, 32);

  long err = wlan_ioctl_get_scan_results(0, (uint8_t*) &scanResult);

  // Check if we are at the end of the list
  if ( (scanResult.bssid[0]==0x00) &&
    (scanResult.bssid[1]==0x00) &&
    (scanResult.bssid[2]==0x00) &&
    (scanResult.bssid[3]==0x00) &&
    (scanResult.bssid[4]==0x00) &&
    (scanResult.bssid[5]==0x00) ) {
```

```
WiFiTracking | Arduino 1.0.5

(scanResult.bssid[4]==0x00) &&
(scanResult.bssid[5]==0x00) ) {
  return false;
}

// Skip the wifi that can move with us
if ( (scanResult.bssid[0]==0x4C) &&
  (scanResult.bssid[1]==0x54) &&
  (scanResult.bssid[2]==0x99) &&
  (scanResult.bssid[3]==0x51) &&
  (scanResult.bssid[4]==0x0C) &&
  (scanResult.bssid[5]==0xC5) ) {
  return true;
}

// Add the new result if we have one and the list is not full
if ( (err==0) && (scanResultsCount<40) ) {
  if (scanResultsCount==0) {
    scanResults += ",";
  }
  scanResults += parseScanResultToJson(scanResult);
  scanResultsCount++;
}

// All done for now, but more results to come
return true;
}

// Function to send the results to the server
// The main body of code...

WifiScan WifiScanner;
uint8_t scanning = 0;
int startTime = 0;
int lastScan = 0;
int lastStep = 0;

// Setup the hardware
void setup() {
  Serial.begin(115200);
```

```
WiFiTracking | Arduino 1.0.5

if(!cc3000.begin()) {
  Serial.println(F("Failed. Check your wiring?"));
  return;
}

if(!cc3000.connectToAP(WLAN_SSID, WLAN_PASS, WLAN_SECURITY)) {
  Serial.println(F("Failed to connect to AP-"));
  return;
}

// The main loop
void loop() {
  unsigned int now = millis();

  if ((now - lastScan) > 10000) {
    if (!scanning) {
      WifiScanner.startScan();
    }
    scanning = 1;
    startTime = now;
    lastScan = now;
  }

  //auto scan
  if ((now - lastStep) > 100) {
    scanNext();
    lastStep = now;
  }
}

int scanNext() {
  if ((millis() - startTime) > 10000) {
    scanning = 0;
  }

  if (!scanning) {
    return 0;
  }
}
```

```
WiFiTracking | Arduino 1.0.5

if (!WifiScanner.Next()) {
  Serial.println(WifiScanner.scanResults);
  Serial.println("*****");
  scanning = 0;
  sendData(WifiScanner.scanResults);
  Serial.println("*****");
}

return scanning;

void sendData(String stringToSend) {

  ip = 0;
  // Try looking up the website's IP address
  Serial.print(WEBSITE); Serial.print(F(" -> "));
  while (ip == 0) {
    if (!cc3000.getHostByName(WEBSITE, &ip)) {
      Serial.println(F("Couldn't resolve!"));
    }
    delay(500);
  }

  cc3000.printIPdotsRev(ip);

  // Optional: Do a ping test on the website
  Serial.print(F("\n\nPing: ")); cc3000.printIPdotsRev(ip); Serial.print("...");
  replies = cc3000.ping(ip, 5);
  Serial.print(replies); Serial.println(F(" replies"));
}

/* Try connecting to the website.
Note: HTTP/1.1 protocol is used to keep the server from closing the connection before
*/
Adafruit_CC3000_Client www = cc3000.connectTCP(ip, 80);
if (www.connected()) {

  /* Try connecting to the website.
Note: HTTP/1.1 protocol is used to keep the server from closing the connection before
*/
Adafruit_CC3000_Client www = cc3000.connectTCP(ip, 80);
if (www.connected()) {
```

```
WiFiTracking | Arduino 1.0.5

/* Try connecting to the website.
Note: HTTP/1.1 protocol is used to keep the server from closing the connection before
*/
Adafruit_CC3000_Client www = cc3000.connectTCP(ip, 80);
if (www.connected()) {

  www.fastprint(F("PUT "));
  www.fastprint(F("/unn/dogtrack/update2.php"));
  www.fastprint(F(" HTTP/1.1\r\n"));
  www.fastprint(F("Host: "));
  www.fastprint(F("www.ogm.me.uk"));
  www.fastprint(F("\r\n"));
  www.fastprint(F("\r\n"));

  delay(2000);
  while (www.connected() && www.available()) {
    www.read();
  }

  www.fastprint(results.c_str());
  delay(2000);
  www.close();
}

Serial.println(F("-----"));

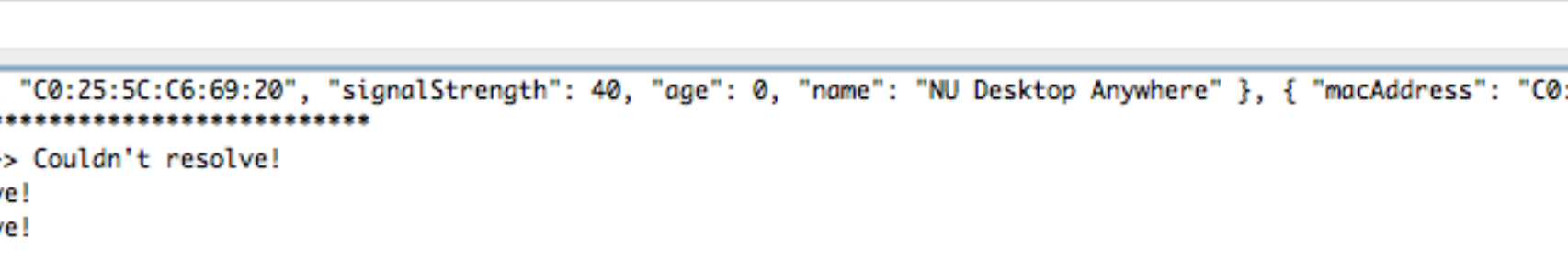
/* Read data until either the connection is closed, or the idle timeout is reached. */
unsigned long lastRead = millis();
while (www.connected() && (millis() - lastRead < IDLE_TIMEOUT_MS)) {
  while (www.available()) {
    char c = www.read();
    Serial.print(c);
    lastRead = millis();
  }
}
www.close();
Serial.println(F("-----"));
}
```

1st Code Test

Initially the very first test was unsuccessful due to what was thought to be a memory issue for the TinyDuino. As seen, the servers were being recognised by the json and being defined. However, when it got to actually uploading the information, it completely failed.

As soon as the main webpage was opened, the shield could not locate the database

in which the information was going to be stored. This therefore resulted in the processor to crash in order for it to display "Couldn't resolve!". To try and fix this issue, I would have to amend the code in order to reduce the amount of memory being used in the scanning and uploading process



The screenshot shows a terminal window with a title bar that includes standard macOS window controls (red, yellow, green buttons) and the path `/dev/tty.usbserial-DN00C1T8`. The terminal output displays the results of a network scan, showing two discovered devices with their MAC addresses, signal strengths, and names. Below this, there are three failed DNS resolution attempts for `www.agm.me.uk` and the IP address `95.85.5.234`. At the bottom of the window, there is a scrollbar and a status bar with a checked "Autoscroll" checkbox and a partially visible "No line e" label.

```
/dev/tty.usbserial-DN00C1T8

{ "macAddress": "C0:25:5C:C6:69:20", "signalStrength": 40, "age": 0, "name": "NU Desktop Anywhere" }, { "macAddress": "C0:25:5C:C6:69:20", "signalStrength": 40, "age": 0, "name": "NU Desktop Anywhere" }
*****
www.agm.me.uk -> Couldn't resolve!
Couldn't resolve!
Couldn't resolve!
95.85.5.234

Autoscroll No line e
```



```
// Add the new result if we have one and the list is not full
if ( (err==0) && (scanResultsCount<3) ) {
    if (scanResultsCount!=0) {
        scanResults += ", ";
    }
    scanResults += parseScanResultToJson(scanResult);
    scanResultsCount++;
}
```

```
String urlEncode(String src) {
    return src;
    // needsssssss wwwwooorrrrrkkk

    String dst;
    for (int i=0; i<src.length(); i++) {
        if (src[i]==' ') {
            dst += "+";
        }
        else if ((src[i]>='a') && (src[i]<='z')) {
            dst += src[i];
        }
        else if ((src[i]>='A') && (src[i]<='Z')) {
            dst += src[i];
        }
        else if ((src[i]>='0') && (src[i]<='9')) {
            dst += src[i];
        }
        else {
            dst += "%" + String(src[i], HEX);
        }
    }
    return dst;
}
```

2nd Code

To reduce the amount of memory being used by the processor, I changed the amount of servers being used to 3. This would reduce the amount of information being stored in order to send to the website. As well as this, 3 servers would be enough to gain a fairly accurate location.

Secondly, was to encode the string (which is how the information is sent across to the website) to increase security as well as reduce the size of data. It is basically the equivalent as making a 'zip' file in order to compress the data for when it is being sent

3rd Code

Again, after testing the new code, we were still having no result. As with the previous combination of the Wifi and GPS code, I decided to attempt to incorporate a Buffer into the code. This would help to reduce the size of the data again, and hope fully improve the speed of uploading the information

The Buffersize was reduced to 256 bytes to therefore allow more free space on the memory. I also removed any unwanted pieces of code (including some of the Adafruit cc3000 server), as the shield does not need to use them in order to function. This would allow more space in order for the scanning and transmitting of data to be completed

```

Revised_tracker | Arduino 1.0.5
#include <Adafruit_CC3000.h>
// #include <Adafruit_CC3000_Server.h>
// #include <ccspi.h>
#include <SPI.h>
// #include <string.h>
// #include "utility/debug.h"
// #include <SoftwareSerial.h>

#define ADAFRUIT_CC3000_IRQ 2
#define ADAFRUIT_CC3000_VBAT A3
#define ADAFRUIT_CC3000_CS 8

Adafruit_CC3000 cc3000 = Adafruit_CC3000(ADAFRUIT_CC3000_CS,
ADAFRUIT_CC3000_IRQ, ADAFRUIT_CC3000_VBAT,
SPI_CLOCK_DIVIDER);

#define WLAN_SSID "07734"
#define WLAN_PASS "07946581732"
#define WLAN_SECURITY WLAN_SEC_WPA2

#define IDLE_TIMEOUT_MS 3000 // Amount of time to wait (in milliseconds) with no data
// received before closing the connection. If you know
// you're accessing is quick to respond, you can reduce this.

Adafruit_CC3000_Client client;

#define WEBSITE "www.agm.me.uk"
#define WEBPAGE "/unn/dogtrack/update2.php?data="
#define MAX_RECORDS 32
#define FREQUENCY 10000
#define BUFFERSIZE 256

uint32_t ip;

// The library code...

typedef struct WifiScanResults {
  uint32_t networks;
  uint32_t status;
  uint8_t rssi;
  // Add the new result if we have one and the list is npt full
  if ( (err==0) && (scanResultsCount<MAX_RECORDS) ) {
    // Get the results
    String scanResults = urlencode(parseScanResultToJson(scanResult));
    String scanResults = (parseScanResultToJson(scanResult));
    // Can we fit this in the buffer?
    if ((scanResults.length()+4)<(BUFFERSIZE-1)) {
      // Add a seperator if we already have a record
      if (scanResultsCount!=0) {
        scanResultsPointer[0] = '%';
        scanResultsPointer[1] = '2';
        scanResultsPointer[2] = 'C';
        scanResultsPointer[3] = '+';
        scanResultsPointer+=4;
      }
    }
  }
}

uint8_t security_ssid;
uint16_t time;
uint8_t ssid[32];
uint8_t bssid[6];
} WifiScanResults_t;

class WifiScan {
private:
  String parseScanResultToJson(WifiScanResults_t result);
public:
  int scanResultsCount;
  char scanResultsBuffer[BUFFERSIZE];
  char* scanResultsPointer;
  int startScan();
  bool Next();
};

void wipeStr(char *str, int len) {
  for(int i = 0; i < len; i++) {
    str[i] = 0;
  }
}

void bytesToHex(unsigned char *data, int length, char* buffer) {
  const char * hex = "0123456789ABCDEF";
  int i=0, a=0;

  for(i=0;i<length;i++) {
    unsigned char c = data[i];
    buffer[a] = hex[(c>>4) & 0xF];
    buffer[a+1] = hex[(c & 0xF)];
    buffer[a+2] = ':';
    a+=3;
  }
  buffer[a-1] = 0; //null
  buffer[a] = 0;
}
  
```

```

// Add the new result if we have one and the list is npt full
if ( (err==0) && (scanResultsCount<MAX_RECORDS) ) {
  // Get the results
  String scanResults = urlencode(parseScanResultToJson(scanResult));
  String scanResults = (parseScanResultToJson(scanResult));
  // Can we fit this in the buffer?
  if ((scanResults.length()+4)<(BUFFERSIZE-1)) {
    // Add a seperator if we already have a record
    if (scanResultsCount!=0) {
      scanResultsPointer[0] = '%';
      scanResultsPointer[1] = '2';
      scanResultsPointer[2] = 'C';
      scanResultsPointer[3] = '+';
      scanResultsPointer+=4;
    }
  }
}
  
```

```

#define MAX_RECORDS 32
#define FREQUENCY 10000
#define BUFFERSIZE 256
  
```

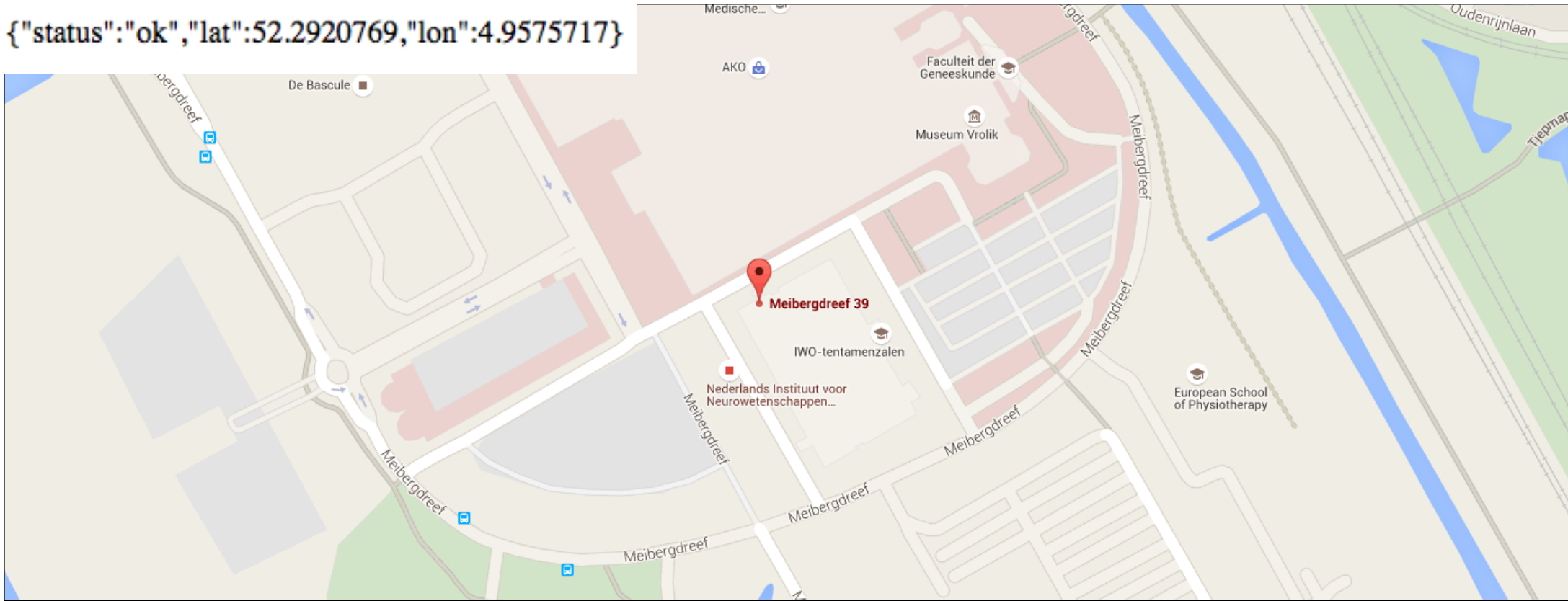

3rd Code Test

With the memory of the cc3000 being maintained by the buffers, I decided to test to see whether or not the code had been more successful. Once running the code a much more positive result was fed back. As seen, a clear location had been displayed in Alastiar's blog.

Once received, I decided to test the co-ordinates in order to see if they were

accurate to my location. I was testing the code in Northumbria University, therefore I accepted a location near there to be displayed. However, once I typed the location into Google maps, I was very surprised by the outcome. For some reason, the readings shown thought that my location was in Amsterdam. This was of course extremely wrong and needed to be corrected

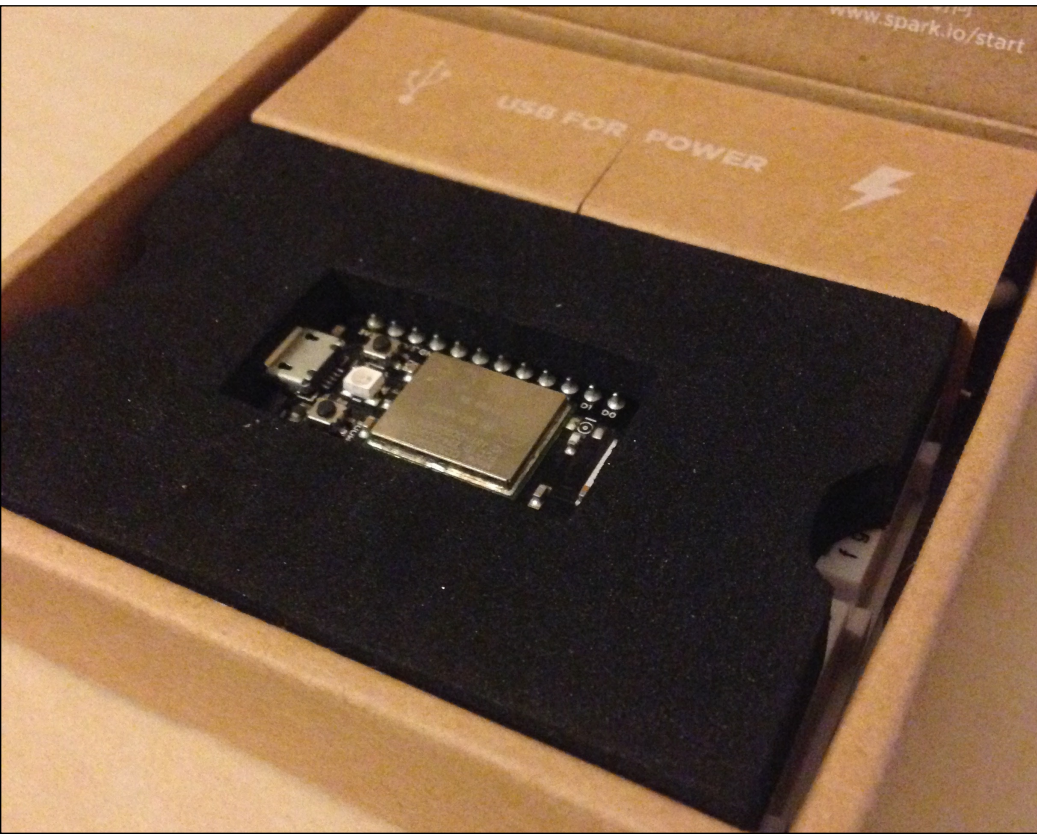
```
{"status":"ok","lat":52.2920769,"lon":4.9575717}
```



Spark Core

After having issues with the TinyDuino, Alistair recommend that I should attempt the same code with just a Spark Core. This uses the same cc3000, yet has a slightly more powerful memory. I input the code onto the Spark Core coding site, and began by testing the location which I received. However, before I could run the test, I had to make sure that I created my own online database for the information to be

transmitted to. In order to do this, I had to create a few php pages. These were very simple to create and code, as Alistair had already given me access to a few pieces of the code, I basically needed to re-code certain areas, in order to make sure that they applied to me



```
dogtrack.ino
1 // This #include statement was automatically added by the Spark IDE.
2 #include "HttpClient/HttpClient.h"
3
4 // The library code...
5
6 typedef struct WifiScanResults {
7     uint32_t networks;
8     uint32_t status;
9     uint8_t rssi;
10    uint8_t security_ssidlen;
11    uint16_t time;
12    uint8_t ssid[32];
13    uint8_t bssid[6];
14 } WifiScanResults_t;
15
16
17 class WifiScan {
18     private:
19         String parseScanResultToJson(WifiScanResults_t result);
20     public:
21         int scanResultsCount=0;
22         String scanResults;
23         int startScan();
24         bool Next();
25
26 };
27
```

index and update.php

In order to create the initial php documents, I had to create a new folder in my “public_html” file. In here I would create all of the php files in order to store the information received by the Wifi shield.

1 - The index file basically stores the information received from the json. This contains all of the information about the servers in order to send to the update php

file

(every 30s)

2 - The update file calculates the geolocation from the servers, in order to publish the results as co-ordinates for the map. It also uses the Mozilla location services feature in order to access this location. The update file is key in making sure that the map keeps displaying the location in real time after the last scan

public_html/backendGo

Collapse All

/home/lukeokeeffe

.appdata

.cagefs

.cl.selector

.cpanel

.htpasswd

.mysql_backup

.trash

application_backups

etc

logs

mail

public_ftp

public_html

backend

cgi-bin

wp-admin

wp-content

wp-includes

tmp

Name
maps
index.php
latest.csv
latest.json
latest2.csv
latest2.json
log.csv
log2.csv
update2.php

<?PHP

1

```
echo '<pre>';

readfile('latest.json');

echo '</pre>';

?>
```

2

```
<?PHP

// A function so send the data to the server
function lookupLocation($inData) {
    // Create the CURL object to make the request
    $session = curl_init();
    curl_setopt($session, CURLOPT_URL, 'https://location.services.mozilla.com/v1/search?key=test' );
    // curl_setopt($session, CURLOPT_URL, 'https://location.services.mozilla.com/v1/search?key=7e40f68c-7938-4c5d-9f95-e61647c213eb' );
    curl_setopt($session, CURLOPT_URL, 'https://www.googleapis.com/geolocation/v1/geolocate?key=AIzaSyD-s-mXL4mBzF7KMRkhTCIbG2RKnRGXzJc' );
    curl_setopt($session, CURLOPT_HEADER, false );
    curl_setopt($session, CURLOPT_RETURNTRANSFER, 1 );
    curl_setopt($session, CURLOPT_HTTPHEADER, array('Content-Type: application/json'));
    curl_setopt($session, CURLOPT_POST, 1);
    curl_setopt($session, CURLOPT_POSTFIELDS, $inData);
    // Send the request
    $theResult = curl_exec($session);
    curl_close($session );
    // Return the result
    return $theResult;
}

// Get the data sent to us
// $accessPoints = file_get_contents('php://input');
$accessPoints = $_GET['data'];
$theResult = json_decode(lookupLocation('{ "wifiAccessPoints": [ ' . $accessPoints . ' ] }'), true);

// Extract the results
if (isset($theResult['location'])) {
    $lat = $theResult['location']['lat'];
    $long = $theResult['location']['lng'];
    $found = true;
}
else if ( (isset($theResult['status'])) && ($theResult['status']=='ok') ) {
    $lat = $theResult['lat'];
    $long = $theResult['lon'];
    $found = true;
}
else {
    $lat = 0;
    $long = 0;
}
```

Map Construction

To create the map that I would be using on my website, I would have to try and use an already existing map (to save time in creating my own)

1 - Firstly, I had to create a page in which the map would be displayed. The map.php acts as a page which the map can be displayed on for people to interact with. This basically shows the java script from the

map.js

2 - I have used an existing microsoft map (bing maps) in order to show the location of the tracking collar. The map is fully intractable and is similar to that of google maps. The code was already available, and I simply copied it across to my specific java script file

3 - However, in order to use the map, I had to gain an API key. This is a secure key that allows me to use certain features (in this case the microsoft maps) onto a different webpage

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>

  <head>
    <script charset="UTF-8" type="text/javascript" src="http://ecn.dev.virtualearth.net/mapcontrol/
mapcontrol.ashx?v=7.0"></script>
    <script type="text/javascript" src="map.js"></script>
  </head>

  <body onload="onLoad()">
    <div id="mapDiv" style="position: absolute; top: 0px; left: 0px; width: 100%; height: 100%;"></div>
```

Dog Tracker

Application URL

http://www.pawprintnurse.com

Key type *

What's This

Basic

Application type *

Broadcast

Enter the characters you see *

```
var map; // The var to hold the map object.
var pin; // The var to hold the pointer object.

// getXmlHttpRequestObject - Creates an XMLHttpRequest object
function getXmlHttpRequestObject() {
  if (window.XMLHttpRequest) {
    return new XMLHttpRequest(); //Not IE
  } else if(window.ActiveXObject) {
    return new ActiveXObject("Microsoft.XMLHTTP"); //IE
  } else {
    alert("Error : Your browser doesn't support the XmlHttpRequest object.");
    return false;
  }
}

function ticktock1() {

  var receiveReq = getXmlHttpRequestObject();
  if ((receiveReq) && (receiveReq.readyState == 4 || receiveReq.readyState == 0)) {
    receiveReq.open("GET", "../latest2.json", false);
    receiveReq.send(null);
    if ((receiveReq.readyState == 4) && (receiveReq.status == 200)) {
      var responseText = receiveReq.responseText;
      var responseData = eval( '(' + responseText + ' )' );

      pin.setLocation(new Microsoft.Maps.Location(responseData.lat, responseData.long));

      var viewBoundaries = Microsoft.Maps.LocationRect.fromLocations(
        new Microsoft.Maps.Location(responseData.lat, responseData.long)
      );
      map.setView({bounds: viewBoundaries});

    }
  }

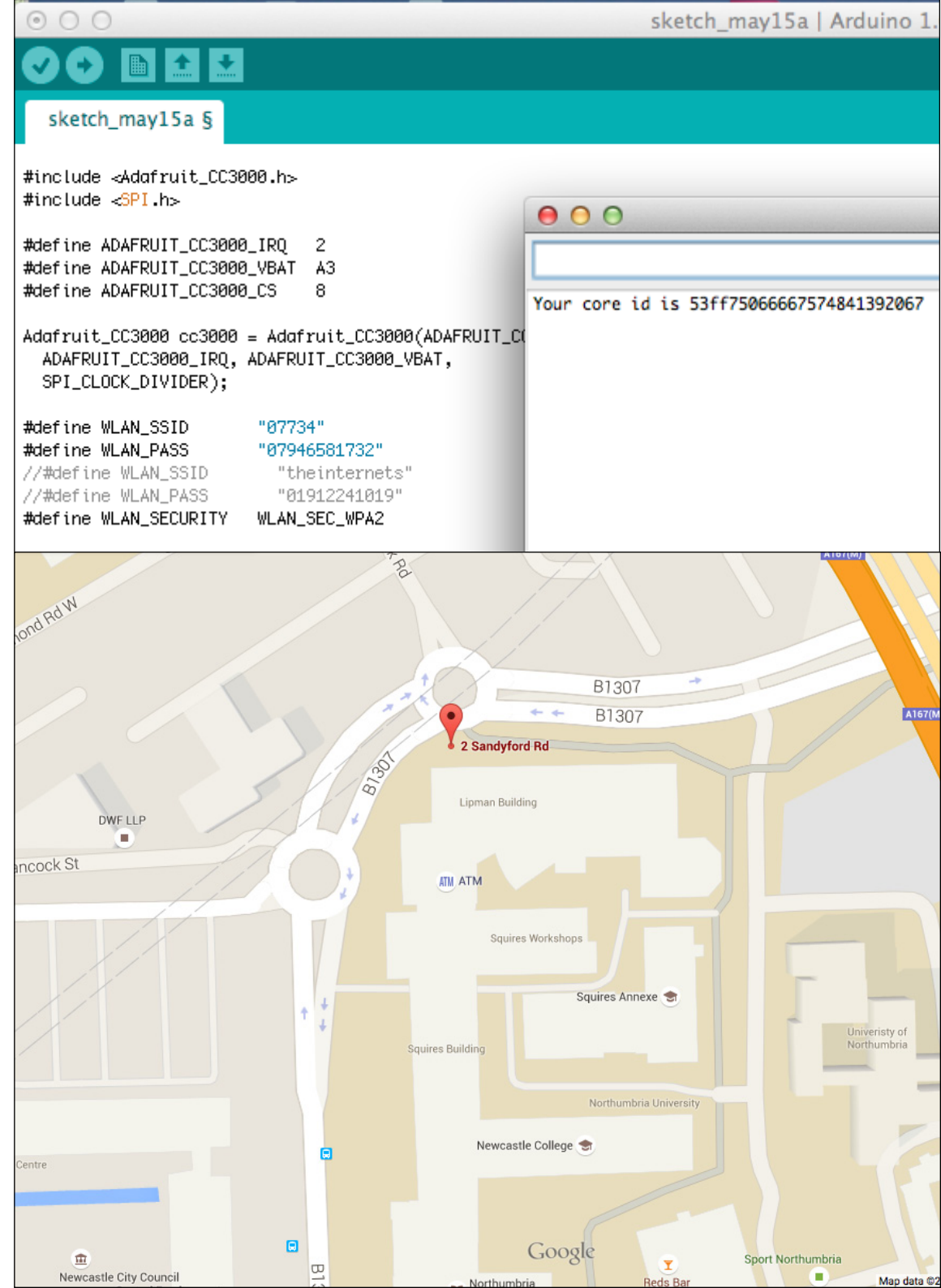
  setTimeout(ticktock1, 60000);
}

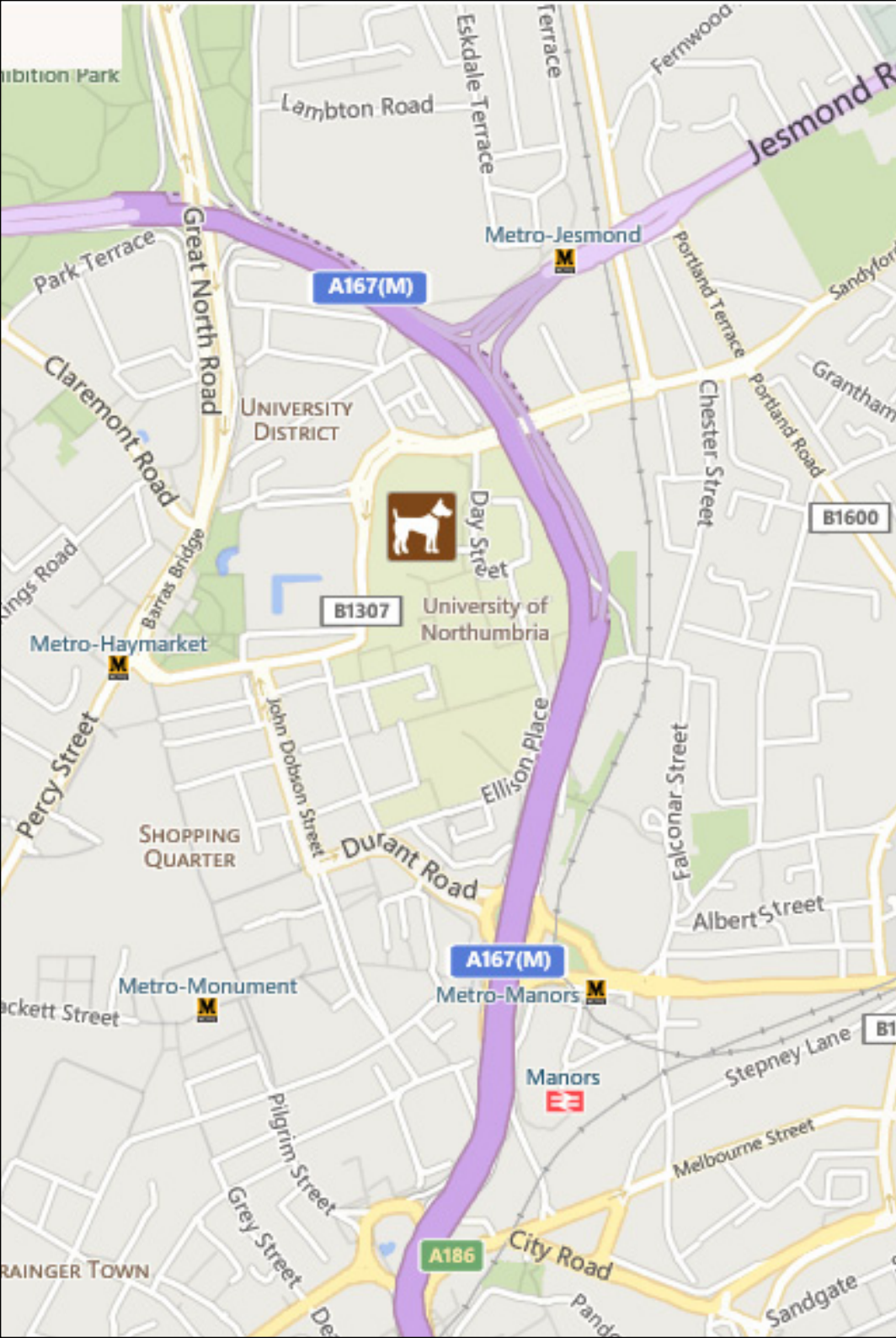
var mapviewer;
```


Spark Core Test

Now I had somewhere to display all of the information from the scans, I attempted the Spark Core test. I had to open up an Arduino sketch (didn't have to work) in order to access the serial monitor. Once opened, by typing in "I", it will display the core id to check if the correct spark core is registered. I then entered "w" which connected me to the local servers, and then I typed in the password of that relevant server I would use.

After connecting to my local wifi, I could then run the test on the update.php page. The result of the test showed that I was located round the corner from my actual location. This was a great achievement, and I was on the route to having the collar being functional





Finalised Code

After gaining success with the Spark Core code, I made a few alterations to the TinyDuino code. I maintained majority of the other factors from the previous codes, however, I reduced the amount of memory being used again. The basic order of the code is as follows:

- Idle time out at 60s
- Gain access to website
- Find locations and display in json
- Check for new servers (ignoring the one connected)
- Record the results and send them
- Re-scan but be careful of memory
- Check to see if website is available
- Keep scanning for new servers
- Start sending the data to the website
- Confirm that the data has been sent
- Confirm the information has been recorded on the update
- Finish the loop
- Turn off if idle for longer than 2mins

```
dogtracktest_13_10 | Arduino 1.0.5

#include <Adafruit_CC3000.h>
#include <SPI.h>

#define ADAFRUIT_CC3000_IRQ 2
#define ADAFRUIT_CC3000_VBAT A3
#define ADAFRUIT_CC3000_CS 8

Adafruit_CC3000 cc3000 = Adafruit_CC3000(ADAFRUIT_CC3000_CS,
    ADAFRUIT_CC3000_IRQ, ADAFRUIT_CC3000_VBAT,
    SPI_CLOCK_DIVIDER);

#define WLAN_SSID "07734"
#define WLAN_PASS "07946581732"
#define WLAN_SECURITY WLAN_SEC_WPA2

Adafruit_CC3000_Client www;

#define WEBSITE "www.pawprintnurse.com"
#define WEBSITE "/backend/update2.php?data="
#define FREQUENCY 10000
#define MAXAPS 3
#define SCANTIMEOUT 100

#define IDLE_TIMEOUT_MS 60000

uint32_t ip;

// The library code...

typedef struct {
    uint32_t networks;
    uint32_t status;
    uint8_t rssi;
    uint8_t security_ssidlen;
    uint16_t time;
    uint8_t ssid[32];
    uint8_t bssid[6];
} WifiScanResults_t;

WifiScanResults_t;
```

```
dogtracktest_13_10 | Arduino 1.0.5

WifiScanResults_t scanResult;
int scanResultsCount;

void wipeStr(char *str, int len) {
    for(int i = 0; i < len; i++) {
        str[i] = 0;
    }
}

void bytesToHex(unsigned char *data, int length, char* buffer) {
    const char * hex = "0123456789ABCDEF";
    int i=0, a=0;
    for(i=0;i<length;i++) {
        unsigned char c = data[i];
        buffer[a] = hex[(c>>4) & 0xF];
        buffer[a+1] = hex[(c <& 0xF)];
        buffer[a+2] = ':';
        a+=3;
    }
    buffer[a-1] = 0; //null
    buffer[a] = 0;
}

// Format a SSID Scan result into a Geolocation API Wifi Object
String parseScanResultToJson() {

    uint8_t length = (scanResult.security_ssidlen >> 2);
    uint8_t rssi = (scanResult.rssi >> 1);

    char ssidStr[32];
    strncpy(ssidStr, (char *)scanResult.ssid, length);
    ssidStr[length] = 0;

    char bssidStr[24];
    bytesToHex((unsigned char *)scanResult.bssid, 6, bssidStr);

    //more standard geolocation format
```

```
dogtracktest_13_10 | Arduino 1.0.5

//more standard geolocation format
String json = "{ \"macAddress\": \"\" + String(bssidStr) + \"\", \"ssid\": \"\" + String(json) + \"\" }";
String json = "\"7b+K22macAddressK22K3a+K22\" + String(bssidStr) + \"K22K2c+K22\"";
return json;
}

int startScan() {
    const unsigned long intervalTime[16] = { 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000 };
    wlan_ioctl_set_scan_params(4000, 20, 100, 5, 0x7FF, -120, 0, 300, (unsigned long *) &intervalTime);

    scanResultsCount = 0;
    return 0;
}

bool Next() {

    wipeStr((char *)scanResult.ssid, 32);

    Serial.println(F("Getting next result."));
    isConnected();
    long err = wlan_ioctl_get_scan_results(SCANTIMEOUT, (uint8_t*) &scanResult);

    // Check if we are at the end of the list
    if ( (scanResult.bssid[0]==0x00) &&
        (scanResult.bssid[1]==0x00) &&
        (scanResult.bssid[2]==0x00) &&
        (scanResult.bssid[3]==0x00) &&
        (scanResult.bssid[4]==0x00) &&
        (scanResult.bssid[5]==0x00) ) {
        Serial.println(F("All results found."));
        return false;
    }
}
```

```
dogtracktest_13_10 | Arduino 1.0.5

// Skip the wifi that can move with us
if ( (scanResult.bssid[0]==0x4C) &&
    (scanResult.bssid[1]==0x54) &&
    (scanResult.bssid[2]==0x99) &&
    (scanResult.bssid[3]==0x51) &&
    (scanResult.bssid[4]==0x8C) &&
    (scanResult.bssid[5]==0xC5) ) {
    Serial.println(F("Wifi AP found."));
    return true;
}

// Add the new result if we have one and the list is not full
if (err==0) {
    Serial.print(F("AP found - "));
    Serial.println(parseScanResultToJson());
    // Add a separator if we already have a record
    if (scanResultsCount!=0) {
        Serial.println(F("Sending separator to server."));
        isConnected();
        www.print(F("K2c+"));
    }
    // Send the record
    Serial.println(F("Sending AP info to server."));
    www.print(parseScanResultToJson());
    scanResultsCount++;
    return (scanResultsCount<MAXAPS);
}

// All done for now, but more results to come
isConnected();
Serial.println(F("Scanning for more APs."));
return true;
}

// Setup the hardware
void setup() {
    Serial.begin(115200);
}
```

```
dogtracktest_13_10 | Arduino 1.0.5

// Setup the hardware
void setup() {
    Serial.begin(115200);

    Serial.print(F("Free memory: "));
    Serial.println(freeRam());

    Serial.println(F("Starting wifi hardware."));
    if(!cc3000.begin()) {
        Serial.println(F("Failed to start wifi hardware."));
        return;
    }

    Serial.println(F("Connecting to AP."));
    if(!cc3000.connectToAP(WLAN_SSID, WLAN_PASS, WLAN_SECURITY)) {
        Serial.println(F("Failed to connect to AP."));
        return;
    }

    ip = 0;
    Serial.print(F("Looking up IP address for "));
    Serial.print(WEBSITE);
    while (ip == 0) {
        if (! cc3000.getHostByName(WEBSITE, &ip)) {
            Serial.print(F("."));
        }
        delay(500);
    }
    Serial.println();
    Serial.print(F("Found "));
    cc3000.printIPdotsRev(ip);
    Serial.println();

    Serial.println(F("Setup complete."));

    int scanning = false;

    // The main loop
```

```
dogtracktest_13_10 | Arduino 1.0.5

// The main loop
void loop() {
    if (!scanning) {
        Serial.println(F("Starting scan."));
        Serial.print(F("Free memory: "));
        Serial.println(freeRam());
        startScan();
        delay(5000);
        startSend();
        scanning = true;
    } else {
        scanNext();
    }
}

int scanNext() {
    // Send next result (in Next)
    if (!Next()) {
        // We have finished scanning so finish sending
        finishSend();
        scanning = false;
        Serial.println(F("Scan finished."));
        Serial.print(F("Free memory: "));
        Serial.println(freeRam());

        // Quick hack to reset Tinyduino
        delay(FREQUENCY);
        Serial.println(F("Rebooting."));
        delay(500);
        asm volatile ("jmp 0");
    }
}

// Start sending the data to the server
void startSend() {
    Serial.println(F("Connecting to server."));
```

```
dogtracktest_13_10 | Arduino 1.0.5

// Start sending the data to the server
void startSend() {
    Serial.println(F("Connecting to server."));
    www = cc3000.connectTCP(ip, 80);
    if (www.connected()) {
        Serial.println(F("Connected."));
        www.fastprint(F("POST "));
        www.fastprint(F(WEBSITE));
        Serial.println(F("Header sent."));
    } else {
        Serial.println(F("Connection Failed."));
    }
}

// Finish sending the data to the server
void finishSend() {
    if (www.connected()) {
        Serial.println(F("Finishing sending."));
        www.fastprint(F(" HTTP/1.1\r\n"));
        www.fastprint(F("Host: "));
        www.fastprint(F(WEBSITE));
        www.fastprint(F("\r\n"));
        www.print(F("\r\n"));
    }

    // Read out any returned data for debugging
    while (www.connected()) {
        if (www.available()) {
            Serial.write(www.read());
        }
    }
    Serial.println();
    Serial.println(F("Connection closed."));
    www.close();
} else {
    Serial.println(F("Connection lost."));
}
}
```

```
dogtracktest_13_10 | Arduino 1.0.5

// Encode a string for sending in a URL
String urlEncode(String src) {
    String dst;
    for (int i=0; i<src.length(); i++) {
        if (src[i]!=' ') {
            dst += "+";
        }
        else if ((src[i]>='a') && (src[i]<='z')) {
            dst += src[i];
        }
        else if ((src[i]>='A') && (src[i]<='Z')) {
            dst += src[i];
        }
        else if ((src[i]>='0') && (src[i]<='9')) {
            dst += src[i];
        }
        else {
            dst += "%" + String(src[i], HEX);
        }
    }
    return dst;
}

// Function to show available RAM
int freeRam() {
    extern int __heap_start, *__brkval;
    int v;
    return (int) &v - (__brkval == 0 ? (int) &__heap_start : (int) __brkval);
}

void isConnected() {
    if (www.connected()) {
        Serial.println(F(">>>Connected"));
    } else {
        Serial.println(F("<<<Dropped"));
    }
}
}
```

COLLAR DESIGN

I will order the components required for the tracking collar and design the overall look and design of the collar itself



Case Ideas

I set out designing a few ideas for where they were to be housed. I felt that the design had to be universal for all types of pet, and still had to be attractive enough for owners to want to put it on their pets collar

Clip

The trigger clip was the first attachment that came to mind. This is the most common way to attach extra features onto animals' collars, as there is normally a metal loop in the design. However, this would not be ideal in order to attach the casing of the components. This is because, the GPS and Wifi shield need to remain secure in order to gain the best outcome



Velcro

Velcro will allow the added security to the case as it will be fixed directly to the collar. Even so, I feel that a few issues may still arise from using such a technique. The Velcro itself, over time can become very worn and therefore lose its strength. As well as this, with the animal travelling outside, the weather (rain) can effect the security of the casing. This could therefore be an issue in the long term



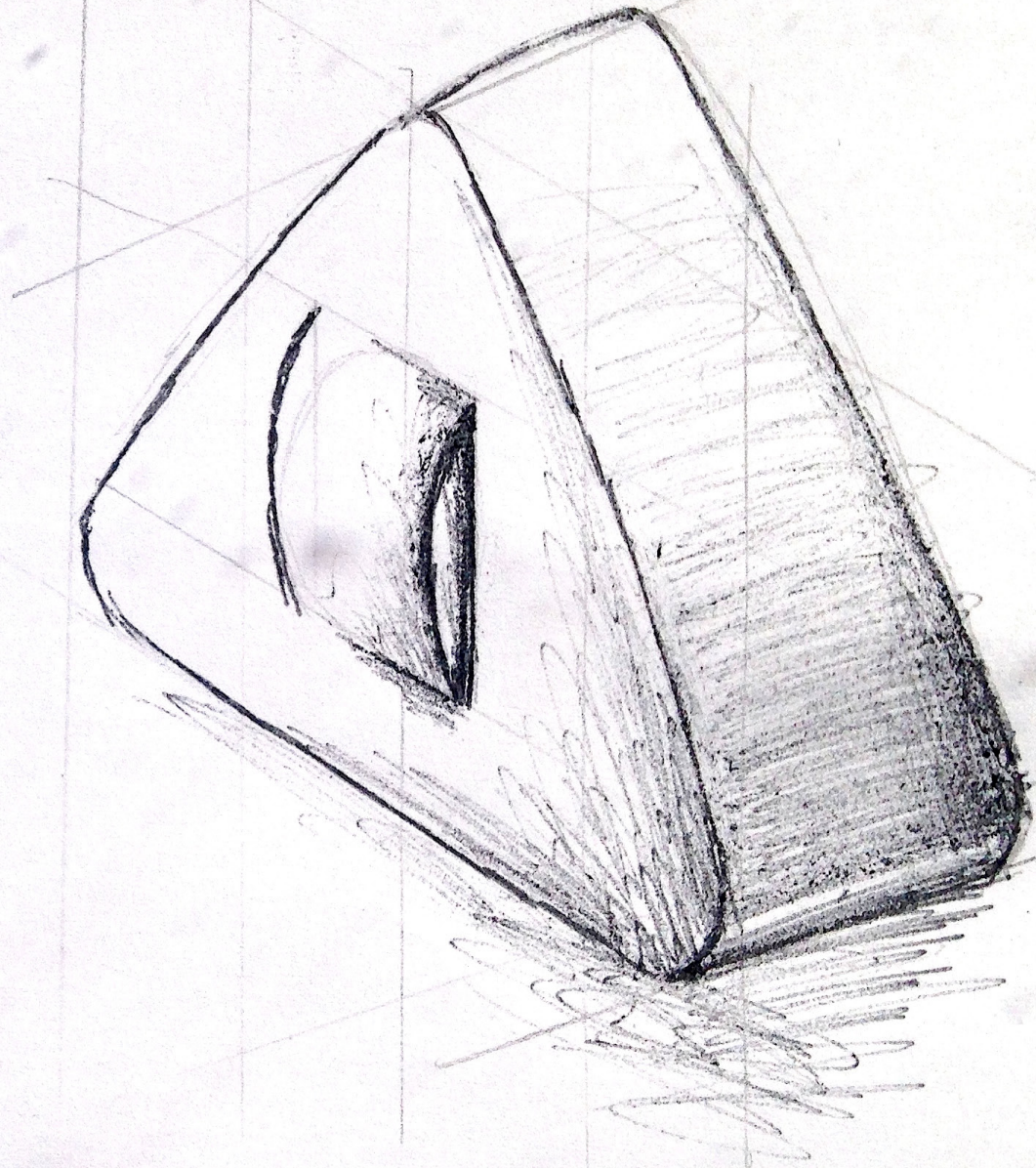
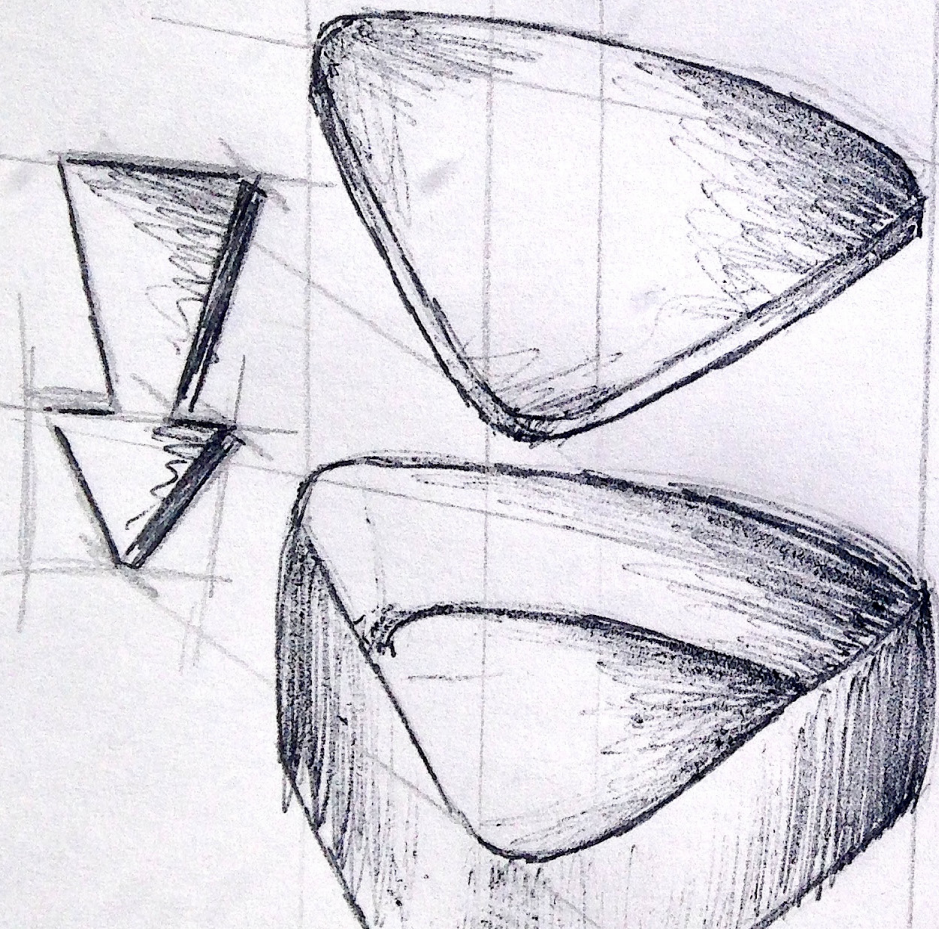
Attach

Finally, the third method I could use in order to attach the case would be through a lip in the back of the case design. This would therefore allow the collar to be simply threaded through into the preferred position. As well this, it will be extremely secure on the collar and won't cause any irritation to the animal. It also reduces the chance of the case becoming loose and potentially getting damaged



Chosen Case

I felt that the most suitable design would be for a rounded triangle shape. This was because it was more appealing than having a simple rectangle, and could allow a better larger design to be put onto the lid. As well as this, I could add more padding around the edges for a more secure casing



container
with clips
in order to
attach



main
collar

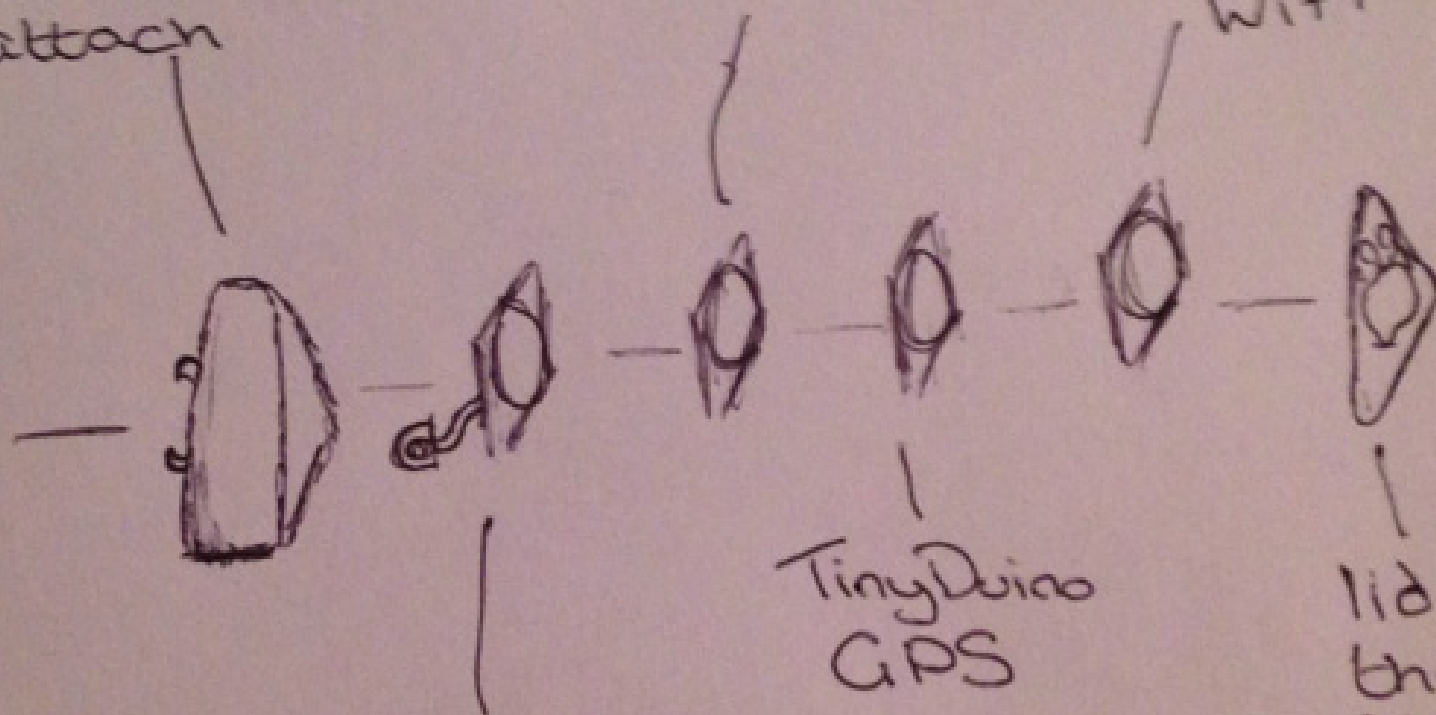
TinyDuino
Accelerometer

TinyDuino
Wifi

TinyDuino
GPS

lid for
the
container

TinyDuino
Processor
Board with
battery
holder



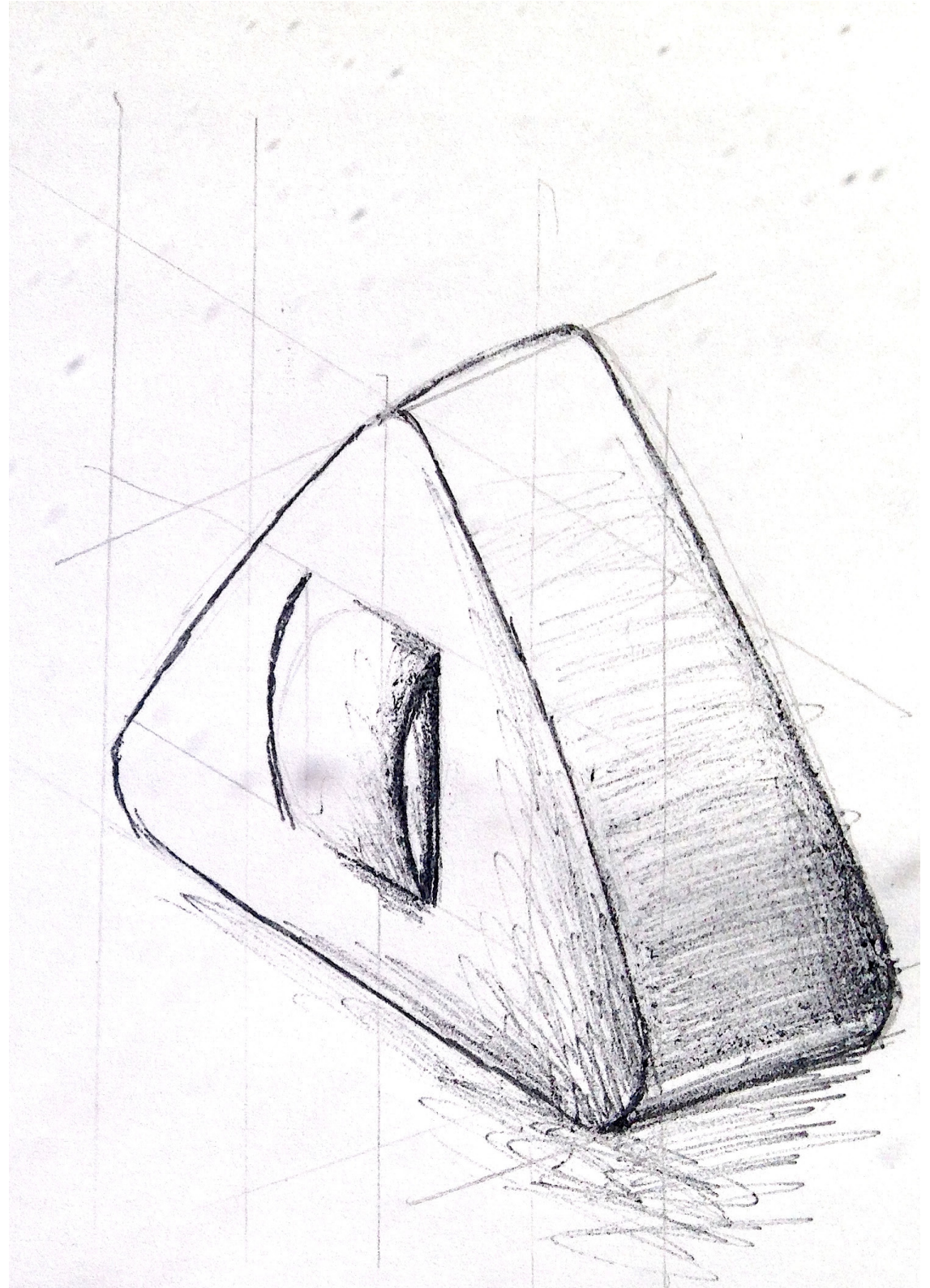
CASE CONSTRUCTION

Now that the components have been combined, I would have to make a case in which to house it whilst on the collar

Design

As shown previously, I have chosen this design in order to act as the case for the components of the tracking collar. Through the use of a triangular shape, the case is much more secure. This is because triangles are proven less likely to deform and are able to balance the stretching and compressive forces inside the structure.

As well as this, a triangle can be a much better economic advantage, as they only have three sides. This requires less materials to make and support, thus minimising all of the cost. Finally, I decided to create the case through 3D printing and therefore would have to construct a 3D prototype design



Creation in SketchUp

1 - I began by creating a square in order to build up the triangle

2 - I removed any unwanted lines of the square to finalise the triangle shape

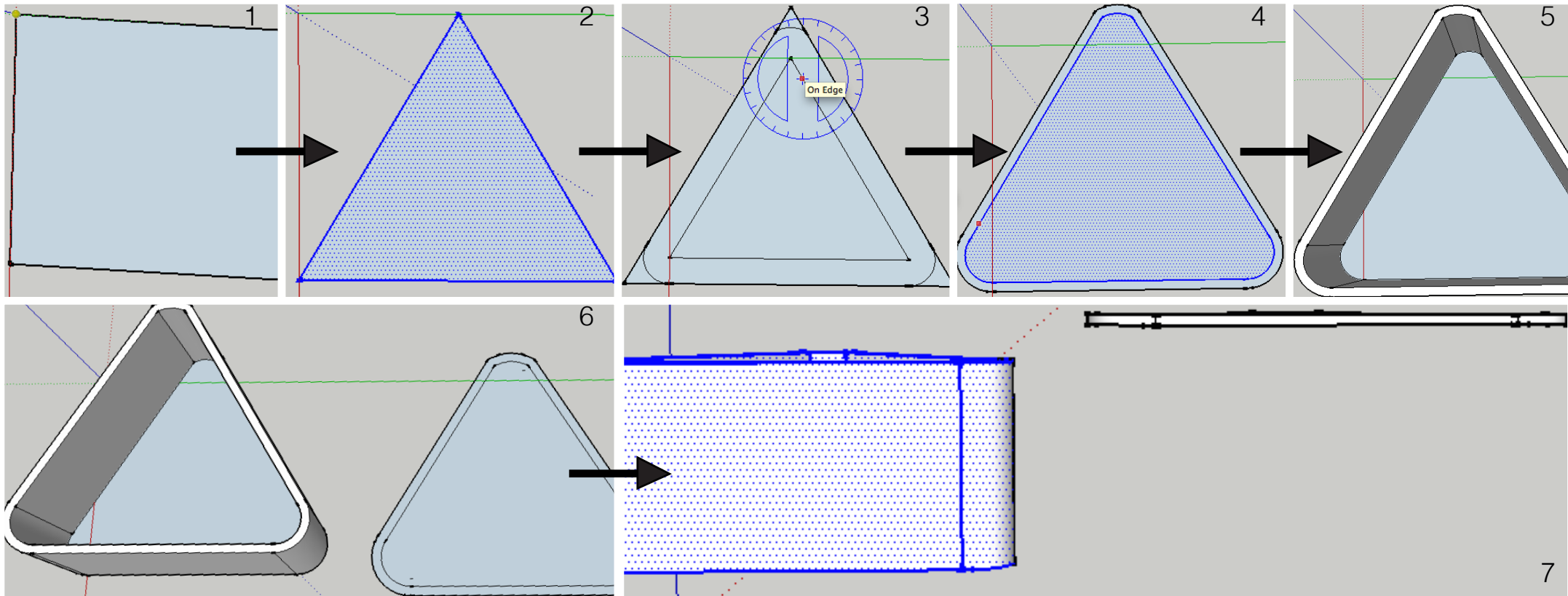
3 - A secondary larger triangle was made to round the edges

4 - The unwanted 'spikes' were removed from the shape

5 - The outer frame was raised to create the outer structure of the case in order to protect the components

6 - I duplicated the base image to make sure my lid was the correct size. I then expanded this new image an extra 1/16" in order to allow a ridge for the user to grab when removing the lid

7 - I raised the lid to the same thickness as the outer frame, in order to add some strength to the structure



Creation in SketchUp

8 - My next stage was to create the hook for the base of the collar case

9 - I measured the dimensions of the arc and created a main structure

10 - I then removed the centre chunk as this was not needed for the loop attachment

11 - the clips for the lid were only small, and therefore had to be accurately measured in

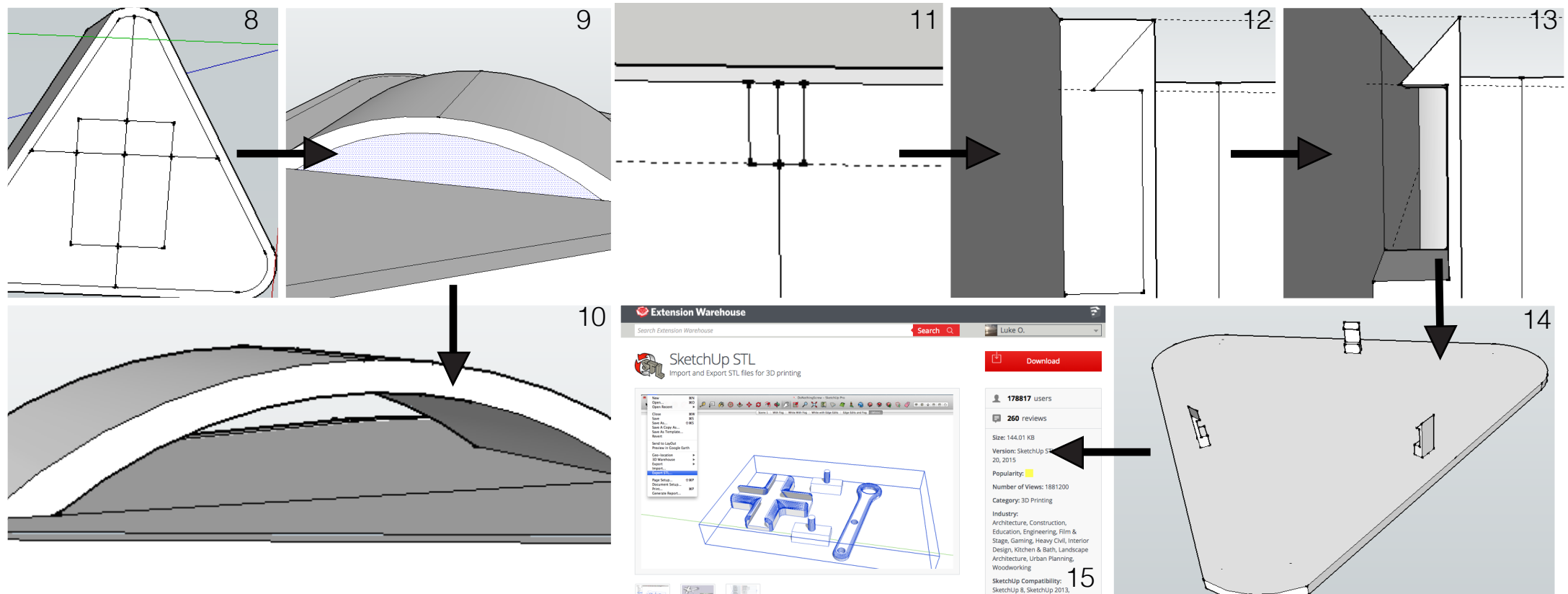
order to work

12 - I created a simple click frame (as found on DVD boxes)

13 - and removed the unwanted sections

14 - I had to make sure that I have to align the clips for the lid in the correct centre area

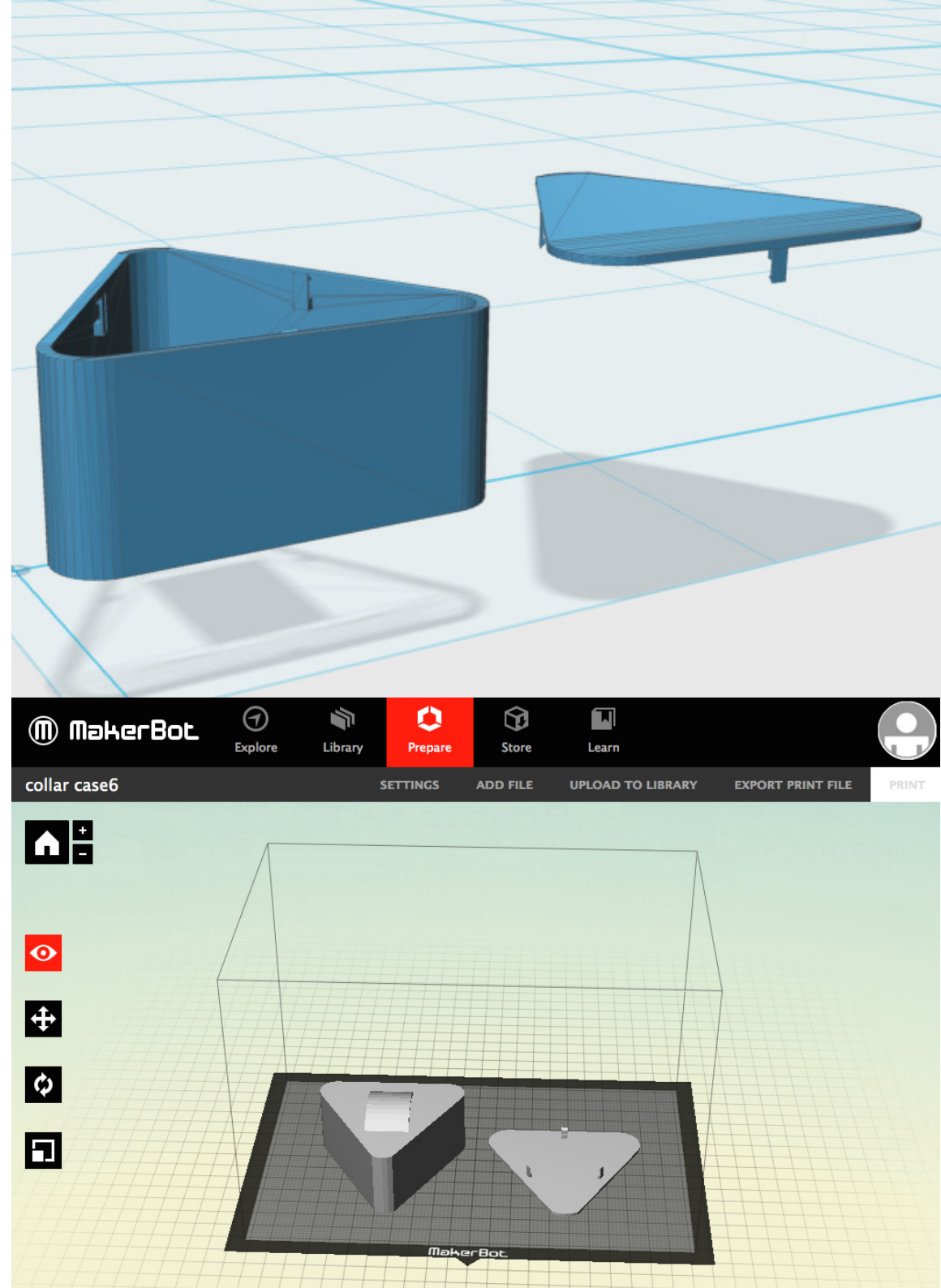
15 - Finally, the file had to be exported in an .stl format in order for 3D printers to use. I had to get an extra download in order to export the sketch up file as an .stl



Testing .stl File

Before attempting to print the case, I made sure that I tested out the .stl file format. This was to make sure that all of the features of the case were still included in the design, and that in the export, no errors or changes were made in the design.

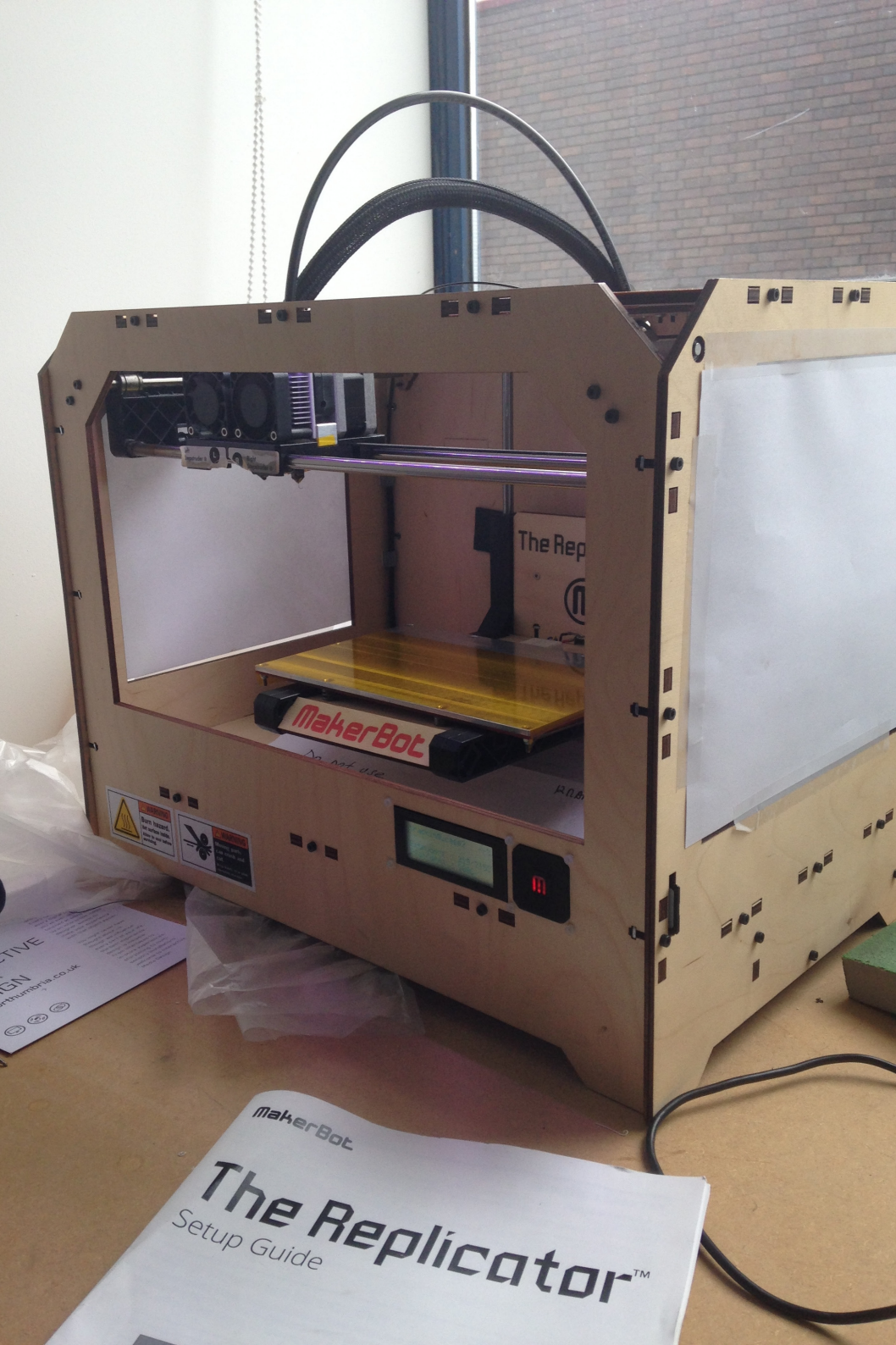
I also made sure that the design would easily fit in the printing tray of the 3D printer (through MakerBot software). This was simply done through the import of the .stl file into MakerBot (who produce the 3D printers). I then re-positioned the different parts of the case so that they were neatly placed in the centre of the printing tray (so no parts could accidentally not be printed)



3D Printing

Now I was sure that the case was suitable to print, I attempted to create it. However, after initialising the print, I came across an issue with the main printer. After the printer had pre heated itself to the set requirements (right tube 217 degrees and the bed 110 degrees), it wouldn't maintain the bed temperature.

This caused a massive issue, as without the bed being at the set temperature, the plastic wouldn't stick. This would result in the arm moving the printed plastic and making the whole final product fall completely out of place



3D Hubs

I managed to find a service known as '3D Hubs' which connects you to all of the 3D printers in your local area. You can then send off your .stl file, and they will decide whether or not they would accept your print.

Fortunately, a local print service round the corner of the university accepted my print. They managed to print the case very effectively and efficiently in order for me to receive the final product as soon as possible


3D HUBS3D PRINTTALKLEARN


Order 995640


1. Pending2. Hub Review3. Payment4. Printing5. Pickup6. Completed


Next action: Your order is being printed.
The Hub will let you know as soon as the print is finished. If you have any questions for the Hub in the meanwhile, you can add a comment to the order below.


Product details

COLOR
white

MATERIAL
PLA

RESOLUTION
High

DELIVERY
pickup

USE CASE
Prototype

DESCRIPTION
Case for a uni project I am making for my final year doing interactive media design. The case will hold a gps tracker to be used on dog collars

Chompwork

★★★★★

MORE

Daan van H

Hi, I'm your 3D Hub for this order, if you need to just place a comment on this page. If you need to contact me you can contact me via Heugten by

Common Questions

What is the delivery time?



INNER CASING

Whilst the main case would protect the components from large physical damage, there is still some protection needed on the inside



Sponge

I felt that the best way to protect my components in the case was to use sponge. This was because it would be able to absorb any shock which the case would under go, as well as appear much more professional.

However, the main issue I found was that black sponge is extremely difficult to come by. I searched a large number of craft shops throughout Newcastle, and none of them were able to supply me with any. Instead, I had to make do the sponge from some paint brushes. This was the right material I needed, and I collected a variety of sizes to fit the needs of the case. I first began by removing any plastic from the inside of the brushes

Base Layer

I took one of the larger sponge brushes from the set in order to create the base layer for the case. This was simply done by holding the sponge above the main frame of the case, and trimming the edges in order for it to fit inside.

I then neaten up the sides of the sponge in order for it to appear more professional. This would also allow it to slot easier into the case, and reduce the risks of it moving around or coming lose in the future





Component Frame

I then placed the components in the centre of the of the case, and measured the remaining area which surrounded them. This assisted in the cutting of the remaining sponge, as I was able to make a correct and secure fit for the actual tracking device.

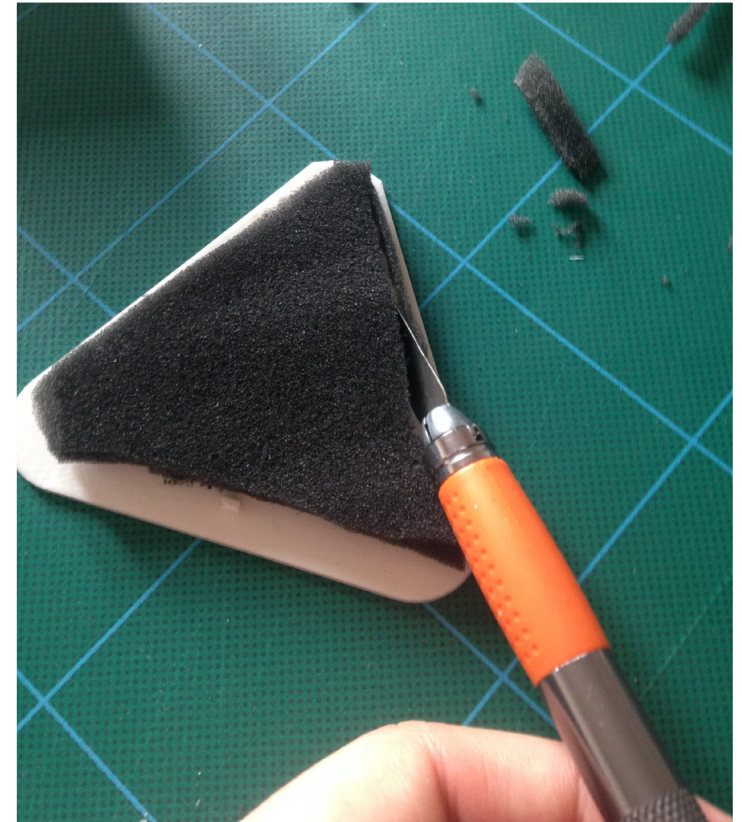
With the device safely contained between the sponge, I tested to see how secure it was. However, just with some slight movements, the components managed to shake themselves free from the sponge and kept hitting against the lid. This inspired me to add another safety layer onto the base of the lid, in order to make sure that the components firmly stayed in place

Sponge Lid

As with creating the base layer, I began by using one of the larger sponges. I cut the sponge to a rough shape, as close as I could to that of the inside of the frame. However, instead as with the base layer, I had to make sure that I cut the sponge in half (thickness). This was to make sure it wasn't crushing the TinyDuino. I then began by making sure that the sponge lined up in the centre of the case.

This was to prevent the clips from being blocked and made sure that the lid could still be safely secured. In order to keep this in place, I used super glue to fasten it down.

Finally, I made sure that the edges of the sponge were neat and double checked they didn't cover the clips in anyway). To do this, I simply went along the edges with a craft knife to trim off the excess.



WEBSITE

For all of the information to be clearly displayed; as well as improving the brand image for PawPrint, I would have to begin constructing the website

Branding

Instead of focusing towards a website that targets the nurses (as I did in my previous semester), I decided to focus all attention onto the promoting of PawPrint itself. Therefore, in order to do this, I would have to create a site that would be able to give information about PawPrint, as well as sell the products which are available

I would therefore have to make a website that would be compatible for all devices (including mobiles)



Wordpress

In order to create the website, I decided to use wordpress. This is because it would be a lot easier for me to construct and manage all of the individual pages. As well as this, the hosting in order to make the webpage live is much simpler to set up

Another benefit of using wordpress in the creation of my website, is that I would be able to incorporate a theme. This would contain a pre-set design in which I could easily edit and personalise to make my own. As well as this, the theme would already be compatible for other devices, allowing anyone to access the site on the go

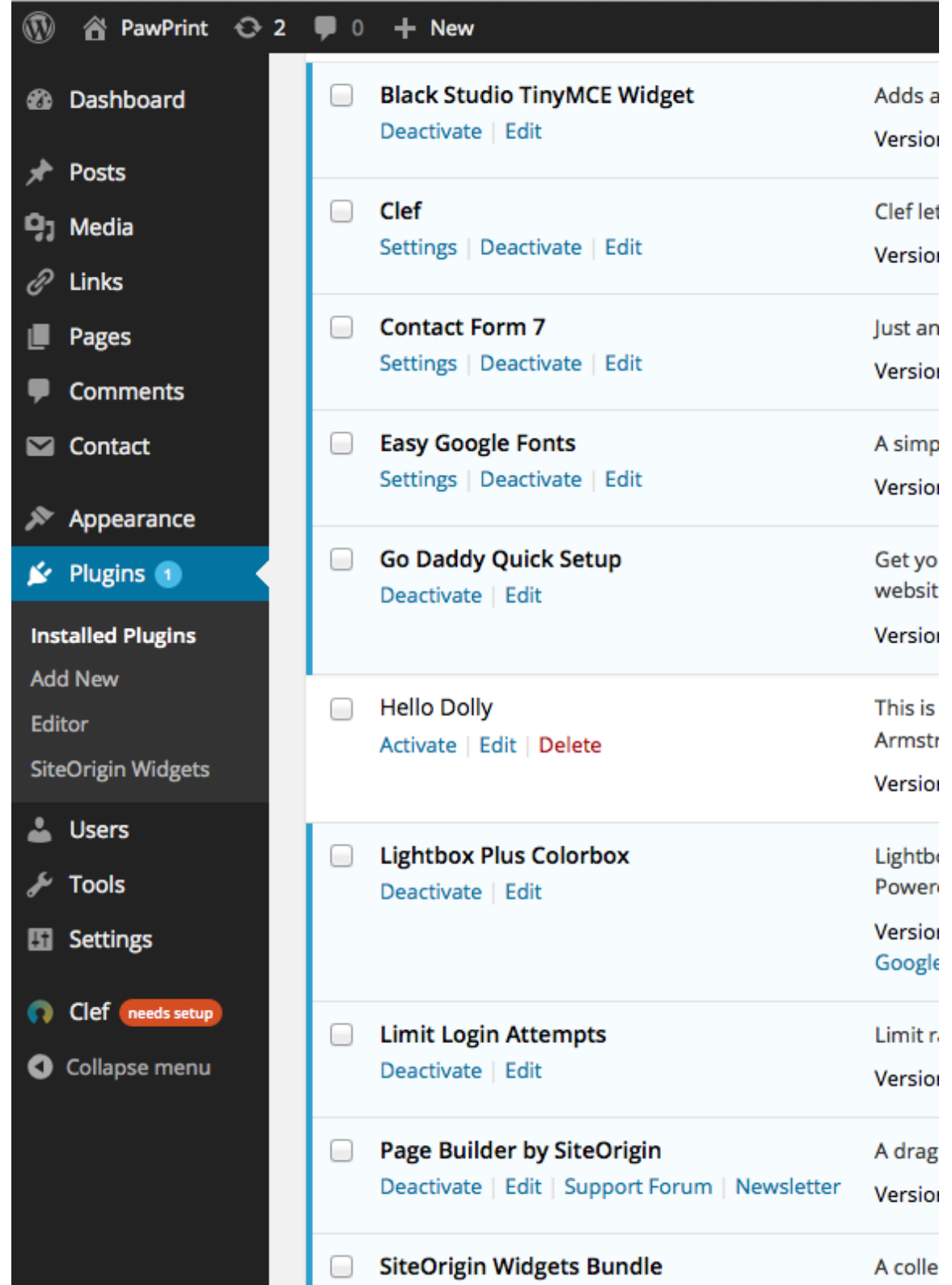
The image shows two screenshots. The top screenshot is a Godaddy order confirmation page for 'Luke'. It displays order details (Order Number: 820445441, Order Total: £3.59, Order Date: 21/04/2015), account information (Luke O'Keeffe, lukeokeeffe@hotmail.co.uk, 24 Wentworth Avenue, Manchester, M41 9DW, United Kingdom, +44.7955464458), and a 'Need help?' section with support contact information. A large green banner below the order details says 'It's go time! Get started using your products now.' and features a 'Managed WordPress' button. The bottom screenshot is the WordPress dashboard. It shows a sidebar with navigation links (Home, Site Stats, My Blogs, etc.) and a main content area with sections like 'At a Glance' (showing 1 Post, 1 Page, 1 Comment), 'Quick Draft' (with a title field and 'Save Draft' button), and 'Activity' (showing a recently published post 'Hello world!').

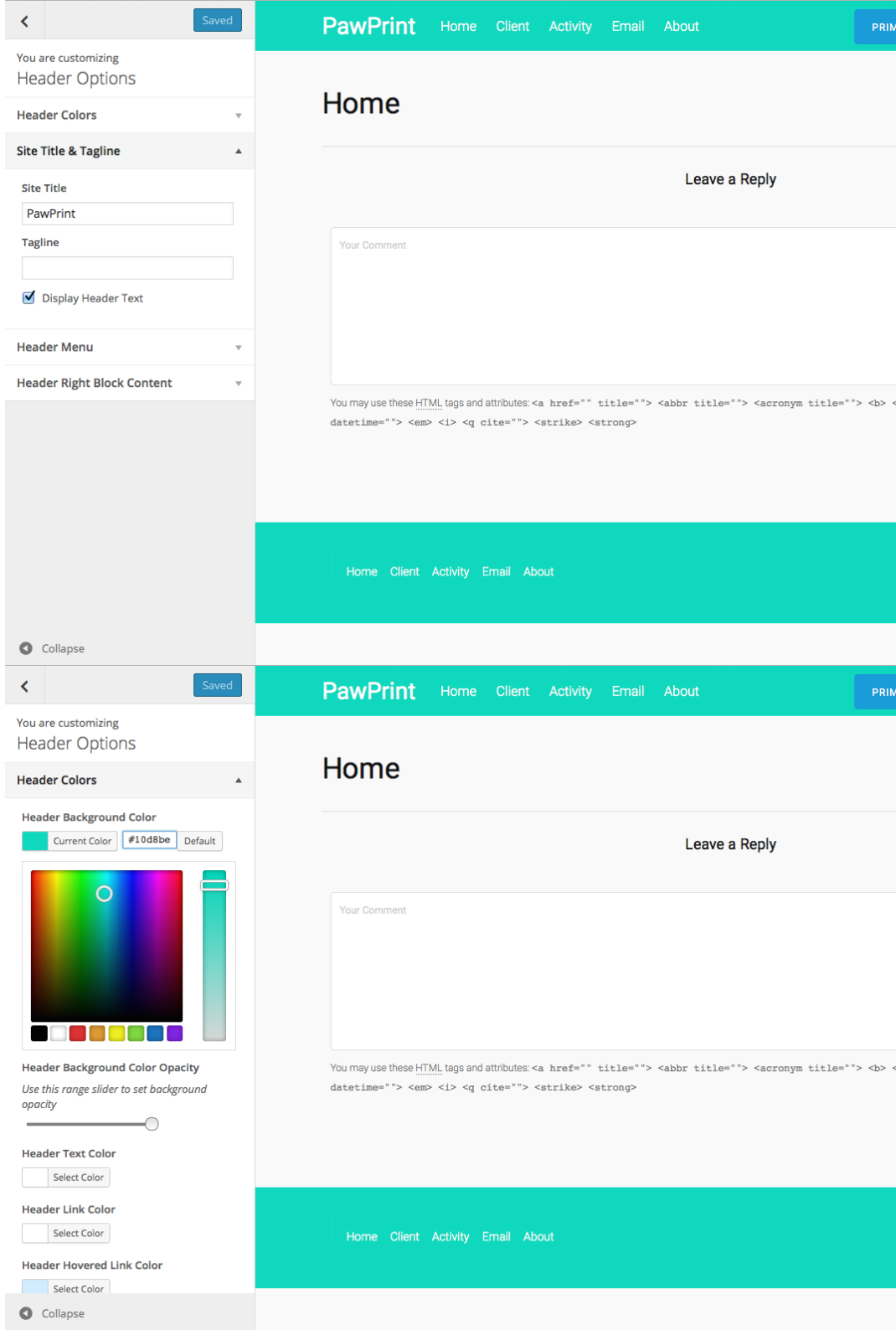
Plugins

To assist with the construction of the web pages, I downloaded a selection of plugins. These would help to reduce the amount of time in which I would spend in re-deigning each of the pages. As well as this, they reduce having to input the code manually in order to create the chosen effect/position of the images or text which I wish to include

The plugins I downloaded are:

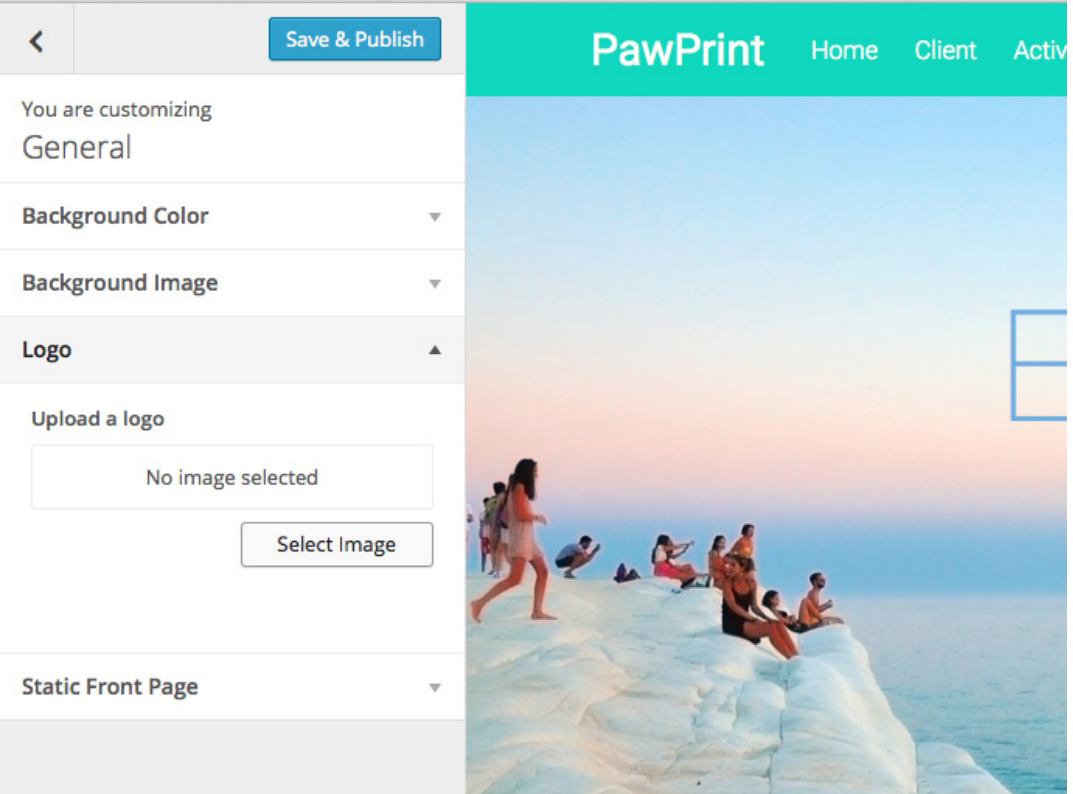
- Page Builder by SiteOrigin
- SiteOrigin Widgets Bundle
- Visual Editor by BlackStudio TinyMCE Widget
- Contact Form 7
- Easy Google Fonts
- Lightbox Plus Colorbox
- Spacer
- Title Remover





Editing the theme

As stated before, the benefit of using the theme is that I could easily customise the existing layout in order to make it my own. This was simply done through the selection of the customise section in the options bar. Here I was able to change the border colour to match the PawPrint brand, as well as change the main header to PawPrint itself



Choosing logo

Again, as with the editing of the main layout, I could easily input my own logo into the design. This would therefore allow the PawPrint logo to be uploaded onto the main page and act as the home button, as well as advertising the brand itself.

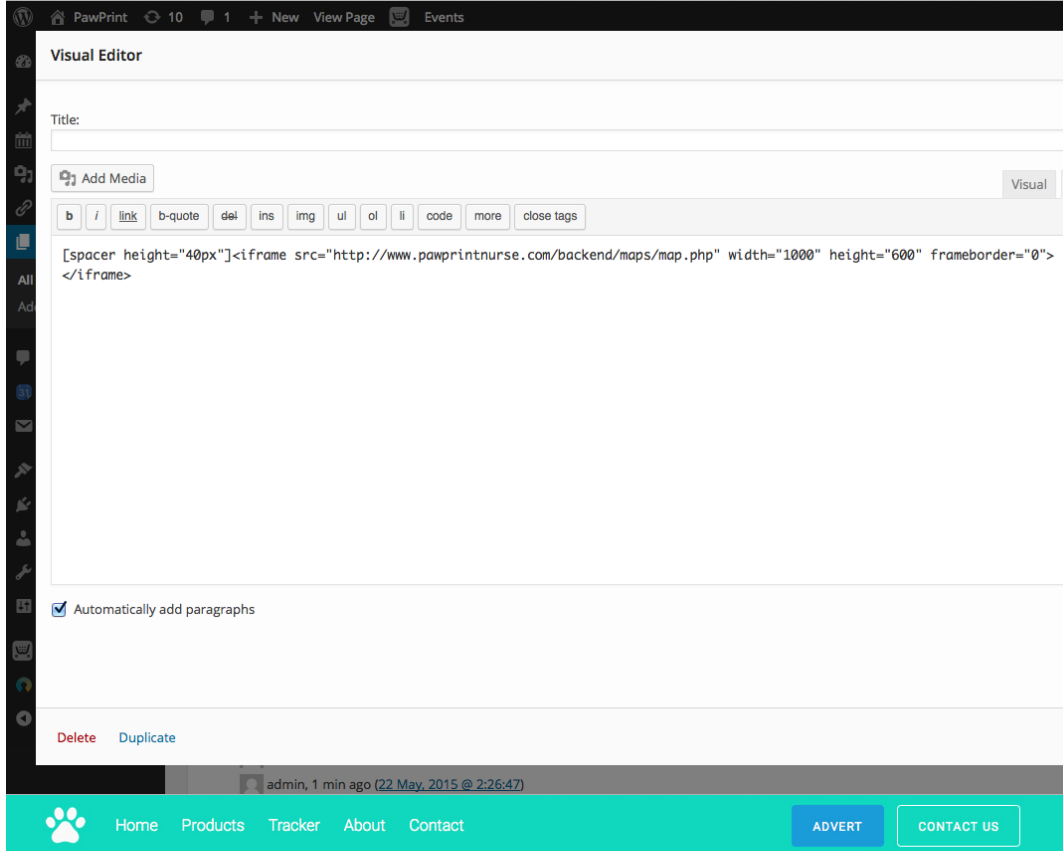
The logo was very simple to upload and would appear on each of the pages as the main header did not alter





Images

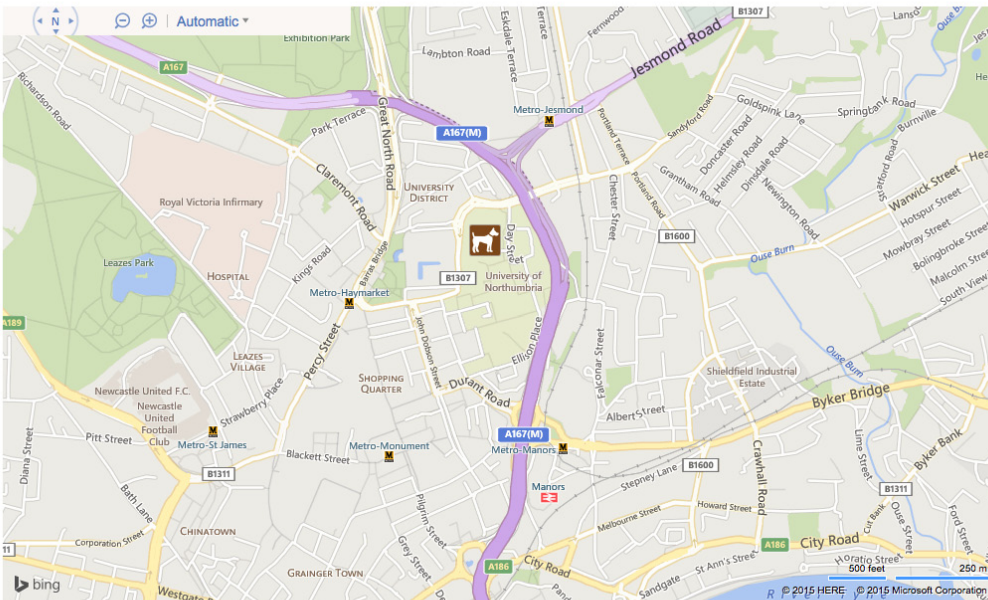
Majority of the images in which I used throughout the design of the website, were acquired from Unsplash. This is a website that allows users to download and use photos with completely free licenses. I felt that images which were displayed on this site were a much higher quality than the ones I had produced. Therefore, I took advantage of the fact that all of the images are completely free to use, and felt they would be perfect to use on my own webpage



Tracker

As I had already tested the tracker to make sure that it was up and running, I had to find a way in order to display the actual map. There was a very simple way in order to do this, that involved incorporating the update map from the php documents, onto the page.

In order to do so, I would simply have to use a “iframe” in the main text of the page set up. Basically, this implants the details of the other web page (in this case the map) into the site, and displays a fully interactive version of that web page



[My Sales](#)[Catalog](#)[Promotions](#)[Settings](#)[Products](#)[Categories](#)[Product Types](#)

Products

[All Products \(3\)](#)[* New product](#)[✎ Modify](#)[✖ Delete](#)[📄 Import products..](#)

| <input type="checkbox"/> | SKU | Name ^ |
|--------------------------|----------------------|-------------------------------|
| Filter | <input type="text"/> | <input type="text"/> |
| <input type="checkbox"/> | 0000000 | GPS Tracker |
| <input type="checkbox"/> | 0000001 | PawPrint App |
| <input type="checkbox"/> | 0000002 | PawPrint Lead |

GPS Tracker

[New product](#)[Duplicate](#)[General](#)[Attributes](#)[Options](#)[Gallery](#)[Files](#)[Tax and Shipping](#)[Related Products](#)[Embed product](#)**Name**

GPS Tracker

SKU

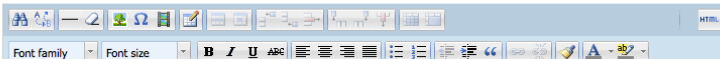
0000000

Weight, kg

0

☒ This product requires shipping ⓘ☒ Save (Ctrl+S)☒ Save & Close[Preview product](#)**Price**

£ 25.00

AvailabilityEnabled [[Disable](#)]**Stock Control**In Stock (∞) [[Manage](#)]**Description**

The PawPrint GPS Tracker is a new way in order to look after and care for your pets. The very simple attachment is easily and securely connected to your pet's collar. The miniature Wifi shield inside uses the load servers to triangulate the current loaction of your animal. This is then sent directly to the PawPrint app in order for you to discover your animals whereabouts.

The casing itself is made from a lightweight yet durable plastic to protect the contents. It has been designed to be small enough, as to not irritate the animal in any way. If you have any more questions or concerns about our products, please do not hesitate to get in contact.

Path: p

[Categories](#)[Manage categories \(global\)](#)

Products

The way in which I made all of the products available was to incorporate an Ecwid online shop onto the webpage. Basically the online shop would display all of the products which PawPrint had to offer, and give information on each one. It would then allow the user to choose whether or not they would want to purchase this product, or any other of the ones found on the webpage.

The store is a template made online in which you simply insert to any web page which you would like. You just have to add the information and images you would wish to use online, and this will automatically update in your store

USER TESTING

After all separate features were all individually up and running, I decided to test them to see if they would work

A video of the testing can be found on youtube at:
<https://youtu.be/7AxnmVDRTFY>

Tracking Collar

The way I made sure that the collar was fully functioning, was through doing a test on myself. I decided to gain assistance from one of my friends, Sam Turpin, in order to complete this test. Basically, I decided to take the collar, and travel to a random point in Newcastle from my flat. It would then be up to Sam, to try and find me, to determine if the collar gave an accurate enough reading.

As shown in my video, I travelled 19 mins away from my original location. The collar managed to track me as best as possible throughout this time, giving an accurate enough reading for Sam to eventually come and find me. Therefore, the test proved a success, meaning that the collar was fully working. However, a day after completing the test, The processor board was not functioning (possibly due to memory). Unfortunately, this meant that the tracker had crashed and would need some serious work to resolve for the future



```
dogtracktest_13_10 | Arduino 1.0.5

dogtracktest_13_10 $
#include <Adafruit_CC3000.h>
#include <SPI.h>

#define ADAFRUIT_CC3000_IRQ 2
#define ADAFRUIT_CC3000_VBAT A3
#define ADAFRUIT_CC3000_CS 8

Adafruit_CC3000 cc3000 = Adafruit_CC3000(ADAFRUIT_CC3000_CS,
    ADAFRUIT_CC3000_IRQ, ADAFRUIT_CC3000_VBAT,
    SPI_CLOCK_DIVIDER);

#define WLAN_SSID      "*LIFE GUARD FOR HIRE*"
#define WLAN_PASS      "caroline30"
#define WLAN_SECURITY  WLAN_SEC_WPA2

Adafruit_CC3000_Client www;

#define WEBSITE        "www.agm.me.uk"
#define WEBPAGE        "/unn/dogtrack/update2.php?data="
#define FREQUENCY      10000
#define MAXAPS          3
#define SCANTIMEOUT     100

#define IDLE_TIMEOUT_MS 60000

uint32_t ip;

Problem uploading to board. See http://www.arduino.cc/en/Guide/Troubleshooting
Binary sketch size: 24,544 bytes (of a 30,720 byte maximum)
avrduide: stk500_recv(): programmer is not responding

13 Arduino Pro or Pro Mini (3.3V, 8 MHz) w/ ATmega328 on /dev/tty.usbserial-1
```


Dog

To make sure that the case would work with an actual dog, I enlisted the help of another friend of mine, Lauren Muxworthy. She allowed me to test my prototype case on her dog Rosie, in order to see whether or not it would survive everyday use.

The test proved better than expected, as Rosie really tested the durability of the case. It stayed very secure during the time she was running around the field, and wasn't damaged when she ran through the brambles. As well as this, the test went a little further than anticipated, as she rolled the case in fox poo. Even though this wasn't planned, it showed that the case could survive the full weight of a large dog, and could easily be cleaned after use



EVALUATION

Now that my testing was complete, I
would be able to evaluate my project

Evaluation

Overall, I am very pleased with the outcome of my final project. I felt that I have demonstrated a large range of abilities and methods throughout this design process; and I have been able to tackle some of the most difficult situations in my field of work. I am disappointed however, that the case took me so long to get to work, and then has malfunctioned towards the end of the project. Luckily, I did manage to gain the testing and proof of work on video.

The complexity of the actual tracking collar, has prevented the incorporation of some of the proposed features (i.e. the accelerometer), yet I feel that with more time I could have been able to try and resolve this matter. I have tried to make the brand PawPrint appear as professional as possible, and I feel that this is confirmed through the website design. The use of a fully functioning market online, has allowed the opportunity to really develop the brand further, potentially producing more

products in the future.

I find that I am very pleased with the end result of my work, and that I have managed to meet the requirements I set myself to the best of my ability. Clearly, if I was to develop the collar further, I would reduce the overall size, and look further into other technologies that have much larger memory. Even with all of the issues faced, I am proud that I have managed to produce a final result, and look forward to continuing the brand development in the near future