# Arduino Anti-theft Vehicle Device / AVAD Technical Manual


## Author: Aidan McGivern

**Supervisor: Stephen Blott**

**Student number: 15414228**


**Date of completion: 19/05/2019**

# 0.Index

# 1. Motivation

## 1.1 Background

I have always had an interest in cars and the computers that control them. I have had my own car for the past three years and have always tried to update it as it is sixteen years old. I have always tried to bring some present day technology into the car to make it more comfortable. While I have implemented some changes this is the biggest technological upgrade to date.

## 1.2 Idea conception

It wasn't until my friends brothers car was stolen, which was worth quite a lot of money, I thought wouldn't it be handy if you could have a tracker in your car that you wouldn't have to pay extortionate money for and wouldn't have to be put on a data plan that would cost at minimum twenty euro a month. It was then that I had the idea of a text based tracker that would simply sit in the car and trigger if stolen and text the owner.
I had no idea where to start so I left the idea until I had to come up with a fourth year project and decided to take the plunge on the project and use an arduino with multiple modules in order to achieve my goal. While I had never heard of anyone I knew actively looking for a product like mine, anytime someone hears about it they're very interested in it and look to acquired it for their own car.

# 2. Research

## 2.1 Background

Before this project I had no previous experience in using an arduino or coding in C++. As a result I spent a long time researching to first make sure that what I was trying to achieve could be done. From my research I was certain that I could accomplish what I was setting out to do.

## 2.2 Initial concerns

As I had never used an arduino before I did a lot of research to see how I could make sure that boards would be compatible with each other. I saw that different boards worked at different voltages and certain boards wouldn't work with certain arduinos, so I needed to make sure that the boards would work well together.

## 2.3 Hardware

- GSM board:
  With the GSM board I had to ensure that the board would accept a 2G sim card. This was my main concern as without being able to use the 2G network the price of the board would be too expensive. I eventually settled on an adafruit FONA GSM board which I ordered.
  In order for the board to function, it required a battery to allow it to pull extra voltage as well as the main power line.

- Bluetooth board:
  I decided to go for an RN-42 bluetooth device as it seemed to come highly rated and used in most arduino projects that use bluetooth.
- OBDII board:
  There was only one OBDII arduino board that I could find, unfortunately I was unable to acquire it due to it only being available on one site and it was out of stock. Instead I opted for an OBDII click module which seemed like it may work for my particular application.
- GPS Board:
  The GPS board that I decided to go for was the adafruit Ultimate GPS board. This board was used in many different projects and seemed to have a lot of documentation and videos about it on the internet.
- Arduino:
  Finally the last piece of hardware I had to pick was which arduino to buy. I found out pretty quickly that a normal arduino uno would not do as it only has one serial port. As a result I had to buy an arduino mega which has four serial ports to allow me to interface with each of the boards I had bought.

# 3. Design

## 3.1 Arduino Diagrams

My original arduino diagram started off like this:

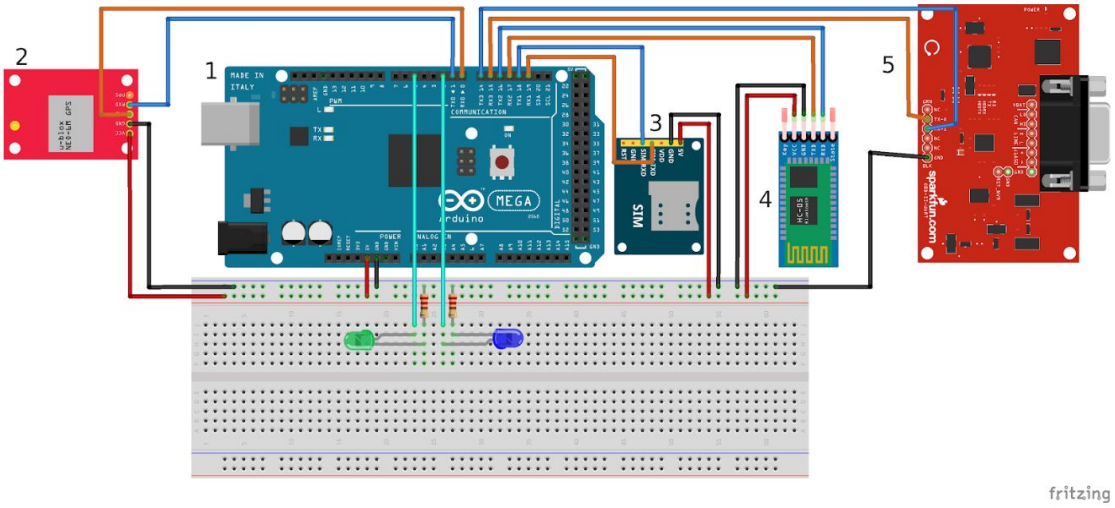


Fig1

My current configuration looks like this with each diagram for each module labeled on the device:

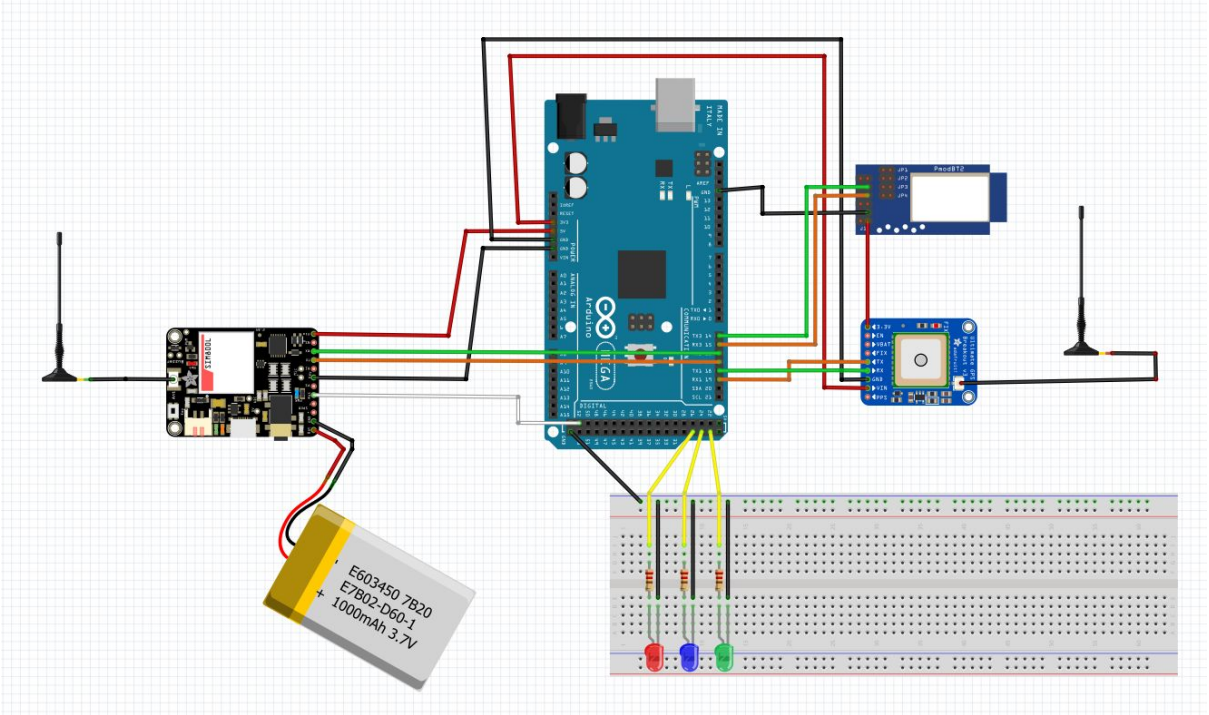


Fig2

The first diagram has evolved now while being quite similar it now has omitted the board marked ‘5’, the OBDII module and now includes a battery which is connected to the GSM module. I have slightly changed the wiring as the GPS required its own power to ensure it got a fix more reliably. The GPS module has a pin that provides 3.3v of power so I have instead used this to send the current to the bluetooth module. I have added an extra LED and I have also added a wires to each module and the arduino to enable the arduino to reset the modules.

## 3.2 Code

### 3.2.1 Diagrams

#### 3.2.1.1Use case diagram

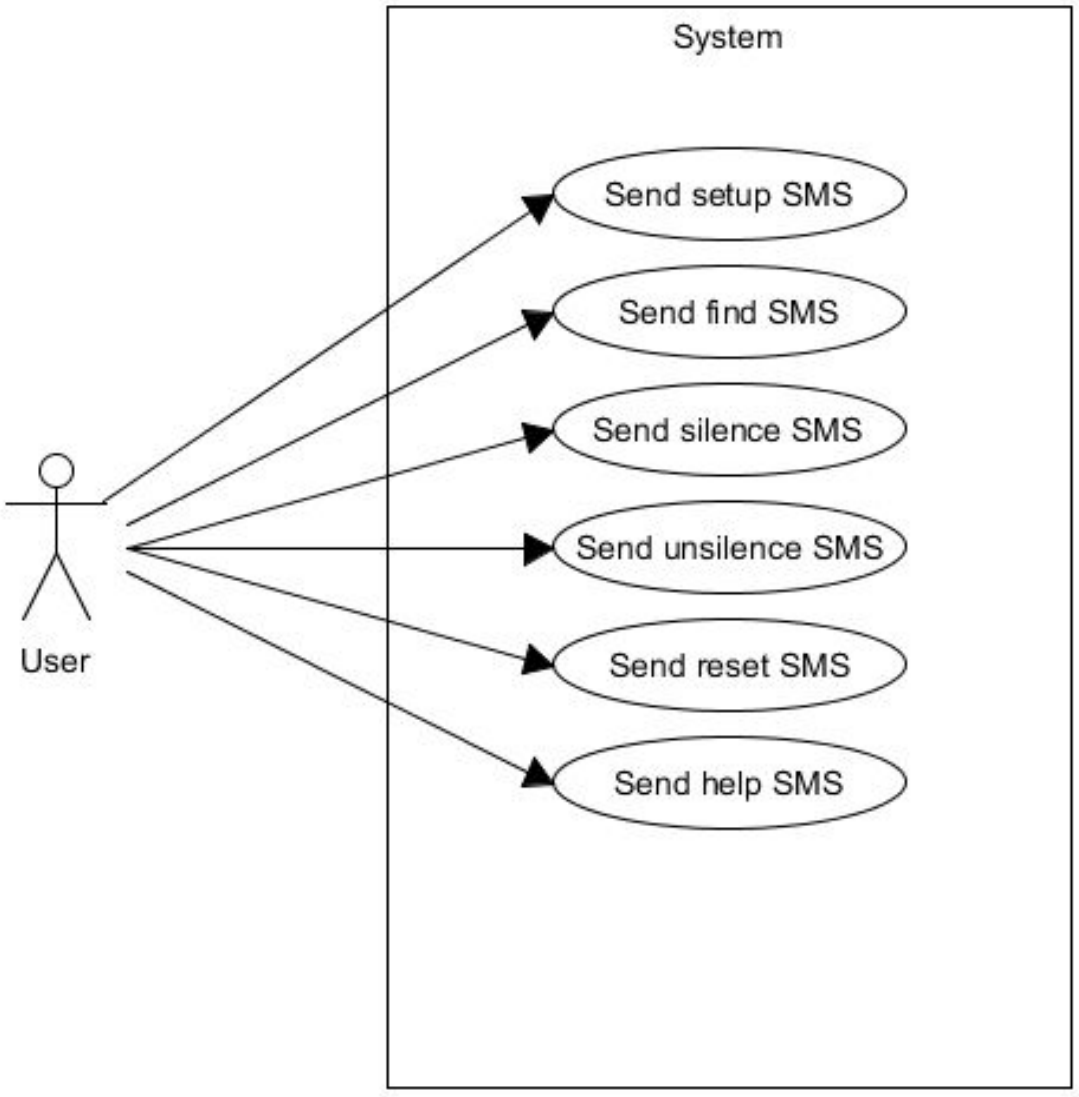


Fig3

### 3.2.1.2 High level architecture diagram

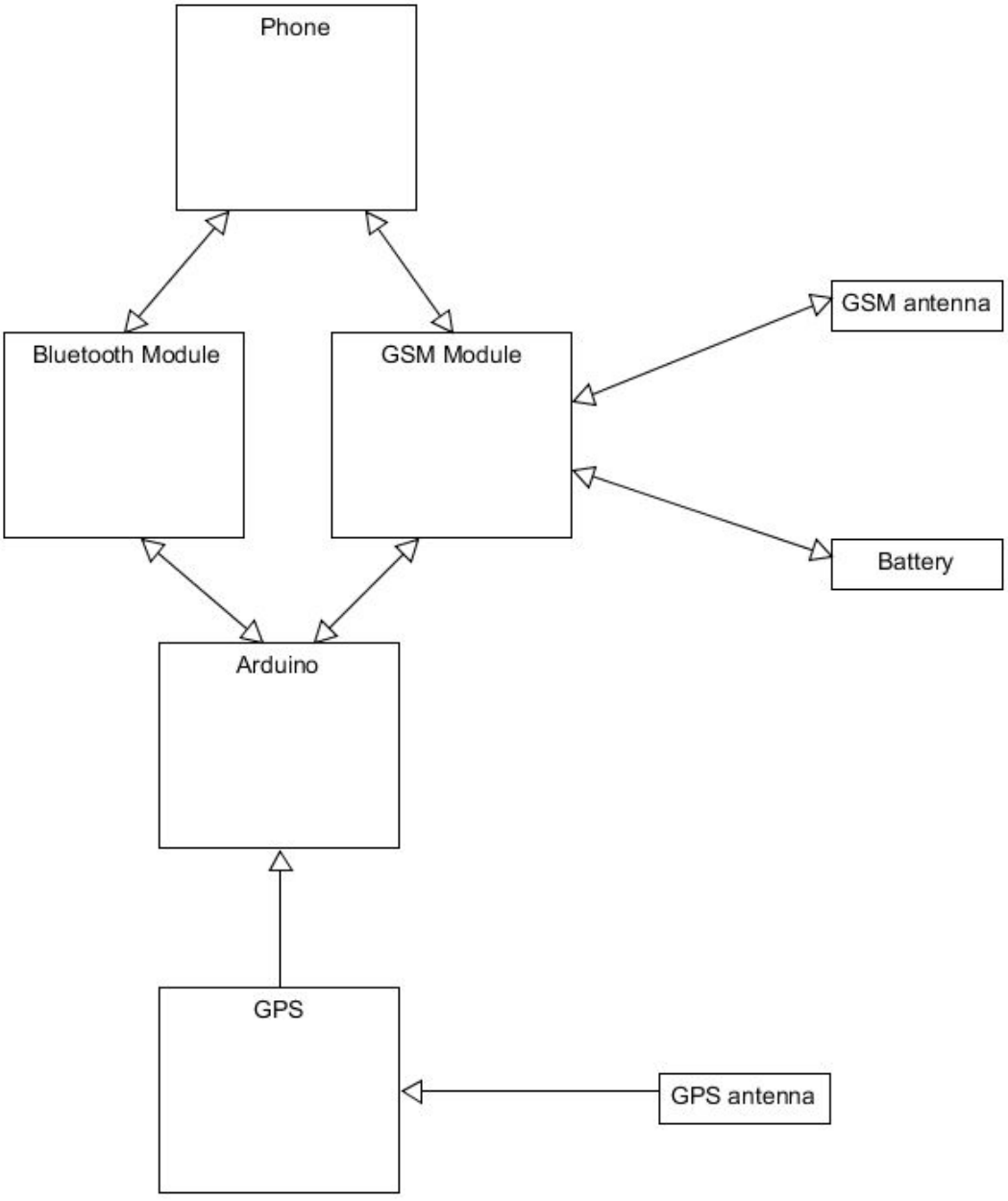


Fig4

### 3.2.1.3 High level context diagram

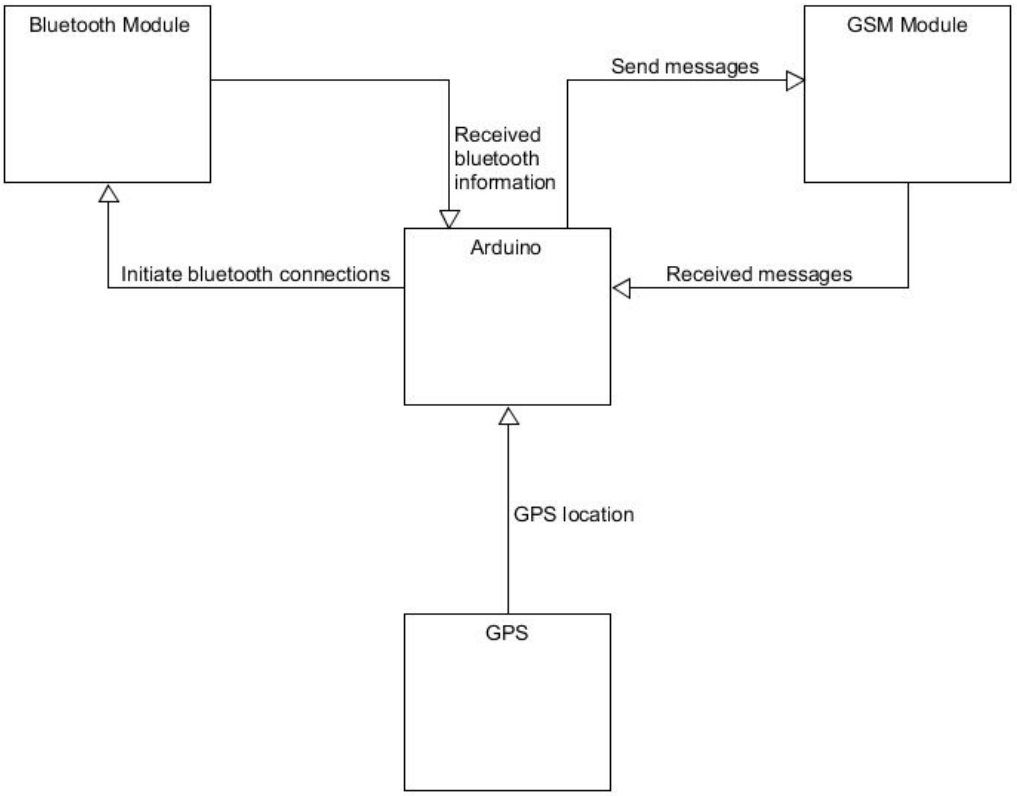


Fig5

### 3.2.1.4 Data flow diagram

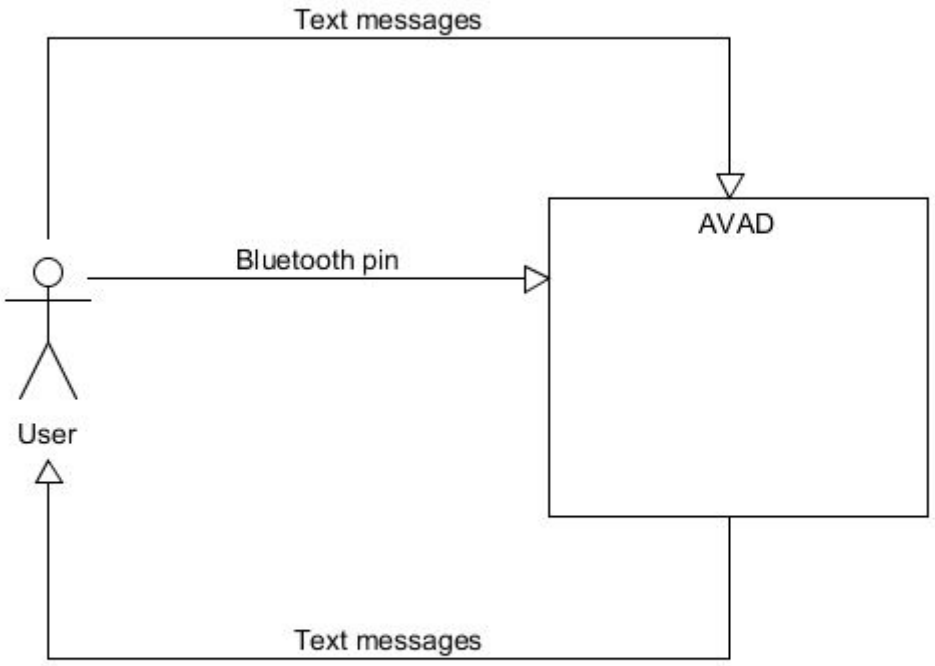


Fig6

### 3.2.1.5 Flow Diagram

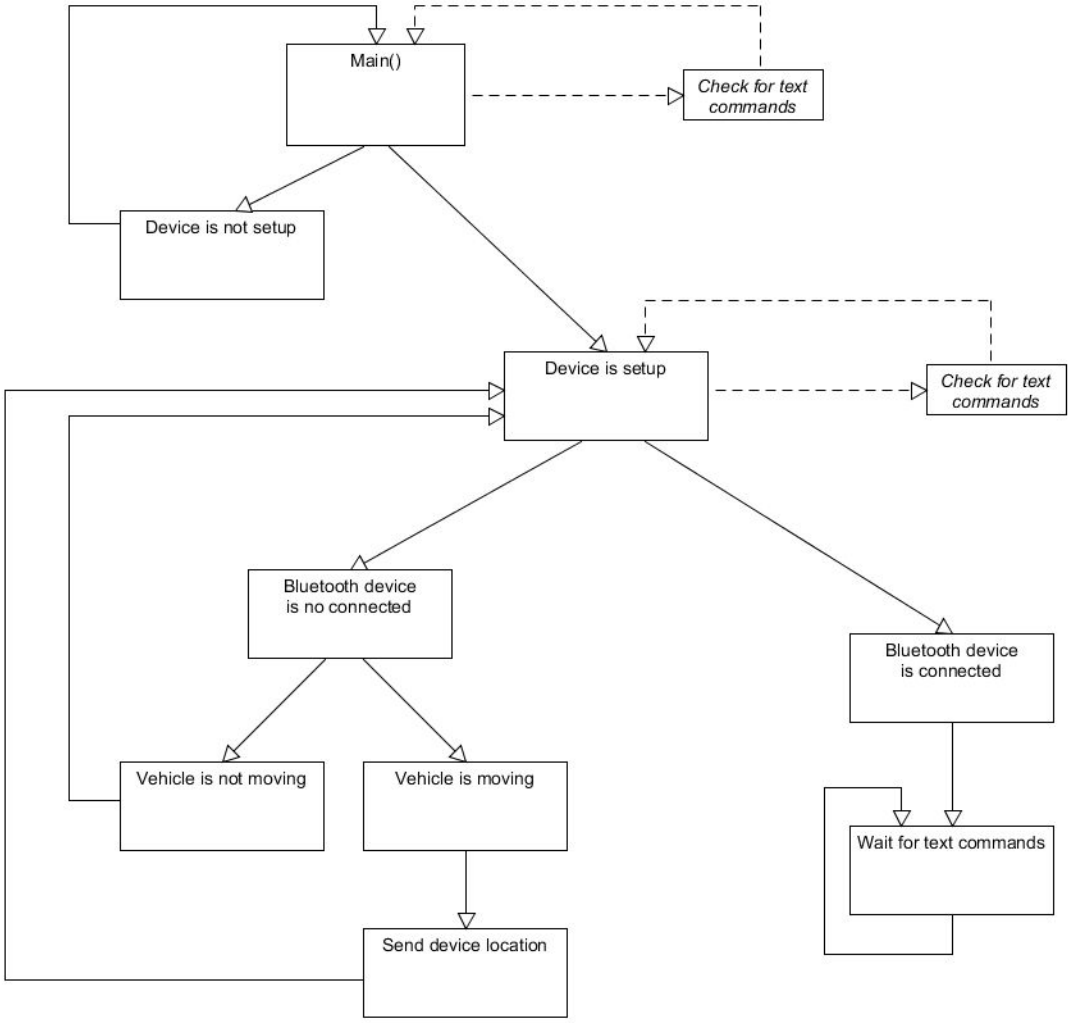


Fig7

### 3.2.2 Modes

I designed my code in a procedural method. The code consists of one main function that loops. In this loop I have three if statements which determine which mode the device enters. The three modes/states are:

- Not setup
- Setup and bluetooth not connected
- Setup and bluetooth connected

The deciding factor of the if statements lie in global variables that are changed by the functions of the code as it runs.

I split up the code to group together all functions for each individual module to make it more readable and more maintainable.

### 3.2.3 Main functionality

The device starts in setup mode where it waits for the user to send a text with the string ‘setup’ followed by a four digit pin. Once received the device is considered setup and stores the phone number of the user, the pin code and the mac address of the users bluetooth address and writes it to the EEPROM for future use. The device now checks if the correct bluetooth connection is present if it is it goes to an accepting state and waits for either a text command or the arduino to reboot. If the correct bluetooth device isn’t detected then the device will check to see if the vehicle is moving faster than 10 kilometres per hour. If the vehicle is detected moving at above 10 kilometers an hour twice in a row the AVAD enters a stolen state where it will send the location of the vehicle to the owner every sixty seconds unless the correct bluetooth device connects or the device receives a text command to silence or reset the device. The location text message is in the form of a google maps link so it can be clicked and google maps will display the location of the car.
Text commands are listened for at all stages of the program. Sending a text message “find” to the AVAD will result in the AVAD texting the owner the location of the vehicle in the form of google maps link. This link will open google maps (if installed) when pressed. The vehicle will be shown as a red marker on the map.
Sending the text message “silence” to the AVAD will result in the AVAD not sending text messages unless a find command is sent to it. Opposingly, sending the text message “unsilence” to the AVAD will result in the AVAD re-enabling text messages sent from the AVAD.
Sending the text message “reset” to the AVAD will restore the AVAD to its factory defaults. This will wipe stored user data. Sending the text message “help” to the AVAD will result in the AVAD sending a text to the owner of the car with the pincode.

## 3.3 Physical

Physical design was a necessary part of my design stage as the goal of the AVAD is to remain hidden so as not to be disabled in the case of a stolen car. As I was unable to utilise the OBDII port to obtain a power source, the next best thing was to hard wire in an extra cigarette lighter into a switched power source. A switched power source is a power source that only provides power when the key is turned in the ignition. The antennae must also be placed direct view of the sky so it is important to take into consideration the placement of the device in relation to the front windscreen. Although in most cases a long wire for the antennae can be snaked through the dash and put into the correct position.

# 4. Implementation

## 4.1 Arduino wiring

### 4.1.1 Arduino

The arduino is powered by a usb cable that is connected to switched power on the car.

#### 4.1.2 Bluetooth module

The bluetooth module has four pins that are used:

- Vin - Wired to the 3.3v of the GPS module
- Ground - Wired to the ground of the arduino
- TX - Wired to arduino RX
- RX - Wired to arduino TX

#### 4.1.3 GPS module

The bluetooth module has four pins that are used:

- Vin - Wired to the 3.3v  pin of the arduino
- Ground - Wired to the ground of the arduino
- TX - Wired to arduino RX
- RX - Wired to arduino TX
- 3.3v - Wired to the vin pin of the bluetooth module

#### 4.1.4 GSM module

The bluetooth module has four pins that are used:

- VCC - Wired to the 5v pin of the arduino
- Key - Wired to the ground of the arduino
- TX - Wired to arduino RX
- RX - Wired to arduino TX
- Vbat - Wired to the positive of the battery
- Ground - Wired to the negative of the battery
- Reset - Wired to pin  4

## 4.2 Code

#### 4.2.1 Environment

The arduino IDE was used to write the code as it enabled me to perform quick changes to the code and push the changes to the arduino. The arduino IDE also has a serial monitor function that allows the user to see what is being output to the serial port which is useful for troubleshooting.

#### 4.2.2 Familiarisation/Basic functionality of modules

In order to familiarise myself with each of the modules commands, I created an arduino sketch that let me send individual commands to the modules. I took one module at a time and slowly made changes to configure the boards or send commands to get back information. After familiarising myself with the modules I was then able to implement code in the main sketch.

#### 4.2.3 Libraries

All of the three modules came with built in libraries with ready made functions that would callable from the main code. I used both the libraries for the GPS module and the GSM module as they were quite good but I did not use the bluetooth modules library because it was

not useful and only made the code more complex. Instead I made my own functions by writing to the bluetooth modules serial port.

### 4.2.4 Main loop

The main() function loops continuously, therefore this is where all of my code would be called from. I defined some global variables, for example btConnected, which would alter the flow of the main() function. These global variables decided what mode the system would be in. I implemented the main() function first and then built in more functions to be called from the main() function incrementally.

### 4.2.5 Module functions

When expanding on the functionality of my program. I followed the below diagram. I created the code outside the the file by itself. I then tested it to ensure it functioned correctly in conjunction with the external hardware modules. After this I integrated the function into the program code. I then tested the main program to ensure the code integrated correctly.

Implementation diagram

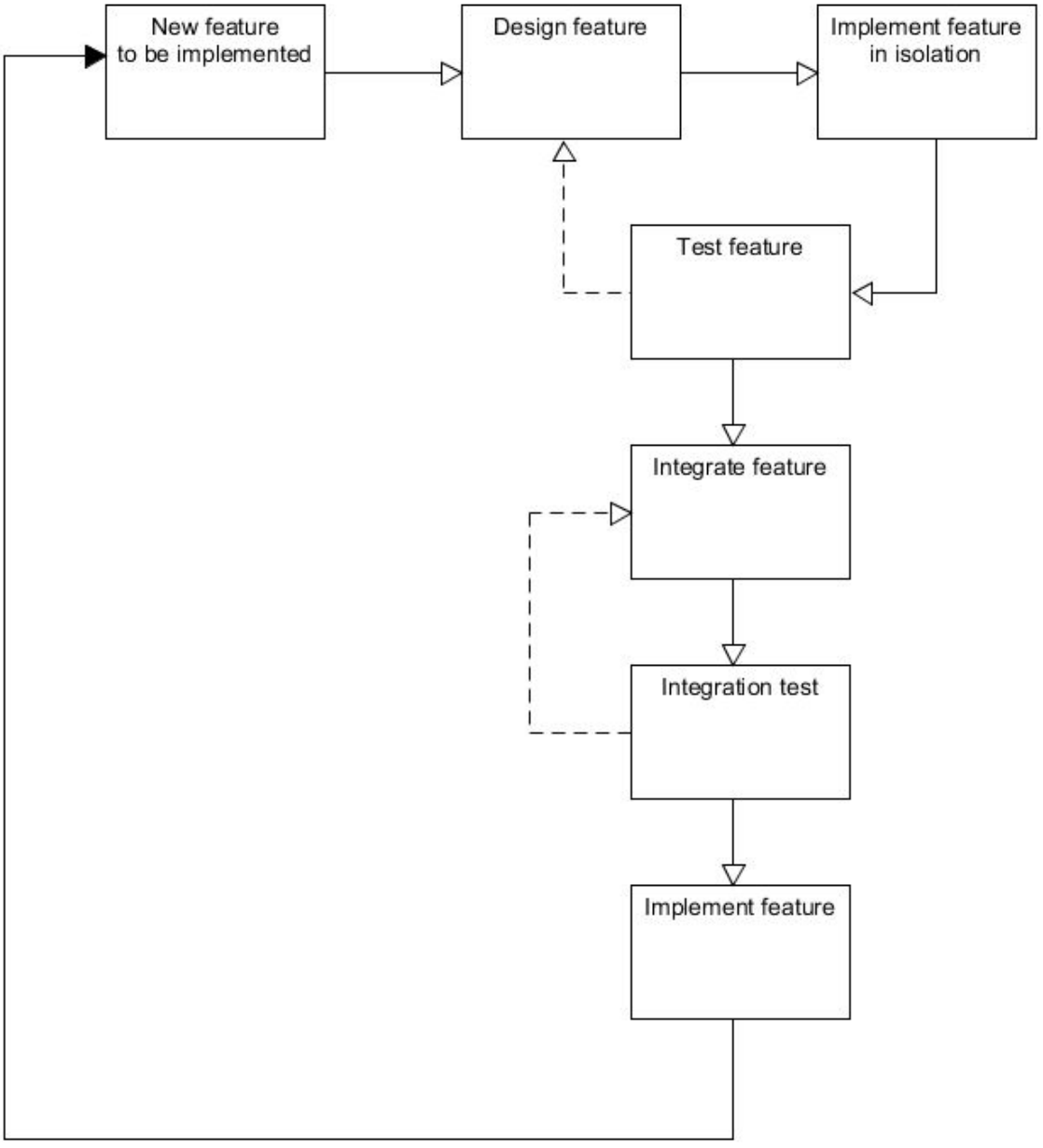
New feature
to be implemented
Design feature
Implement feature
in isolation
Test feature
Integrate feature
Integration test
Implement feature

Fig8

## 4.3 Testing

### 4.3.1 Ad-hoc testing

I performed a lot of ad-hoc testing throughout the development of my code. I did this by pushing the code to the arduino and checking the output in the serial monitor. Print statements allowed me to see the progression of the code as it executed. I was able to fix many issues very quickly by compiling the code and examining the output.

### 4.3.2 Happy path testing

- Setup
- Connect to device using pin
- Waiting
- Find
- Silence
- Unsilence
- Reset
- Help
- Stolen car scenario
- Connected device in car
- Car is under 10 kmph
- Car is over 10kmph
    - Device connected
    - Device not connected

| Test Scenario | Test Case | Pre conditions | Test steps | Test Data | Expected results | Actual Results | Pass/ Fail |
|---|---|---|---|---|---|---|---|
| Valid “setup” text | Check setup is succesful upon sending setup text. | Arduino device has been powered on. | - Send setup text with 4 digit pin to device. | Text: “setup 1234” | Pin is set and device is changed to waiting mode. | Pin is set as 1234 and device is changed to waiting mode. | pass |
| Connect Bluetooth Device | Connect mobile device using setup pin. | Setup text has been sent with test pin. | - Connect phone though blutooth settings. | Pin: 1234 | Phone is connected to arduino device through bluetooth | Phone connected successfully with pin 1234 | pass |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | | | | . | | |
| Valid “find” text | Check if find text returns location of car | Phone is connected to arduino device | - Send find text to device | Text: “find” | GPS coordinates of device are returned by text to phone. | The correct GPS coordinates of the car were received by text on the phone | pass |
| Valid “silence” text | When car is being reported as stolen check if silence stops the arduino device from sending gps coordinates | Device is in stolen mode. | - Send silence text to device | Text: “silence” | Phone no longer receives texts about stolen car | Phone stopped receiving texts about stolen car | pass |
| Valid “unsilence” text | When car is being reported as stolen check if silence stops the arduino device from sending gps coordinates | Device is in stolen mode. | - Send unsilence text to device | Text: “unsilence | Phone receives texts about stolen car | Phone receives texts from stolen car | pass |
| Valid “Reset” text | If reset text received | Device is on | - Reset device | Text: “reset” | Device resets | Device resets | pass |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | reset device | | | | | | |
| Valid “help” text | If help text received send pincode to owner | Device is on | - Send pincode to owner | Text: “help” | Sends pincode to owner through text | Sends pincode to owner through text | pass |
| Car goes over 10kmph and phone is connected | Check if texts are not sent to phone to notify that car has been stolen | Device is in waiting mode and phone is connected through bluetooth | - Drive car over 10 kmph | | Phone does not receive any texts alerts about the car being stolen | Device remains in waiting mode and no texts were received on the phone | pass |
| Car goes over 10kmph and phone is not connected | Check if texts are sent to phone to notify that car has been stolen | Device is in waiting mode and there is no phone connected through bluetooth | - Drive car over 10 kmph | | Phone receives regular texts with gps coordinates of the car | Device changes to stolen mode and regular texts are received with valid gps coordinates of the car | pass |

#### 4.3.3 System testing

After completing the program I manually tested the system as the arduino testing libraries did not suit my needs. This testing method involved me sending many different text messages to the device and doing everything I could to try and break the system.

Bugs discovered and fixed through system testing:

1. GPS coordinates sometimes parsed gibberish and as a result gave an incorrect google maps link.

#### 4.3.4 Acceptance testing

After creating a working version of the AVAD I gave installed the device on 3 peoples cars and asked them several questions as follows.

| Participant | Were you able to setup the device? | Did the device function as described? | Did the device meet your needs? | Would you purchase an AVAD if available? |
|---|---|---|---|---|
| 1 | Yes | Yes | No | No |
| 2 | Yes | Yes | Yes | Yes |
| 3 | Yes | Yes | Yes | Yes |

#### 4.3.5 Validation

Having pitched the idea to many of my friends and relatives who owned cars, 18/20 of them said that they would be interested in an AVAD as it would cost them next to nothing in running costs as a data plan would not have to be purchased but instead it would only cost the user money when the car is actually stolen and that would only be the price of a text message. The reason 2/20 of them were not interested was because of the technology involved. They were unsure about using bluetooth and did not like the complexity of the device. As a result of this I would put forward 20-50 year olds to be the target market of the AVAD

# 5. Sample Code

## 5.1 Setup function

The setup function is used to initialise all of the serial ports and setup the LEDs. In the setup function the user, pin and mac address is read into the system from the EEPROM, ready to be used by the main function.

```
56 void setup()
57 {
58   pinMode(greenLED,OUTPUT);
59   pinMode(blueLED,OUTPUT);
60   pinMode(redLED,OUTPUT);
61   digitalWrite(greenLED,HIGH);
62   digitalWrite(blueLED,HIGH);
63   digitalWrite(redLED,HIGH);
64   delay(10);
65   Serial.begin(115200);
66   Serial.println("Serial Ready");
67   GPS.begin(9600);
68   Serial.println("GPS Ready");
69   Serial3.begin(115200);
70   Serial.println("Bluetooth Ready");
71   Serial.println("Start");
72   fonaSerial->begin(4800);
73   if (! fona.begin(*fonaSerial)) {
74     Serial.println("Couldn't find FONA");
75     while(1);
76   }
77
```

Fig9

```
78   Serial.println("FONA is OK");
79   Serial.println("FONA Ready");
80   Serial.println("All systems are go!");
81   GPS.sendCommand("$PGCMD,33,0*6D");
82   GPS.sendCommand(PMTK_SET_NMEA_UPDATE_1HZ);    //configuring gps NMEA sentences
83   GPS.sendCommand(PMTK_SET_NMEA_OUTPUT_RMCONLY);    //Only receive RMC NMEA messages
84   char md = EEPROM.read(255);
85   if (md == 'f')//create user, pin and mac address from eeprom
86   {
87     if (digitalRead(redLED == HIGH))//turning off red led to show its no longer in setup mode.
88     {
89       digitalWrite(redLED,LOW);
90     }
91     setupMode = false;
92     Serial.println("Setup mode is false");
93     readUser();
94     delay(100);
95     readPin();
96     delay(100);
97     readMACaddr();
98     delay(100);
99   }
100  Serial.println("End of setup!");
101  delay(5000);
102 }
```

Fig10

## 5.2 Main loop

The main function interacts with all the modules and continuously loops on itself changing the global variables as it iterates through the code.

```
104 void loop()
105 {
106   if (setupMode == true) //Enter setup mode
107   {
108     startSetup();
109     Serial.println("Setup mode");
110     delay(1000);
111   }
112
113   else if(setupMode == false && btConnected == false)//Enter working mode
114   {
115     String strGPS = readGPS();
116     strcpy(currentGPS,strGPS.c_str());
117     Serial.println(currentGPS);
118     Serial.println("MIddle");
119     checkForSMS();      // check for a silence text
120     if (carStolen == true)
121     {
122       if (silentMode == false)
123       {
124         sendLocation(user, currentGPS);//send sms
125         Serial.println(user);
126         Serial.println(currentGPS);
127         Serial.println("Sending SMS");
128         delay(60000);
129       }
130     }
```

Fig11

```
131
132     if (btConnected == false)
133     {
134       connectBT();           //if the correct bluetooth device connects enter into third mode which only receives texts for location
135     }
136
137     float spd = getSpeed();
138     Serial.println(spd);
139     if (spd > 10.00 && btConnected == false)
140     {
141       if (speedCounter < 1)   // to ensure it doesnt trigger before 10kph check that it is over 10kph 2 times in a row
142       {
143         speedCounter++;
144       }
145       else
146       {
147         carStolen = true;
148         Serial.println("CAR IS STOLEN");
149       }
150     }
151     else if (speedCounter != 0)
152     {
153       speedCounter = 0;
154     }
155     //Serial.println("END of middle");
156
157   }
```

Fig12

```
158   else if(setupMode == false && btConnected == true)
159   {
160     if (digitalRead(blueLED == HIGH))
161     {
162       digitalWrite(blueLED,LOW);
163     }
164     String strGPS = readGPS();
165     strcpy(currentGPS,strGPS.c_str());
166     Serial.println(currentGPS);
167     if (disconnected==false)
168     {
169       disconnectBT();
170       disconnected=true;
171     }
172     Serial.println("END");
173     checkForSMS();//wait for texts
174     delay(5000);
175   }
176 }
```

Fig13

# 6. Problems Solved

## 6.1 GSM module

Initially I bought a GSM module and the suggested battery from the supplier. I tried to operate this module for a month. The module would sporadically work and would often not turn on. I could not diagnose why the module wasn't working as I had never dealt with anything similar to it before.
After a month of struggling with it trying to diagnose why it was not working consistently, the module died and would not turn on at all. The only possible theory I had was that the battery was either the wrong voltage or the wrong polarity.
I decided to get another board but when doing this I double checked with the suppliers technical support that the battery was indeed suitable. They informed me that they did not know but they pointed me towards a different battery this time.
I purchased the battery and the new board and set them up. The board instantly booted up and performed very reliably.

## 6.2 Bluetooth module

The bluetooth module I purchased specified in the technical data sheet provided with it that the device could work at either three volts or five volts. I powered the bluetooth module through the 3.3v port on the arduino. When I went to configure the bluetooth module I plugged it into my computer with a usb to serial cable. I was able to configure the module but when I rebooted it it would not boot back up.
I came to the conclusion that I must have broken it by mishandling it. I purchased another and tried to configure it the same way through the usb to serial cable. After ten minutes the same thing happened and the module would not boot back up. I made sure that I had not mishandled it so I knew that that was not the reason.
I decided to reconsult the datasheet. After some more reading of the datasheet, it turns out that in order to use the module with a 5v power source, a resistor must be attached. The datasheet specified this on the 40th page. It is probably a commonly known thing but as I am so inexperienced with arduinos and hardware, it was something that I did not know and broke two bluetooth modules as a result.
The major downside to this other than the cost, is that it cost me a lot of time in troubleshooting that could have been better spent.

## 6.3 OBDII module

Originally the OBDII module was going to be used to extract data from the cars computer. The main piece of data that was going to be extracted was speed. The speed was going to act as a trigger for the AVAD to know when the car was moving.
When I began to try and setup the OBDII module and interact with the vehicle, I could not extract any information off of the car's computer. All of the commands that I had researched would not work. I decided that instead of using the OBDII port on the car to get the cars current speed I decided to instead use the GPS module to acquire the speed as it was functioning correctly.

## 6.4 Bluetooth profiles

One thing I had not considered when starting this project was that there was more than one type of bluetooth. The module that I bought was capable of utilising two types of bluetooth profiles, SPP and DUN. This caused me a lot of confusion because these profiles don't allow a simple method of connecting to an android phone. At first I was able to pair with the module and I was able to change all of the settings of the module. I soon realised that the bluetooth was not able to connect to an android device unless it had an SPP server running to allow a bluetooth connection from an SPP device. To get around this I found an app on the app store that allows you to change the bluetooth settings on an android to allow this to happen.

## 6.5 Signal

Often it was difficult test the hardware of the AVAD as my computer would be inside and would cause the GPS module to struggle to acquire a fix. The GSM module would also sometimes suffer from no reception.This often made it difficult to work with the hardware as the GPS antenna needs a full view of the sky. There were times when the antenna had a full view of the sky but still couldn't acquire a fix as I was in a built up area.
This made it difficult to diagnose problems with the boards and the software. I was unable to figure out if the GSM board I had was faulty or if it was simply because it wasn't receiving a strong enough signal. Eventually when the board stopped working altogether and I tested the new board I found out that the signal was never the issue.
While the GPS board was working it was difficult to get it to a position where it would get a fix on a satellite. As a result it took me a long time to figure out how to properly parse the Data that was being sent from the GPS module and perform the necessary computations to transform the data into readable GPS coordinates.

## 6.6 Delays

When coding the arduino and performing some ad-hoc testing I realised that many of the commands that I was sending were not working. I quickly realised that in order to send information form the arduino to a module  and receive some information back there must often be a delay command issued to ensure the module has enough time to reply. A delay command simply tells the arduino to pause for however many milliseconds you specify in the comand.
For example, when scanning for a bluetooth device the arduino sends the command "i,7" which tells the bluetooth module to scan for bluetooth devices for 7 seconds.
Without "delay(9000);" the arduino will not be able to record the returned devices as it will instantly try to read the bluetooth modules return. In this example it is pretty obvious that a delay must be implemented. However in some other cases it was not so obvious as the returned data for not be formatted correctly unless a small delay of ten milliseconds or so was send.

## 6.7 Character arrays

A character array was something that I had never encountered before. From my research of arduinos, I had seen in many places online that it is not advised to use strings as they use a lot of memory. The solution to this problem was to use character arrays instead. While I was able to execute simple code with character arrays, I ran into problems when it came to returning

them in a function. I decided to solve this problem that instead of making my code very complex I would create global character arrays to avoid having to return character arrays and instead edit the global variables from inside the functions.

## 6.8 Bluetooth library

With the GPS module and the GSM module, both provided a library that could be used with the arduino IDE. While this was also true for the bluetooth module, the library was not very comprehensive and relied instead on the user consulting the module and sending a 'h' character in order to bring up a list of commands. I decided instead to not use the bluetooth modules library and instead issue the commands myself through the serial.
This meant that all of the commands that I would be issuing to the bluetooth module would have to be created by myself in a function and I would have to test each one to ensure that it was returning the correct information and within the correct time frame. This took a while to get used to as the commands were sometimes temperamental and took a lot of fine tuning as the delay times for each were different. The most complex part of the bluetooth module was trying to search for a bluetooth device and the mac address for that device.

## 6.9 Serial transfer

I had a lot of problems initially communicating between the arduino and the modules through the hardware serial. The Serial uses two cables,  a TX and an RX for transmit and receive. At first I wired it up correctly but did not understand BAUD rates and the difference between each serial.
If the arduino is communicating with a module that runs at 115200 BAUD rate and the arduino thinks it is running at 9600, then the characters will not be received or transmitted properly and will more than likely end up looking like gibberish. After some configuration I managed to configure each module to run at the same speed that the arduino was expecting it to run at. This enabled the modules and the arduino to communicate with each other and pass characters back and forth.
Another problem I came across was that sometimes when reading information from a serial port, it would be incomplete data. I narrowed this problem down to the fact that there was a limit on the serial buffer size. I managed to fix this problem by increasing the size of the buffer in the code of the arduino hardware.

```
// often work, but occasionally a race co
// Serial behave erratically. See https:/
#define SERIAL_RX_BUFFER_SIZE 256
#if !defined(SERIAL_TX_BUFFER_SIZE)
#if ((RAMEND - RAMSTART) < 1023)
#define SERIAL_TX_BUFFER_SIZE 16
#else
#define SERIAL_TX_BUFFER_SIZE 64
#endif
#endif
#if !defined(SERIAL_RX_BUFFER_SIZE)
#if ((RAMEND - RAMSTART) < 1023)
#define SERIAL_RX_BUFFER_SIZE 16
#else
#define SERIAL_RX_BUFFER_SIZE 64
#endif
#endif
#if (SERIAL_TX_BUFFER_SIZE>256)
typedef uint16_t tx_buffer_index_t;
#else
```

### 6.10 Bluetooth entering command mode

When I was initially configuring the bluetooth module to change the settings, the datasheet stated that in order to enter the configuration mode the string “$$$” must be sent to it and it would return “CMD”. I could not get this to work for a long while and nowhere online stated this as a problem.
After a lot of time searching for a solution I eventually changed a few settings and managed to enter the configuration mode. It turns out that in order to enter configuration mode, “$$$” must be sent with now newline or carriage return character but in order to issue any other commands it was necessary to include either a newline or carriage return character. Once I figure out this problem I was able to configure the bluetooth module.

# 7. Future Work

### 7.1 Bluetooth Profile

I would investigate to see if there were other bluetooth modules on the market that would operate using a different bluetooth profile to SPP. This would allow the user to simply connect to the AVAD and not have to install any application. One profile which I would investigate further is A2DP as  all phones can't connect to most bluetooth speakers without having to install any extra applications in order  to create a connection with them.
This profile may allow the AVAD to simply connect to the device ensure its there and then once the connection has been initiated successfully, it could disconnect as it would be satisfied.

## 7.2 Battery

Currently the AVAD is configured to work as a device that only turns on when the vehicle is turned on. This would prevent the AVAD from draining the vehicles battery while the vehicle is turned off.
If I had more time on this project I would investigate a way of using a separate external battery to power the AVAD all the time. This would allow the user to find the location of the car even when the car is not turned on but also prevent the AVAD from draining the vehicles battery while not in use and preventing it from starting.

## 7.3 Fingerprint scanner

After experiencing all the problems with the bluetooth module, I had an idea that a fingerprint scanner could possibly be substituted in. The fingerprint scanner could not only act as a device to ensure that the user of the car is indeed in the car but as well as that could possibly act as an immobilizer and prevent the car from starting if the user does not provide their finger print. This would eliminate the need for the user to interact with the device through bluetooth.

## 7.4 Further functionality

I would investigate the possibility of adding extra functionality to the AVAD. Such as adding a function that would immobilize the vehicle unless the AVAD allowed the vehicle to move. Another function I would investigate would be the possibility of a camera on the dash that would trigger when the vehicle was detected as stolen and send a picture of the driver to the owners phone.

## 7.5 OBDII module

At the start of this project I had tried to implement an OBDII module. There were two main reasons behind this, it would be easier to acquire the speed of the vehicle and it would also provide a power source for the device so it would not have to be hardwired in.

## 7.6 iPhone integration

As of this moment I have not tested the device on an iPhone as I do not have access to one. If I was to take this project further I would try to make it suitable to use for an iPhone as well as other phones. While the texting system of the AVAD would work with other phones the bluetooth module may not work as I have not investigated the bluetooth profiles that work with other phones other than android.