ADS-B Flight Tracker

By Nicholas Zanon, Krutik Shah

Digital Signal Processing, Final Project Report

Professor Richard Pedersen

Background Information

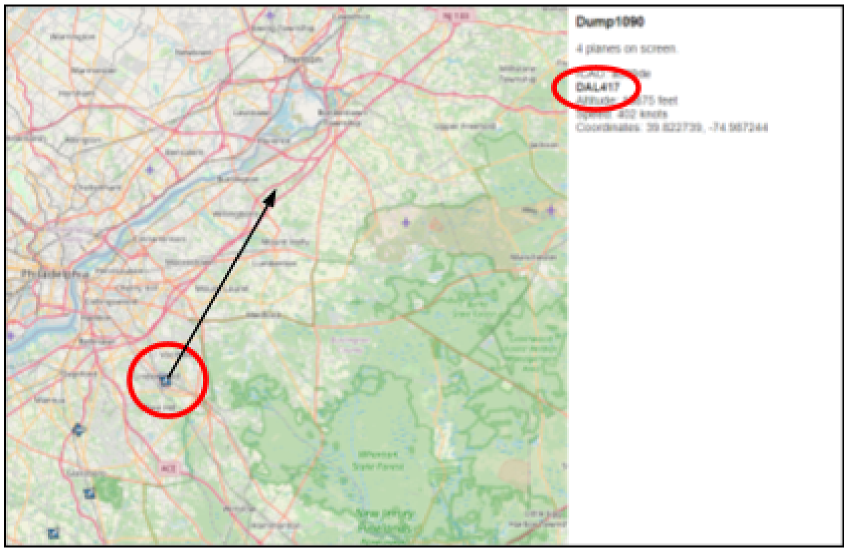
First we have to understand the basics of the Identification Friend or Foe system (or IFF). Essentially, aircrafts use a transponder to listen to interrogation signals from a broadcaster (such as air traffic control towers) and then will respond back with various information about the aircraft. The response is dependent on the type of interrogation mode that is used. Modes 1-5 are for military use, while Modes A, B, C, D, and S are for civilian use. For the purposes of this project we will focus on civilian modes, for obvious reasons. Transponders receive interrogation signals on 1030 MHz and respond back on 1090 MHz. Mode 3A simply provides a 4-digit squawk code (transponder code), and Mode 3C improves on this by providing altitude. Mode S takes it a step further by allowing selective interrogation (hence the S in Mode-S). This means that the broadcaster can interrogate different information from different aircraft separately. Additionally, Mode-S assigns a 24-bit address to each aircraft, meaning there are 2^{24} different possible addresses. Compared to Modes 3A/C, Mode S allows a lot more information to be interrogated. However, there is one issue with these modes, and it is that they require the interrogator to figure out the position based on where the signal is coming from. With ADS-B, this mitigates that issue because ADS-B collects and responds back with that information from the GPS module(s) found within the aircraft. All that is needed from the interrogator is to decode the ADS-B signal.

Our goal was to put together an ADS-B, to monitor air traffic from the surrounding area. We followed a Tutorial from Make Magazine where the user utilized a BeagleBone Black board, an RTL-SDR kit like the one we used in class, and a program called "Dump 1090 is a Mode S decoder specifically designed for RTLSDR devices". The BeagleBone Black runs on Debian Linux, which we can access through SSH from a local network. The project includes a software that automatically tunes the receiver to 1090MHz, the frequency at which commercial air flight data is transmitted. According to mode-s.org: "Parameters such as position, velocity, and identification are transmitted through Mode S Extended Squitter (1090 MHz).". This data will be output to a webpage server running off of the Beaglebone board. There we will see a map and the location of the airplane we are tracking.

We also referenced The 1090 Megahertz Riddle: A Guide to Decoding Mode S and ADS-B Signals. (TU Delft OPEN; 2021. doi:10.34641/mg.11) in order to learn more about the decoding process of the Mode S signals used to track the plans.

Flight Tracker

We followed the Tutorial from Make Magazine and were welcomed with the following webpage in Figure 1: Flight Tracker interactive Page. This page plots the information that your SDR kit received with a little airplane symbol, on a map of the area in which you are receiving it. On the right side of the screen, you will see the number of planes that the software has recognized. When selecting a flight, the menu will show you the hex formatted ICAO, the plane's callsign, altitude, speed, and current coordinates updates every 10ms.



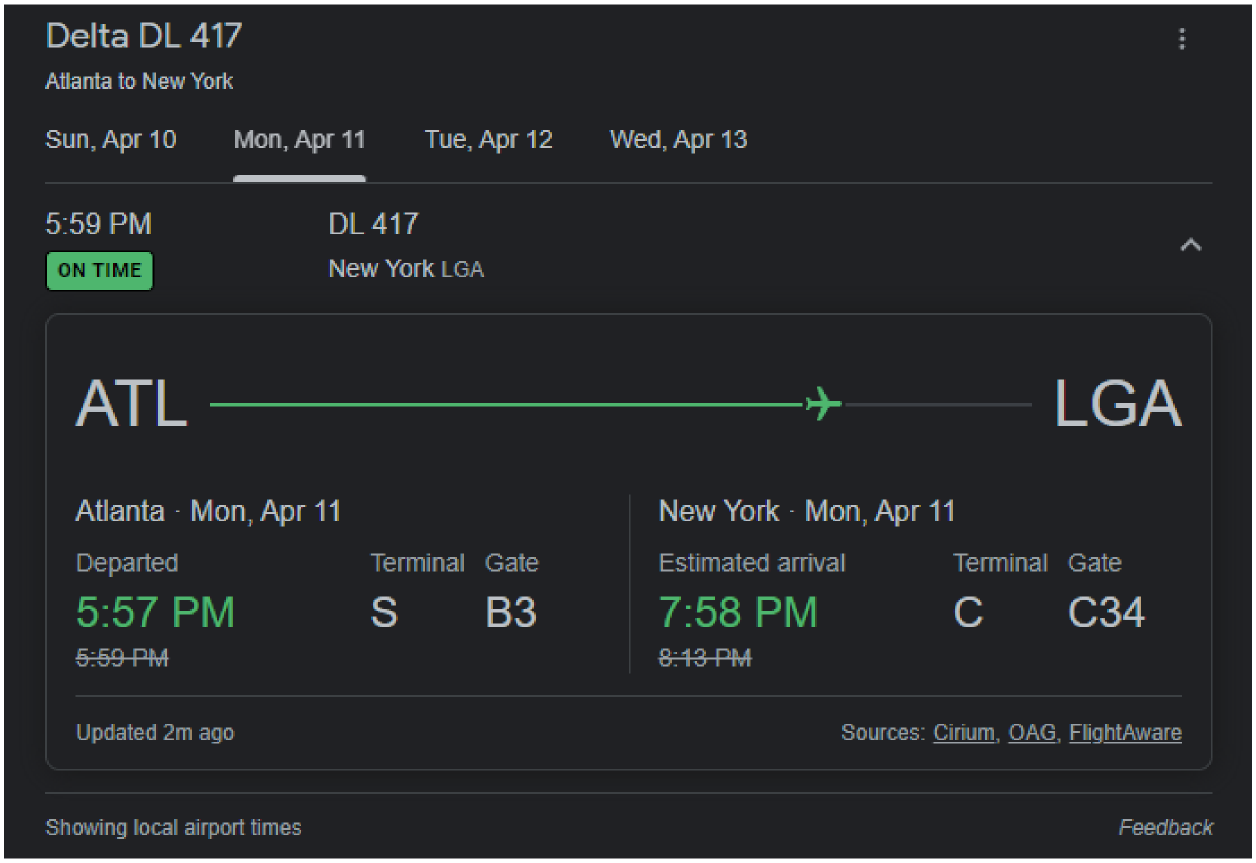
Dump1090

4 planes on screen.

ICAO: [illegible]de
DAL417
Altitude: [illegible]575 feet
Speed: 402 knots
Coordinates: 39.822739, -74.987244

Figure 1: Flight Tracker Interactive Page

Figure 1 shows flight DAL417, a Delta Airlines flight headed towards New York. We used google to confirm this, shown in Figure 2. We found that the software would always use call signs with 3 letters in the beginning, however the google results often showed 2 beginning letters.


Delta DL 417

Atlanta to New York

Sun, Apr 10 | Mon, Apr 11 | Tue, Apr 12 | Wed, Apr 13

5:59 PM ON TIME | DL 417 New York LGA

ATL → LGA

Atlanta · Mon, Apr 11

Departed	Terminal	Gate
5:57 PM ~~5:59 PM~~	S	B3

New York · Mon, Apr 11

Estimated arrival	Terminal	Gate
7:58 PM ~~8:13 PM~~	C	C34

Updated 2m ago

Sources: Cirium, OAG, FlightAware

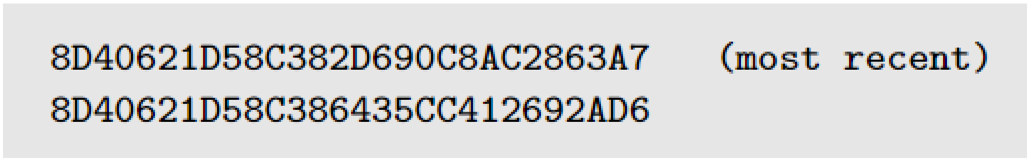
Showing local airport times

Feedback

Figure 2: DAL417 confirmation.

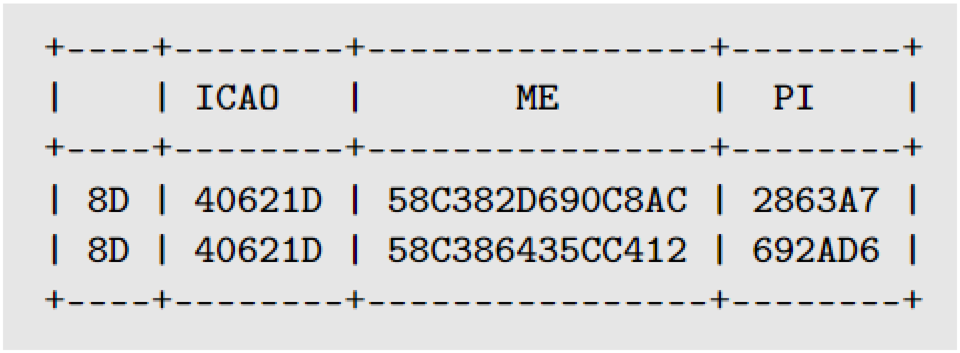
Decoding ADS-B Messages

While reading through the source code of “dump1090”, we realized that we weren’t entirely sure what exactly we were looking for. We turned to the Campbell library to find some literature, and used “The 1090 Megahertz Riddle: A Guide to Decoding Mode S and ADS-B Signals” to learn more about what was being transmitted. We learned that Mode S uses “Pulse Position Mod-ulation (PPM)” to transmit signals, which is a method of transmitting square waves at a given frequency (in our case, 1090MHz). The software takes these signals and converts them to hexadecimal. From there, the data is divided into 4 sections; the preamble, the ICAO address, the ME field (message), and the PI field (parity). Figures 3a and 3b shows an example of the raw hex data broken up into these fields.



```
8D40621D58C382D690C8AC2863A7   (most recent)
8D40621D58C386435CC412692AD6
```

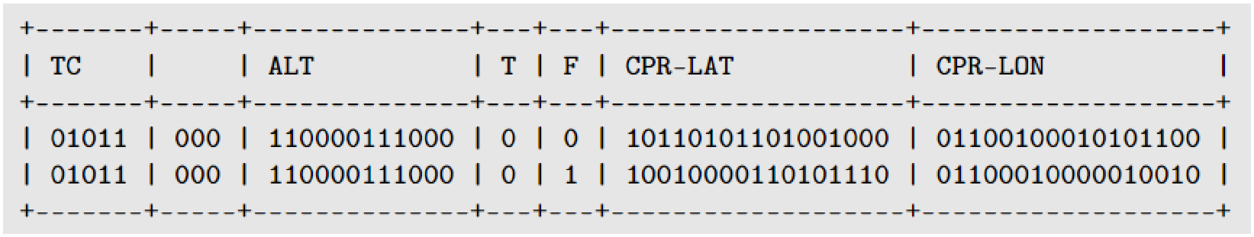
Figure 3a: ADS-B data raw



```
+----+--------+----------------+--------+
|    | ICAO   |       ME       |  PI    |
+----+--------+----------------+--------+
| 8D | 40621D | 58C382D690C8AC | 2863A7 |
| 8D | 40621D | 58C386435CC412 | 692AD6 |
+----+--------+----------------+--------+
```

Figure 3b: ADS-B Sections

These fields are further deconstructed to reveal information about the flight. The ME field contains information including the Altitude, latitde coordinates, and longitude coordinates. The ME field is converted to decimal, broken up into sections, and then each section's decimal value is used in a calculation. Figure 4 shows the ME fields' sections.

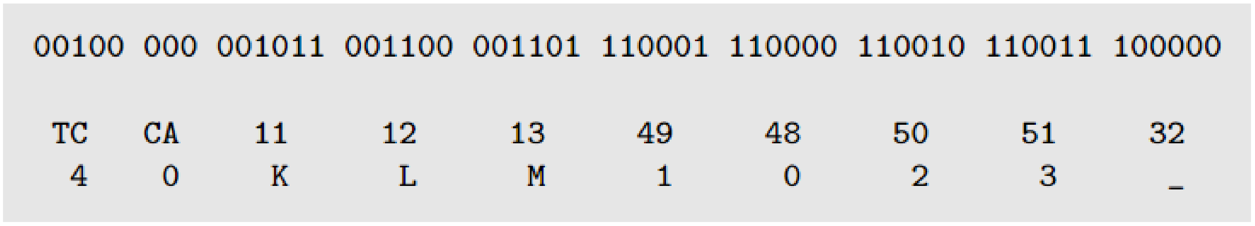


```
+-------+-----+--------------+---+---+-------------------+-------------------+
| TC    |     | ALT          | T | F | CPR-LAT           | CPR-LON           |
+-------+-----+--------------+---+---+-------------------+-------------------+
| 01011 | 000 | 110000111000 | 0 | 0 | 10110101101001000 | 01100100010101100 |
| 01011 | 000 | 110000111000 | 0 | 1 | 10010000110101110 | 01100010000010010 |
+-------+-----+--------------+---+---+-------------------+-------------------+
```

Figure 4: ME Field Sections

The book goes into great detail about the calculations for each piece of information, while following along with an example.

We attempted to follow along and decode the hex data of some flights we tracked with the program, however we were getting stuck at the step of finding the call sign. We believe that the callsign is linked to the ICAO address, however at the time we were trying to find it solely within the TC section of the ME field. The callsign is transmitted using the lower 6 binary bits of ASCII characters. Figure 5 is the book's example of callsign "KLM1023".


```
00100 000 001011 001100 001101 110001 110000 110010 110011 100000

 TC   CA    11     12     13     49     48     50     51     32
  4    0     K      L      M      1      0      2      3      _
```

Figure 5: Callsign decoding example

Problems

We ran out of time before fully decoding a signal that we found, we got stuck at finding the flight's callsign. In the section above, we show the ME field being decoded to show both the plane's location, and the flight's callsign. We were unable to figure out the difference between the two ME fields, and when it shows the callsign and when it provides the flight's location.

Conclusion

While we were unable to do a deep dive into the source code, or fully decode a flight's transmission, we did learn a great deal about ADS-B and Mode S transmissions. This was a great introduction to the practice, and can be used as the groundwork for projects in the future. Using PPM to transmit signals, then decoding those signals within another systemWe were focused on decoding the ME field, as that provides the vital information specifically about the flight itself.

Firstly, to start the decoding process, we need to collect raw data from the BeagleBone Black. To do this, we can start the dump1090 with the following command in the terminal window:

```
./dump1090 --raw
```

The results that will show will be in hexadecimal. For our demonstration for decoding, we received the following raw value on the BeagleBone Black:

```
8dacaf0be11d2e00000000c98c39
```

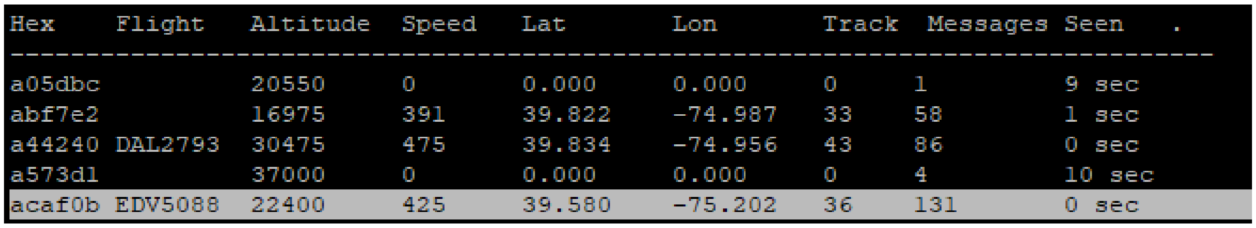
Below, we started decoding the raw value. First, we have to separate the different fields in hex. Afterwards, we have to convert them to binary. For our purposes, the ICAO field and the ME field are most important. The acronyms are defined below.

Acronyms:
DF = Downlink Format
CA = Transponder Capability

ICAO = ICAO Aircraft Address (can be looked up, or matched with dump1090 interactive mode)
ME = Message
PO = Parity/Interrogator

	DF	CA	ICAO	ME	PI
Hex	8D		ACAF0B	E11D2E00000000	C98C39
Binary	10001	101	101011001 0101111000 01011	1110000100011101001011100000 0000000000000000000000000 000	11001001100011 0000111001

The ICAO address in hexadecimal can be looked up on Google to find flight data, or can be found in dump1090's streamlined flight view. The flight is found below with the corresponding ICAO address in the leftmost field in the column labeled "Hex":



```
Hex    Flight   Altitude  Speed   Lat       Lon       Track  Messages Seen   .
--------------------------------------------------------------------------------
a05dbc          20550     0       0.000     0.000     0      1        9 sec
abf7e2          16975     391     39.822    -74.987   33     58       1 sec
a44240 DAL2793  30475     475     39.834    -74.956   43     86       0 sec
a573d1          37000     0       0.000     0.000     0      4        10 sec
acaf0b EDV5088  22400     425     39.580    -75.202   36     131      0 sec
```

The RTL-SDR collected 131 different messages from this aircraft from the point we started dump1090 to when we took this screenshot. We are decoding only one of those messages. Our goal is to further decode our signal beyond just the ICAO address. We need to focus on the ME field, which is what contains our message. Here, we need to further split up the binary code and the way it is split up is determined by the TC. The ME field is 56 bits, and the first five bits are considered the "Type Code" or TC. This determines what type of message we are decoding. In our case, the TC is 11100, which is 28 in decimal. Upon further research in the 1090MHz Riddle article cited in the background, we discovered that a TC of 28 means the message indicates aircraft status. We unfortunately could not figure out how to decode the rest of the ME field as there were no resources available describing an aircraft status message TC.