Chapter 1

GSoC Proposal 2017 - AerospaceResearch.net

Project Title : Signal detection and decoding for ADSB
Name: Valliappan
Degree: B.E(Hons) Electronics and Instrumentation
University : Birla Institute of Technology and Science-Pilani, India
Location/Timezone: India, GMT +5:30
Mentor: Andrews Hornig & Muzy

1.1 Introduction

On 1090 MHz airplanes are transmitting their meta data like aircraft id, location, speed and direction.It is called adsb. I have to create A Module out of the python libdump1090. The module needs to be loadable by the ground station main software. The output shall be each adsb signals by their data and the starting time (sample count) within the input file (an iq-stream).

1.1.1 Main features of libdump1090

1.Robustness in decoding the weak messages.
2.Single bit error correction.
3.Pre-calculated checksum table for D11 and D17.
4.This libdump1090 can decode DF0, DF4, DF5, DF16, DF20 and DF21 where the checksum is xored with the ICAO address by brute forcing the checksum field using recently seen ICAO addresses.

1.1.2 Limitations

1)The libdump1090 works fine only if the data is less than or equal to 1024Kb.
2)The main code is written in C and python wrapper to exicute in python. Instead of uniform python module. Hence making it two language dependent.
3)Mode_s.c code uses huge amount of precomputed values. For example the magnitude lookup table which has a size of 65536.
4)The output of the ADSB decoded message is not compact, as it displays the unwanted parameters corresponding to the Dataframe number.

1.2 Project Goals

I would also like to incorporated some changes to the main code:-

- Primary objective of the project is to make the code work for input file size greater than 1Mb.

- Return a new aircraft structure for the interactive mode.
- CPR coordinates decoding and track calculation from velocity using manual[5]
- Display of the compact decoded message consisting of only the essential parameter corresponding to the Dataframe. The output shall be time, ADSB signal and the decoded message.

Intergrating changes to ground station main software. Hence making a working model ready for Distributed Ground Station Network.

Simultaneously, I would like to implement the Mode_s decoding algorithm from scratch as a python module. I would handle this by implenting each block and testing their work and proceeding to the next block as shown in **Figure 1.1** . There is already an academic implementation available in C [1], but it is not well-supported with the python wrapper.

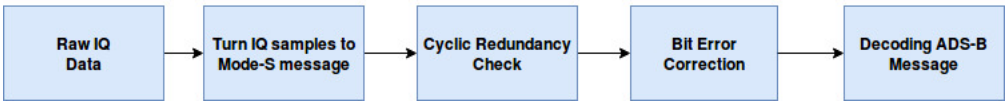

Figure 1.1: Flowchart based on the understanding of libdump1090 code

1.3 Implementation

I will be discussing the implemention procedure for each of my project goals discussed above:-
1)Making the code working good for any input file size. Here are variables, that has created possible problem in the earlier code.

```
MODES_ASYNC_BUF_SIZE = 4*16*16384 \\ 1Mb  defined in Mode_s.c
ADSB_BUF_SIZE = 4*16*16384 # 1MB defined in Mode_s.py
```

Initially, I used the pincipal of divide and conquer, where I will divide the big file into many small 1Mb files. If the file is less than 1Mb then remaining values will be appended with 1's. Link to the code. This idea was adopted from [1] where

```
MODES_DATA_LEN = 16*16384. \\ 256Kb
\\ Each time the buffer is filled with 256Kb of data and used for decoding
\\ It goes on reading till end of the file.
\\ If the buffer doesn't have sufficient data, then the rest of the data will be
\\ filled with 1's.
```

The output of **Mode_s.py** program should display the decoded message as compact as follows:

```
Message : 8d3c6dd358b980a9ee026ce0c7eb;
CRC: 000000 (ok)
DF 17: ADS-B message.
   Capability     : 5 (Level 2+3+4 (DF0,4,5,11,20,21,24,code7 - is airborne))
   ICAO Address   : 3c6dd3
   Extended Squitter  Type: 11
   Extended Squitter  Sub : 0
   Extended Squitter  Name: Airborne Position (Baro Altitude)
   F flag   : even
   T flag   : non-UTC
   Altitude : 36000 feet
   Latitude : 21751 (not decoded)
   Longitude: 620 (not decoded)
```

This can be done by adding if-else statements in `displayModesMessage()` function.

The final part of the project would the implementation of python module from the scratch. I would like to make an approach to this implementation systematically as shown in the block diagram **Figure 1.2**.

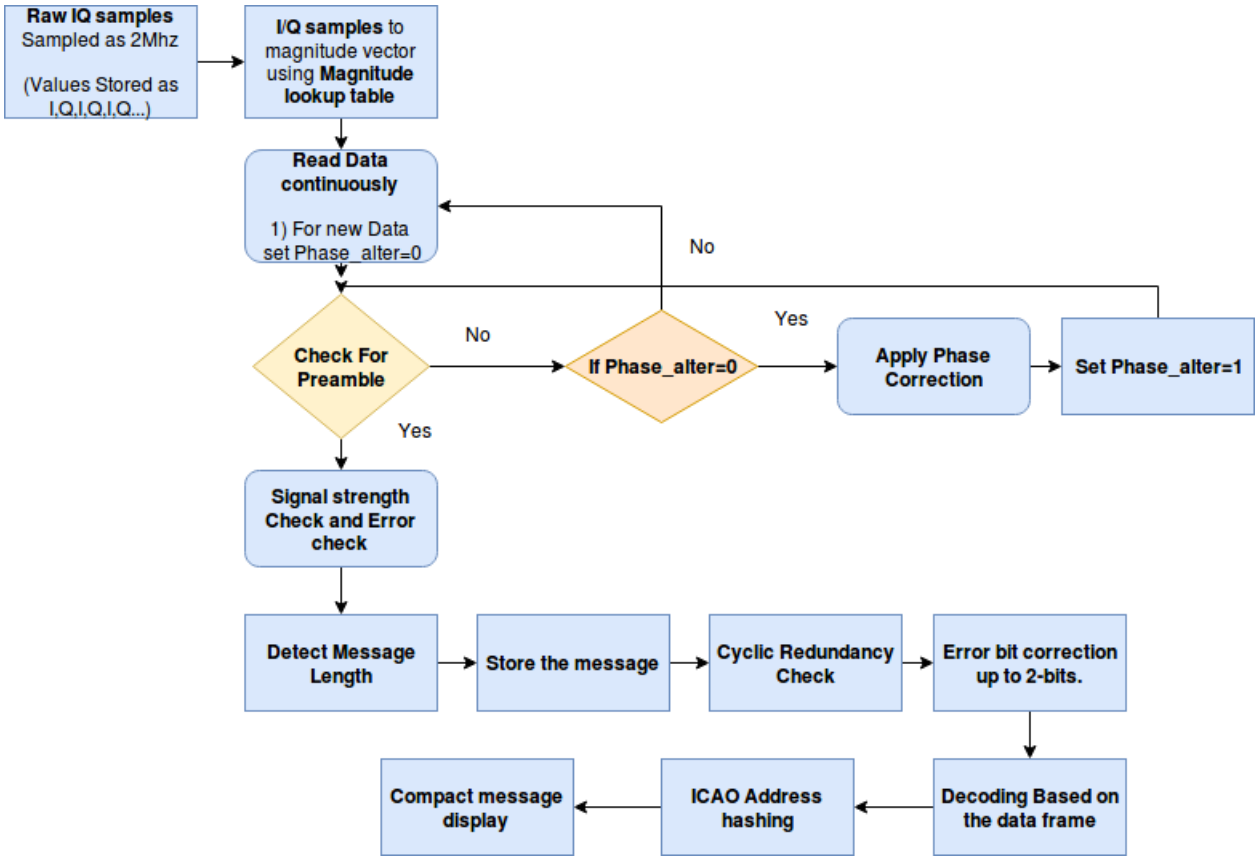

Figure 1.2: Implementation work Flow

For the implementation I will write many helper functions and their purpose are self explanatory from the function names. The function list is rough sketch based on the requirements. The structures defined in the mode_s.c will be converted implemented as class in the new python implementation which improves the security of the information.
The Function list for **RawIQ to message**:

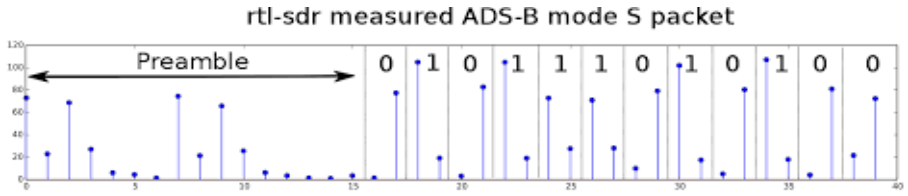

Figure 1.3: Preamble & Payload

```
computeMagnitudeVector(IQ_Data) # this function uses the magnitude lookup table

decodemodeS(magnitude_vector)
Iterate through the magnitude vector check for a valid Preamble of 16 bits as
shown in figure (1.3).
   |--> first check for consecutive 10 samples representing a valid preamble.
```

```
   |--> The samples between the two spikes must be < than the average
        of the high spikes level.
   |--> Similarly samples in the range 11-14 must be low.

If the above attempt with this message failed retry with Phase_correction()

If the proper preamble is found then we will decode the message length based on
conditions implemented by muzy
   |--> Here we would like to cross check if the message length found from
        dataframe is same as the message length detected.
   |--> Finaly before decoding message we check for Signal strength > 0x02FF and
        Errors <= 3.
   |--> Decode the message using decodeModesMessage(message).
   |--> update the iterate pointer to 'preamble_size + message_len'

If you cannot find the message retry with phase correction.

Phase correction Function:

detectOutOfPhase(Preamble)
# Return -1 if the message is out of phase left-side
# Return  1 if the message is out of phase right-size
# Return  0 if the message is not particularly out of phase.

clamped_scale(value, scale)
#returns a scaled valued <= 65535 , this function is used in PhaseCorrection()

PhaseCorrection(magnitude_vector[preamble+paylode])
#Algorithm is same as libdump1090 code written by muzy.
```

Error Correction will be the next major task which is implemented similar to libdump1090 code:

```
Pre-Defined modes_checksum_table of size 112 each 32bit hexadecimal value

modesChecksum(message, msg_len)  #Returns 24 bit checksum syndrome.

modesInitErrorInfo()
# Compute the table of all syndromes for 1-bit and 2-bit
# error vectors making it less intensive for the computer


fixBitErrors(message, bits_to_fix)
# Search for syndrome in table and if an entry is found, flip the necessary
# bits. Make sure the indices fit into the array
```

Lastly decoding the message:

```
decodeModesMessage(message)
Decode message type DF0, DF4, DF5, D11, DF16, D17, DF20 and DF21.
With special emphasis on DF20 and DF21
Mode S Enhanced Surveillance (EHS)
Downlink Format and message structure
    Parity and ICAO address recovery
    Address Parity
```

```
    BDS (Comm-B Data Selector), BDS 2,0 (Aircraft identification), BDS 4,0
    (Selected aircraft intention)
    BDS 4,4 (Meteorological routine air report), BDS 5,0 (Track and turn report),
    BDS 6,0 (Heading and speed report)
    as illustrated in [2]
Lastly CPR coordinates decoding and track calculation from velocity based on[2 & 5].
interactiveShowData()
   Interactive Display of aircraft information with CPR latitude and longitude .

ICAOCacheHashAddress(address)
# if crc == 0 try to populate our ICAO addresses whitelist.

addRecentlySeenICAOAddr(address)
# if crc == 0 try to populate our ICAO addresses whitelist.

ICAOAddressWasRecentlySeen(address)
# Compare the checksum with the whitelist of recently seen ICAO
# addresses. If it matches one, then declare the message as valid
```

1.4 Timeline

Community Bonding Period

- A huge part of being part of the open-source community and Google Summer of Code is sharing your ideas and work. I've started a blog here to give weekly reports of the summer work I will be doing, and try and break down the world of probabilistic topic models to everyone.
- Before the official time period begins I intend to fix the issue of libdump1090's failure for input file size $>$ 1Mb. Along with the displayModesMessage() for displaying compact Mode_s message.

Week 1:

- Start with the Python implementation of decoding Raw IQ data to magnitude vector.
- Testing is done use the pre-existing **output.bin** and **IQ_data.dat** given by Andreas during the discussion period.
- Last 2 days of the this week is alloted for backup and documentation.

Week 2 and 3:

- Decoding Modes from the magnitude vector by checking for preamble
- Phase correction, Signal Strength check & Error check < 3
- Ultimately detect the message length .
- Testing and documentation

Week 4 and 5:

- Cyclic Redundancy check and Check sum implementation
- Bit error correction
- Implementing error correction important step. Hence to be handled with intensive care.
- Testing and documentation and update the blog.

Week 6 to 7:

- Decoding the Mode_s message based on the manual [2,3 & 5]
- Keep working on the blog as well!

Final Phase :

- Integrating the changes to DGSN
- Making requirement changes that the above implementation can also be used for data recieved from Software Defined Radio (RTL-SDR).
- Signal ADSB Visualisation in the interactive manner. based on interactiveShowData()[1,4 & 5].
- Keep working on the blog as well!

1.5 Furture work

Apply other kind of Correction for magnitude vector as discussed in paper [6,7], to investigate improved squitter reception techniques. For more capable error correction algorithms, and more selective preamble detection.

1.6 Commitments

My next summer vacation starts by May 10^th and next semester course work begins by August 2^{nd} week. Hence I will not be occupied with any other work during the GSoC Program and will dedicate my full concentration on project.
I would take Sundays off but will maintain a minimum of 40Hrs per week.If required by mentors, I will make required changes to my proposal. Also, I have not applied for any other organisation.

1.7 Reference

[1]Dump1090 written by Salvatore Sanfilippo. [2]ADSB-decoding guide [3]Mode_s Implementation manual by Lincoln Laboratory [4]MalcoRobbs Dump1090 implementation. [5]http://www.lll.lu/ edward/edward/adsb/DecodingADSBposition.html
[6]Barhydt, Richard, et al. "ADS-B within a multi-aircraft simulation for distributed air-ground traffic management." Digital Avionics Systems Conference, 2004. DASC 04. The 23rd. Vol. 1. IEEE, 2004.
7]Harman, W., J. Gertz, and A. Kaminsky. "Techniques for improved reception of 1090 MHz ADS-B signals." Digital Avionics Systems Conference, 1998. Proceedings., 17th DASC. The AIAA/IEEE/SAE. Vol. 2. IEEE, 1998.

1.8 Contacts

Facebook :Valliappan CA
Twitter :@valliappanca
Gmail & Gtalk :spmanikam@gmail.com
Skype :valliappan.ca
Github Handle : PRRvalli
Blog related to my GSoC Project
Phone & Whatsapp : +91-8378988093
Github Repository :Project that I coded
Google drive : Link

Chapter 2

Motivation Letter

Being an undergraduate student for me studies is all about exploring new fields of science. Through out my undergraduate studies I have worked with many signals, starting from bio-medical signals like brain signals(EEG), Speech signals, ECG etc. I have also worked on artificial signals like Broadband reflectometry signal, this gave me the first experience working with IQ data. Every signal has information embedded in them as engineers my duty is to study them to decode the information. So far I have been success full in doing so with the above mention signal. My github projects and CV is evidence for my knowledge in this field.

My academic courses on Satellite communication, Mobile telecomunication network & Telecom-Switching Network has enriched me with the required knowledge for understanding the idea **"Signal Detection and Decoding of ADSB"**. I'm an active code in hackerrank and SPoJ as I'm interest towards Data Structures and Algorithm. My prefered languages are C/C++, Python and Matlab. I prefer Matlab for visualisation of data graphically. Hence I think I am very well suited to work on the proposed project imlementation as it required good comfort level in Python and C++ along with algorithm to implement efficiently.

Every Single project ideas proposed in AerospaceResearch.net organisation are interesting and enthusiatic especially because of the wonderful visuals especially "**Lone Pseudoranger**" project for data visualisation. This bring celestial objects close to us, that excites me. Being an Electronics and Instrumentation student, and my deep interest with signal made me choose the project idea **"Signal Detection and Decoding of ADSB"**. Added to this AerospaceResearch.net organisation has wonderful mentor Mr.Andreas who has be patiently answering my doubts from the start. I have learned a lot of essential information from him and I'm highly motivated to work with him during the summer 2017 by contributing for GSoC project.