EECS 159B Senior Project Design

Flight noise tracking with ADS-B

by ADS-B

2017/6/15

Jianbo Liu, Electrical Engineering

Michelle Lai, Electrical Engineering

Yiqing Zhang, Electrical Engineering

G.P.Li, Michael J. Klopfer

Executive Summary

Aircraft noise is a serious problem for communities who live living around airports and over flight paths. Methods for aircraft noise abatement include the specification of flight tracks and landing/take-off patterns, curfew or time constraint, with compliance focused primary on manual spot sound auditing. The purpose of our project is to build a distributed digital system that is able to detect the continuous logging of aircraft noise with correlation between time and aircraft. This project involves the construction of a single node of a distributed monitoring and compliance system. This device is able to decode the aircraft beacons, do analyzing and extract significant flight information, correlate the information extracted with frequency separated ambient noise levels. SQL server will do the further processing to the data sent over. This whole device is available on an ADS-B decoder circuit, an ARM Cortex-M3 MCU, and a VLSI VS1053b DSP audio processor. ESP8266 Wi-Fi link and MQTT telemetry protocol provide the database connectivity and data transfer.

Noise is a big problem in today's society, which seriously influence people's life. This project focuses on the noise of flight. We analyze the amplitude of aircraft noise then save to a server for further analyse. For places where people frequently report suffering aircraft noises, a noise cancellation device with the noise tracking device we built will be used, when an aircraft appear inside a circle within a certain radius, the noise cancellation device broadcasts an anti sound wave that can cancel the jet noise. We will also get reports of noises made by different flights, and this information can be used to find out ways to minimize noises influenced by their routes, altitude or other factors.

Table of Contents

Table of Contents

Chapter 1. Introduction

Noise is a big problem that decrease people's life quality, leading to complex task performance, modifies social behaviour and causes annoyance, one of the prominent one is traffic noise, including aircraft noise, harmful to children that chronic aircraft noise exposure impairs reading comprehension and long-term memory and may be associated with raised blood pressure[1]. Jet airplane noise is a major concern for many areas around the world, and especially in the metro LA/OC area. These cities are not alone. Newport beach has spent considerable resources combating this around SNA. Based on those points above, using a distributed network of sensors is likely a good way to track aircraft noise and separate it away from street and environmental noise for auditing, remediation, and compliance purposes. Therefore, tracking airplane noise is the first objective before separation. We match each fight with the noise spectrum data we collected along with the time and location information for further analysis.

1.1 Motivation

In modern society, airplane becomes a major transportation tool not only for civil use but also for military use. An airport can have thousands of planes takeoff and land everyday, which caused very serious noise pollution along the airplane paths, especially the areas centered by the airports. Based on the research, the airplane noise is obviously harmful to the public healthy status. Tracking and separating the airplane noise are the original and prior purpose to start this project. Before separating the airplane noise, the most important job has to be done is tracking the flight noise. Besides cancelling the noises directly, analysis of noises and seek for improvement to reduce noises are also considerations. The information results from this tracking system can be used to compare different types of aircrafts and different routes, and solve for optimized route to make smaller noises.

1.2 Contributions

Our group mainly work on building an auditing system for aircraft noise over populations. The device tracks overhead aircraft via received ADS-B signals, while recording ambient audio determine aircraft noise contribution. A summary message is sent back to a database for later analysis for each sampling period.

Chapter 2. Background and Related Work

2.1 Background

Considering the serious influence of airplane noise to the areas under the flight path or near to the airport, tracking the airplane noise and separating it away from street and environmental noise has been pointed out as priority. Hence, the main purpose of this project has been addressed to track the airplane noise with ADS-B which is easy way to implement.

ADS-B: Automatic dependent surveillance – broadcast (ADS–B) is a surveillance technology in which an aircraft determines its position via satellite navigation and periodically broadcasts it, enabling it to be tracked. The information can be received by air traffic control ground stations as a replacement for secondary radar. It can also be received by other aircraft to provide situational awareness and allow self separation.[1] ADS-B broadcasts flights information such as altitude, speed, location in 1090 Megahertz using Amplitude modulation and Manchester coding. Line-of-sight range to a radio is typically 150 nautical miles or greater.[2]

After the Traffic Collision Avoidance System, which requires more complicated equipment, the ADS-B was developed to transmit three-dimensional position and velocity information for aircraft, which uses air-to-air data links to transmit to other aircraft and to the ground station via an air-to-ground data link (ADS -B Out). In turn, other aircraft can use this state information and its own state to calculate and display relative ranges and orientations (ADS-B In). [3]

2.2 Related Work

There are FPGA based ADS-B decoders available that are open source[4]. They demodulate and decode the RF signal and transmitted to a PC to show the air traffic. Strategies have been developed to perform decoding on microcontrollers, CLPDs and FPGAs[5]. The core principle is once we receive the RF signal which is easy and we can decode it into real time flight information. The RF front end can be conventional hardware or can be software defined. The data decoding can be done in a FPGA or or a microcontroller. Our work is different because we use these data tracking flights then recording and processing noise data in the FPGA simultaneously.

Chapter 3. Problem Statement

3.1 How to identify flight noise

If we want to track the flight noise, we need to first figure out what is flight noise looks like. Obviously, we need a microphone to sample all the sound and then do real time spectrum analyse. But how can we distinguish flight noise from the environment? We all know that a moving project will have Doppler effect. Imagine an ambulance car go through the street, you will hear the bell become loud and sharp as it approaches you and gradually dies out as it gets away from you. Flight is this kind of giant moving bell over your head. So we will expected to see some components of our record behave the same way, which are the figures we want to get.

3.2 Power constraints

Since our goal is to build a stand alone node, power is the main constraints is the main concern of our design. Our deployment location might not have a constant power supply or if we use batteries it might be difficult to change the batteries. So we can’t let the microphone sample sound all the time which is power consuming. The smarter way is to sample when there is a fight nearby. But how can our node know? If the node knows its own location and the flights' location, it can calculate the distance between them and smartly start recording when a plane flies over.

3.3 How to get the location information

For the node location we can simply use a GPS receiver which can provide precise location information and timestamp. We can also use the timestamp to mark the events. However how can we know where the flight is? Fortunately, a new surveillance system, which is Automatic Dependent Surveillance Broadcast, is now deployed in most flights. We can receive this broadcast by building a RF front end and the decode the airplane status.

3.4 Receiving ADS-B signal

There are many existing solution to ADS-B RF front end as we mentioned in Section 2.2. The ADS-B is broadcasting at 1090 MHz using Manchester code. We use a serie of amplifier and bandpass filter to get rid of irrelevant frequencies, then use a multiplier to demodulate the signal. Output of RF front end is 1 MHz signal which is still fast. So our choice of decoder should have enough power to deal with it. Considering the fast speed of the ADS-B signal, we should also just do real time analysis rather than record then process. The ADS-B signal has a 8 us preamble field. It will appear two peaks and after 2.5 us there will be another two peaks. Once we detected this pattern, we can start sample the 112 bits ADS-B Manchester code.

3.5 Privacy problem

Since our project will record the sound of environment which may offend the privacy of nearby people. To avoid this situation we should never store any clips of sound we received. This urges us to use a chip that have the ability to do real-time spectrum analysis. It is hard to recreate the original sound if we use only certain frequency points. And we treat the sequence of frequencies as the fingerprint of the flight.

Chapter 4. Technical Approach

4.1 Structural Description

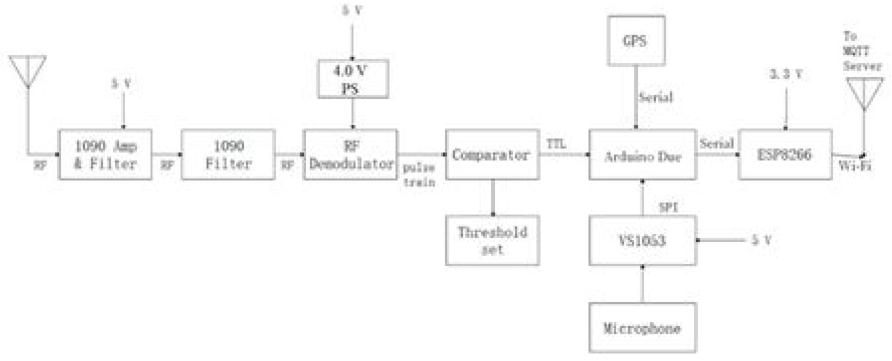


Figure 1. Overall System Structure

The overall configuration of the device is shown in figure 1. The device uses a front-end to receive signal, the ADS-B signal is located at 1090 MHz, 1090 MHz amplifier and filters were used to pass signals of this frequency to a RF demodulator, and the demodulated signals were sent to comparator to eliminate noises, integrated with signals received from GPS, both information decoded by Arduino Due to implement an aircraft’s location with respect to the location of the device, the first algorithm of ADS-B decoder is to decode manchester code into AVR format, and the second algorithm is to decode AVR format into flight information, and GPS chip will give location information and current time. VS1053b in the device is acted as a spectrum analyzer. The report of spectrum analyzer from VS1053b along with aircraft information will be sent to MQTT Server via ESP8266. For demonstration, an arduino is used to integrate information and implement.

ADS-B: Automatic dependent surveillance – broadcast (ADS–B) is a surveillance technology in which an aircraft determines its position via satellite navigation and periodically broadcasts it via satellite navigation and periodically broadcasts it via a transponder, enabling it to be tracked.[1] It broadcasts information such as altitude, longitude, latitude, speed and aircraft type in 1090MHz. The signal is Manchester encoded at 1MHZ in a 56-bit data frame. We have constructed RF modulator and thresholder to provide the raw decoded signal to the Arduino Due for decoding.

VS1053b: This chipset/board is a DSP processor allowing input from an on-board microphone and preamplifier to be sampled and analyzed. The VS1053 can perform real-time spectrum analysis. Our

Arduino Due can control the board and get result via SPI. For simplicity, we will only calculate the spectral fingerprint of noise and transmit the bitstream to Arduino Due. The Arduino will then extract the feature of flight noise and combining them with ADS-B and time data.

GPS: The GPS receiver provides position and time reading for the device, based on the unit to aircraft distance, the system decides whether audio sampling of nearby aircrafts should start. The data coming out from GPS is NMEA sentences. SmartFusion2 decodes and implements GPS receiver via serial port.

ESP8266: The ESP8266 WiFi Module is a self contained SOC with integrated TCP/IP protocol stack that can give any microcontroller access to your WiFi network. Arduino push the UART bitstream to the ESP8266 and a simple MQTT loop will push data to the remote server via Wi-Fi.

4.2 Flow Description

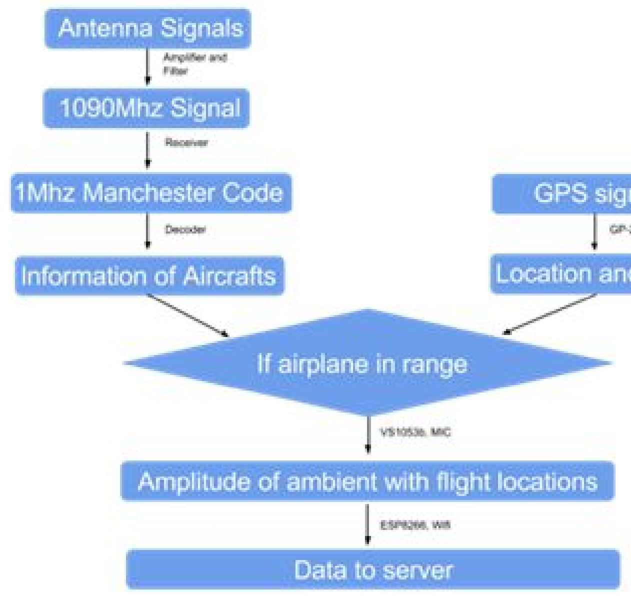


Figure 2. Data Flow Diagram

When signals received via antenna passed through amplifier, filters and comparator, meaning they become ADS-B signals encrypted in manchester code, FPGA decodes them into AVR format, and locations, speed and other information of aircrafts would be collected in FPGA. In the meanwhile, GPS receives location information of the device, and signals are also decoded in FPGA. The FPGA integrate both information and calculate the locations of aircrafts with regards to the device. If the airplane is 5 seconds away (distance/current speed) then microphone starts to work and analyze amplitude of noise. When the report of spectrum analyzer is received by FPGA, the analysis with information of flights would be sent to server via MQTT.

4.3 Communication Description

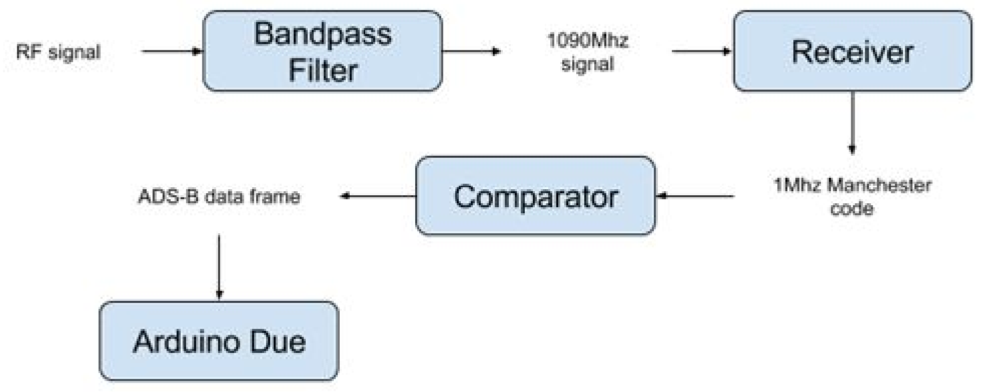


Figure 3. RF front end communication flow

Using RF front end we can filter out the AVR frame but the signal is rather small. The output is around 100 mV. We can not deal with that small signal. So we use a comparator to pull up the signal to 3.3 V as the input of Arduino.

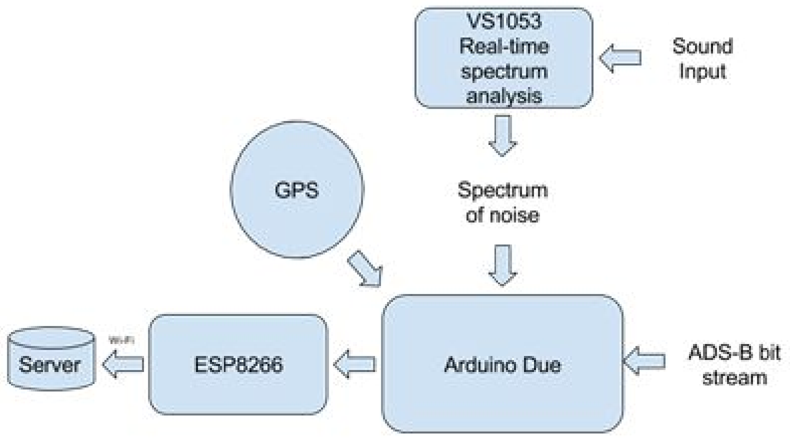


Figure 4. Noise Analyse Communication flow

VS1053 performs real-time spectrum analysis and sends the spectrum to Arduino Due by SPI interface. GPS sends location info to Arduino Due using UART. Arduino Due combine the flight noise and timestamps and send to ESP8266 through UART. ESP8266 connected to the WI-Fi and will push the data to MQTT server whenever possible.

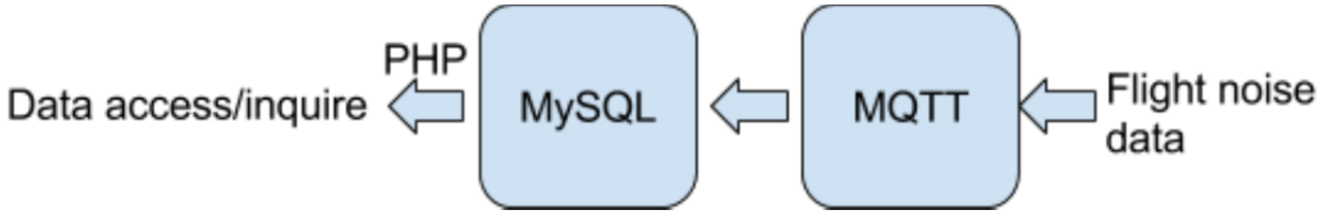


Figure 5. Server flow

A MQTT server is setup so that whenever there is a incoming data fragment it will add an entry to MySQL library. A simple php application is used to take the data from MySQL and display on a static website.

4.4 User Interface

The user interface is a website showing the latests reports for amplitude of noise adhered to flights information on the home page. For more specific demonstration, a map centered at the location of the device is presented, when aircrafts fly nearby, mini log of airplane will show with the flight number of the aircraft, and users can look for detailed information according to the flight number. Historical records of aircraft noises can be found on a page called “history” linked from the home page.

4.5 Technology Options

For RF front end we can use SDR or hardware radio. SDR is more flexible for example we can change the receiver’s functionality by a software upgrade. Also the software is more reliable compare to hardware. Software can reduce cost as well.
We choose FPGA because its parallel processing ability so we can get the real time data. SmartFusion 2 is a perfect choice because it contains a ARM core on board which can handle the network transmission task.
Choosing MQTT for connection because it is helpful to connect with a remote connections, with a small code footprint required or network bandwidth. It transmits the spectrum data from VS1053b and location data from ADS-B to the server.
GPS is required to determine the location and time of the device, so that arduino can calculate the relative position of each aircraft.
VS1053b is used as a spectrum analyzer in the device, to provide amplitude information of aircraft noises, to measure the loudness of aircraft noises.

Chapter 5. Project Validation

5.1 Radio frequency front end

5.1.1 ADS-B SAW Filtered Preamp

This small cell is a bandpass filter, and this particular model is adjusted to be used with the ADS-B frequency (1090Mhz). It also has a low-noise amplifier to pre-amplify the antenna signal. LNA provides at least 16dB gain at 1090Mhz. The insertion loss of the filter is about 2.3 dB, which will increase the total noise figure of the cell to 0.75 dB.

Figure 6. 1090Mhz ADS-B Filtered Preamp

(From https://store.uputronics.com/index.php?route=product/product&product_id=50)

5.1.2 ADS-B 1090MHz Band-pass SMA Filter

1090MHz mode S filter with SMA (male to male) connector. Designed to reduce interference and significantly increase the effective transponder message rate for ADS-B / Mode S receivers.

Through the range of 980MHz - 1150MHz. Impedance is 50 ohms, insertion loss <2.5dB.

Figure 7. ADS-B 1090MHz Band-pass SMA Filter

(From https://www.amazon.com/dp/B010GBQXK8?tag=fligh01-20)

5.1.3 miniADSB Receiver

miniADSB is designed by jetvision.de for non commercial use. Inside the closure we have SAW-Filter - RF-Amplifier (MMIC) - SAW-Filter - Log Amp (Detector)

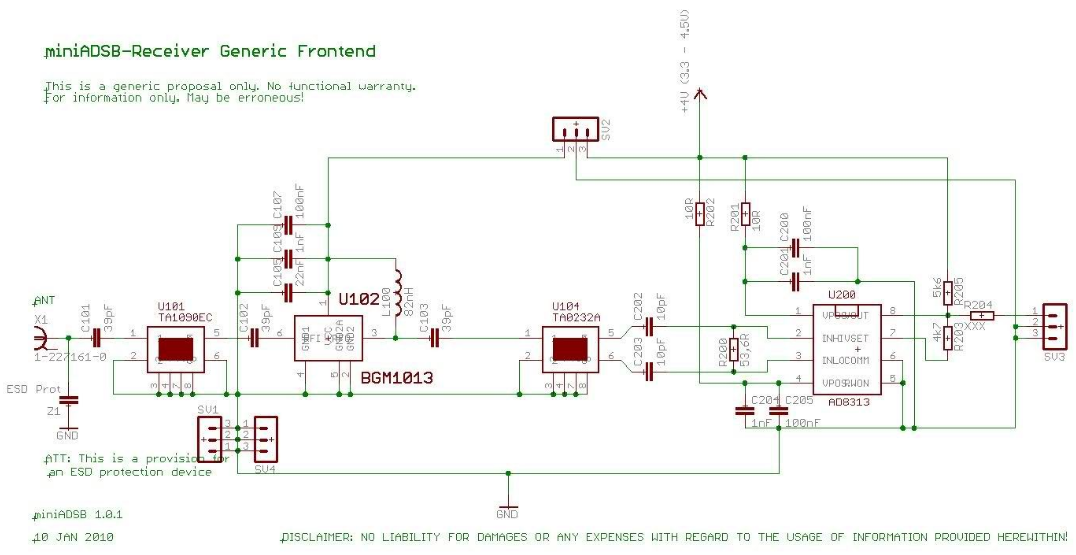


Figure 8. miniADSB Circuit diagram

(By http://miniadsb.web99.de/)

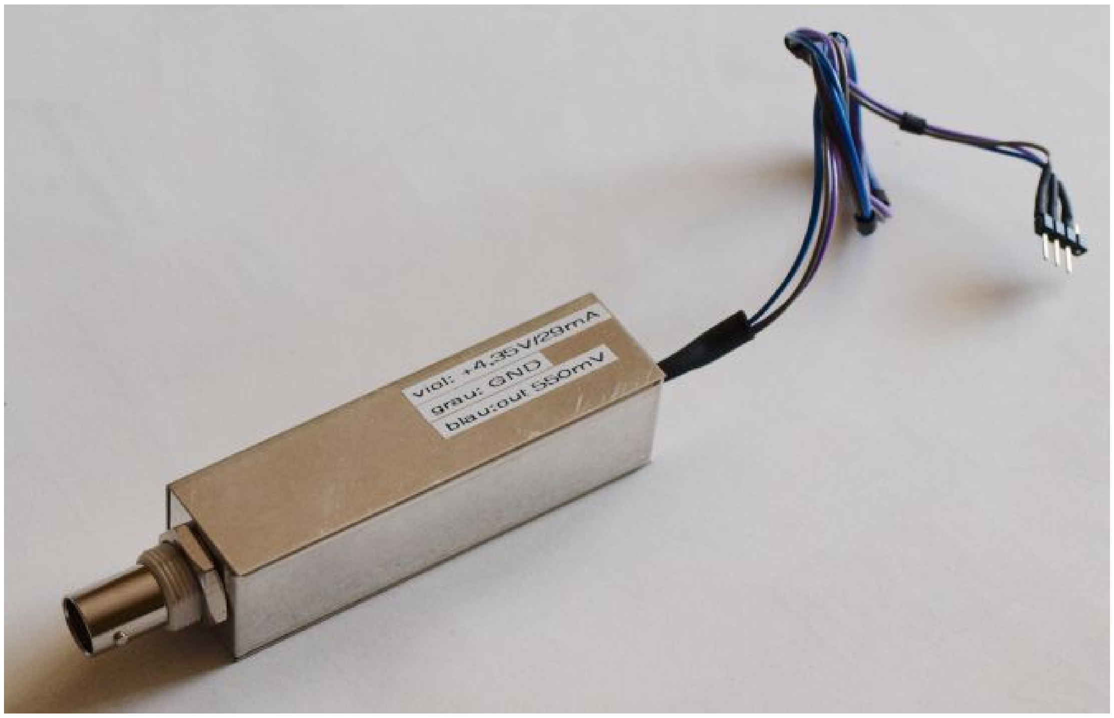


Figure 9. Final Unit

(Michael helped us built this unit)

The RF signal pass through these three component and became a ADS-B data frame.

5.1.4 Comparator

Since the output signal is weak and continuous, we need to convert the analog signal into digital data. We used a comparator to achieve this.

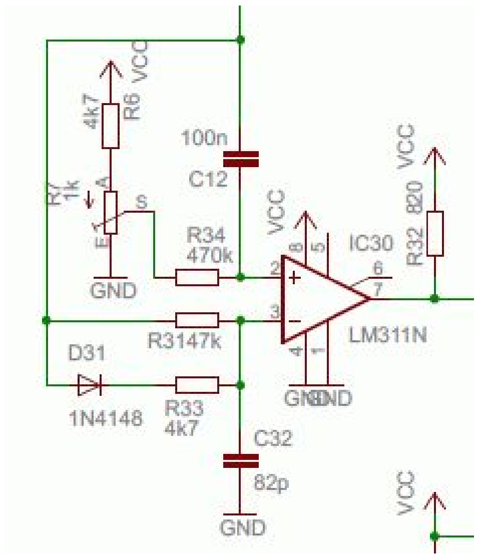


Figure 10. Comparator Design

(From http://rxcontrol.free.fr/PicADSB/index.html)

Core component is LM311N which compare the input and the threshold then generate a bivalue output at 3.3V.

5.2 Arduino DUE

The Arduino Due is a microcontroller board based on the Atmel SAM3X8E ARM Cortex-M3 CPU. It is the first Arduino board based on a 32-bit ARM core microcontroller. It has 54 digital input/output pins (of which 12 can be used as PWM outputs), 12 analog inputs, 4 UARTs (hardware serial ports), a 84 MHz clock, an USB OTG capable connection, 2 DAC (digital to analog), 2 TWI, a power jack, an SPI header, a JTAG header, a reset button and an erase button.

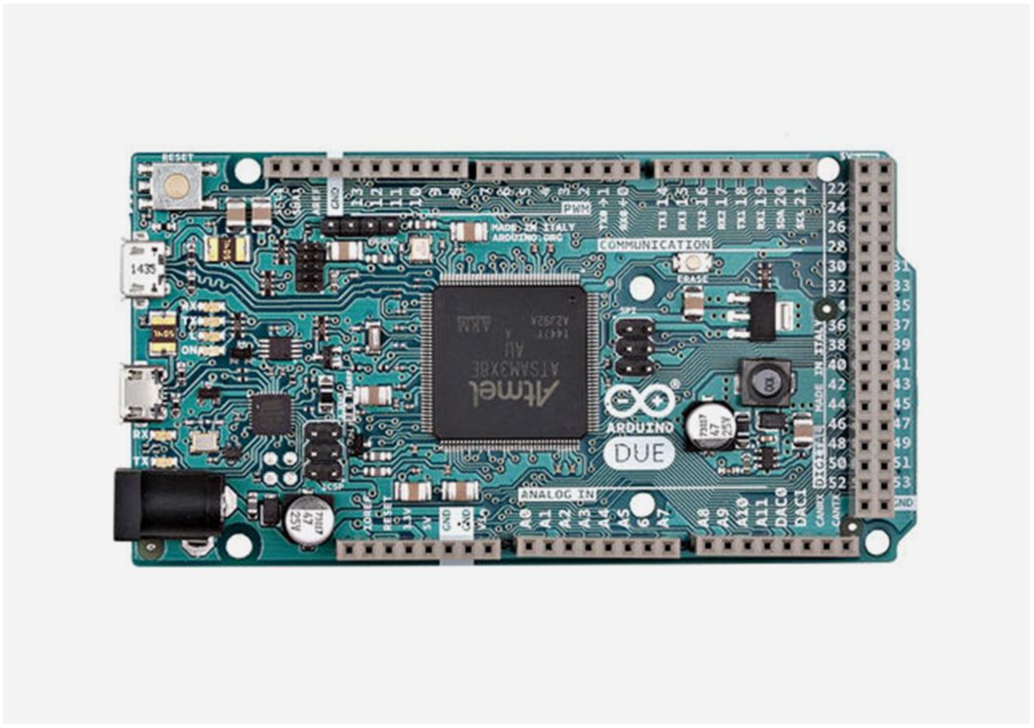


Figure 11. Arduino Due

5.2.1 Decode ADS-B

The ADS-B is encoded in a 1Mhz Manchester code.

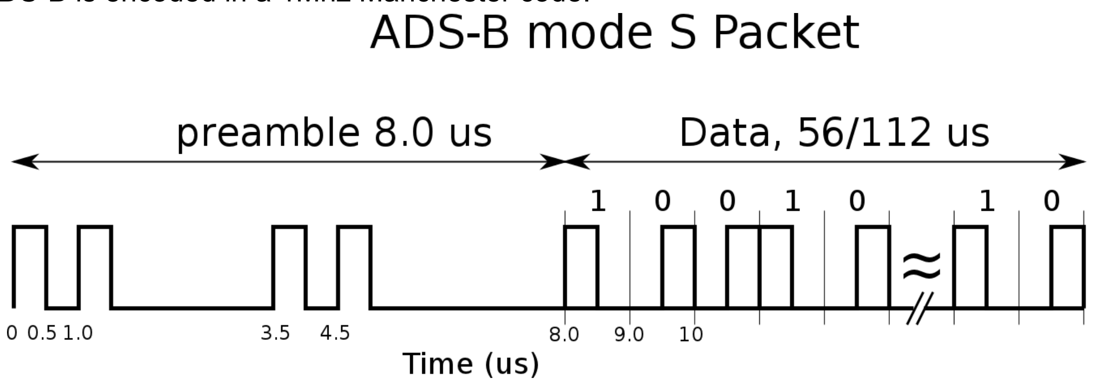


Figure 12. ADS-B signal sample
(From http://www-inst.eecs.berkeley.edu/~ee123/sp14/lab/lab1-Time_Domain_III-SDR.html)

Since the bit-rate is 1Mbit per second, then a sampling rate of 2M sample/sec should be sufficient to capture the bit transitions, detect the preamble and data packets.
If we have detected 1010000001010000000 in our chunks of samples, then the following 56 or 112 bits are the dates we want.
Arduino DUE’s main clock is 84 Mhz which is more than enough for us. We use Arduino timer to detect preamble. The interrupt runs every 0.5 us and we check the change of stats in the input port. Once we detected the preamble we start another timer to sample the data frame.

Once packets are detected, it is very easy to extract the bits of the packets. Recall that each bit is encoded in 2 samples and that the transition defines the bit value. Therefore if sample 1 is greater than sample 2 then the bit value will be 1. Otherwise it will be zero.

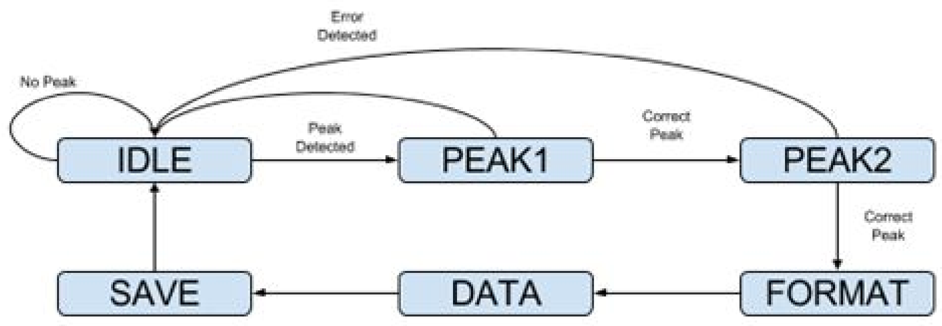


Figure 13. Decode Status Diagram

5.2.2 AVR data format

The ADS-B data obey a AVR data format.An ADS-B message is 112 bits long, and consist of 5 parts:

```
+--------+--------+-----------+--------------------------+---------+
|  DF 5  |  ** 3  |  ICAO 24  |          DATA 56         |  PI 24  |
+-----------------+-----------+--------------------------+---------+
```

This table lists the key bits of a message:

nBits	Bits	Abbr	Name
5	1 - 5	DF	Downlink Format (17)
3	6 - 8	CA	Capability (additional identifier)
24	9- 32	ICAO	ICAO aircraft address
56	33 - 88	DATA	Data
	[33 - 37]	[TC]	Type code
24	89 - 112	PI	Parity/Interrogator ID

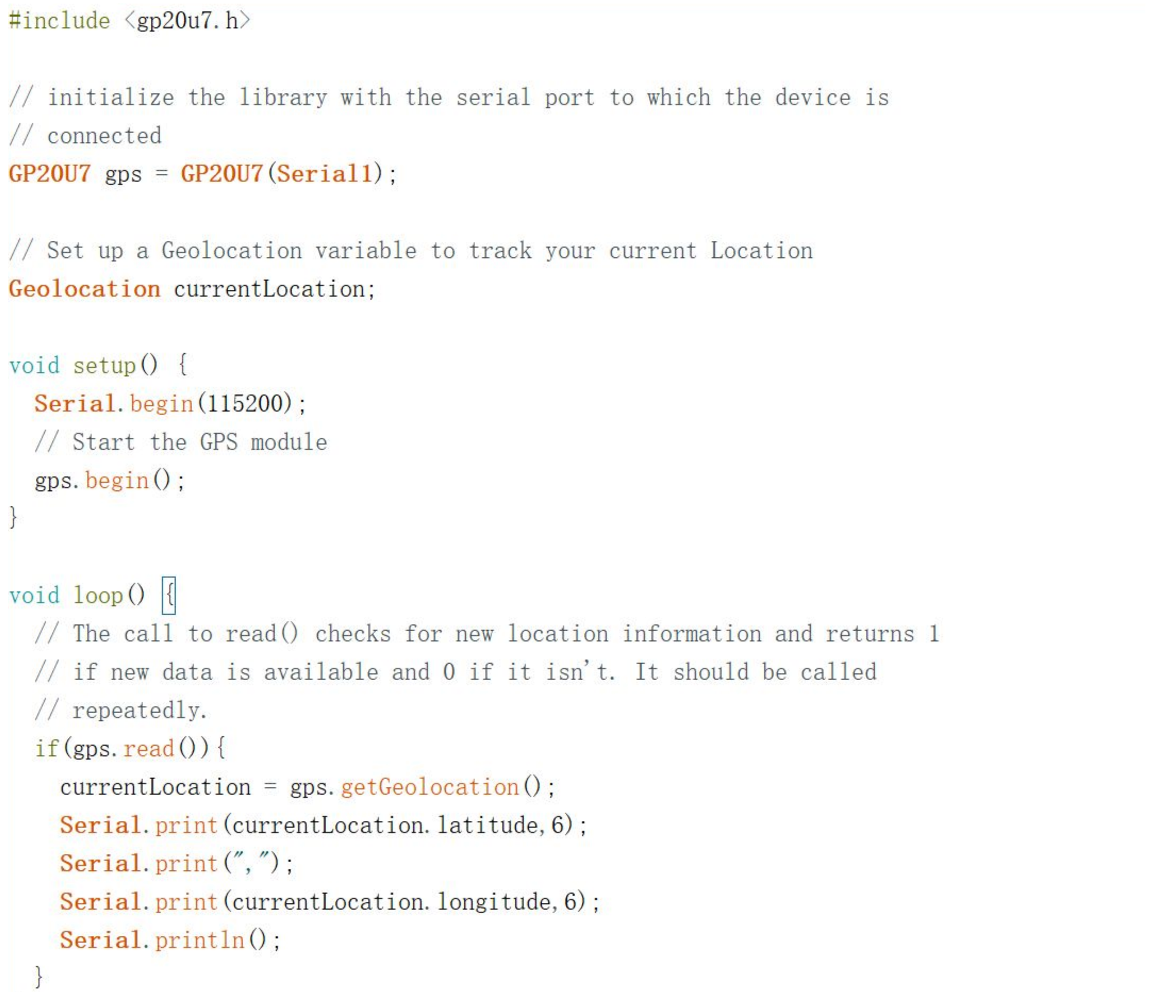
The location information can be calculated easily.

The DATA segment can contain important information such as flight number, three-dimensional position information, and speed, and the information type is determined by the first 5-digit code (TC) of the DATA segment. When TC is 1-4, the data is flight number. Its composition as shown in Table 1 flight number data composition table:

TC (5)	EC (3)	C1 (6)	C2 (6)	C3 (6)	C4 (6)	C5 (6)	C6 (6)	C7 (6)	C8 (6)

5.2.3 GPS

We use GPS Receiver - GP-20U7 to get the location and time information of our node.
Using the library code we can easily get what we want.



```
#include <gp20u7.h>

// initialize the library with the serial port to which the device is
// connected
GP20U7 gps = GP20U7(Serial1);

// Set up a Geolocation variable to track your current Location
Geolocation currentLocation;

void setup() {
  Serial.begin(115200);
  // Start the GPS module
  gps.begin();
}

void loop() {
  // The call to read() checks for new location information and returns 1
  // if new data is available and 0 if it isn't. It should be called
  // repeatedly.
  if(gps.read()){
    currentLocation = gps.getGeolocation();
    Serial.print(currentLocation.latitude,6);
    Serial.print(",");
    Serial.print(currentLocation.longitude,6);
    Serial.println();
  }
```

5.2.4 VS1053b

VS1053b is a single-chip Ogg Vorbis/MP3/AAC/- WMA/MIDI audio decoder and it has the ability to perform digital signal processing like spectrum analysis. We can communicate with VS1053b through SPI interface.

Arduino is mainly used to load patches and retrieve data.

```
  Serial.println(F("Attempting to load Spectrum Analyzer plugin"));
  char pluginname[] = "sa.053"; //Name of plugin
  result = MP3player.VSLoadUserCode(pluginname); //load plugin
  if (result != 0) {
    Serial.print(F("Error code: "));   // return Any Value other than zero indicates a
    Serial.print(result);
    Serial.println(F("  Load function terminated"));
    }
    else {
      Serial.println(F("Plugin Loaded!"));
    }
   SetBands(frequency,nBands);
   //GetAnalysis(data,frequency);
   readFreq();
   Serial.println(nBands);
  }
```

5.2.5 ESP8266

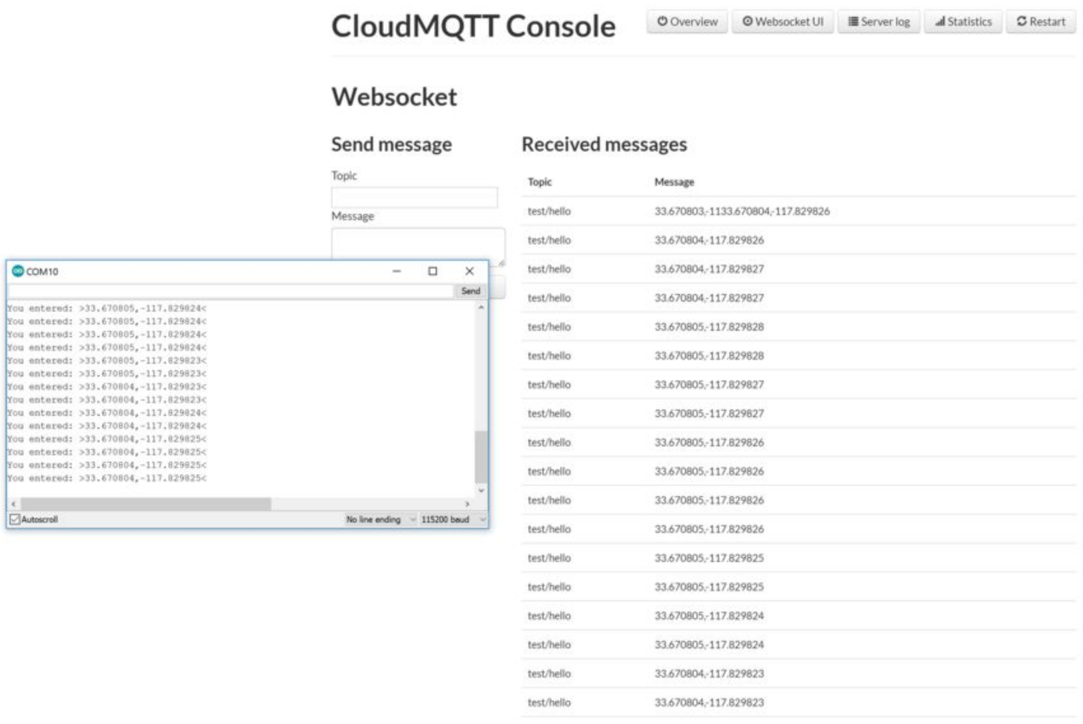
The ESP8266 is a low-cost Wi-Fi chip with full TCP/IP stack and MCU (microcontroller unit) capability. We can program it to do all the network link work. Arduino communicate with ESP8266 through Serial port. Once we detected a fight, we start our microphone and perform spectrum analysis. The spectrum data combined with flight information and time stamp will be sent to ESP8266 through serial port. ESP8266 will then push the data to server.

```
void loop() {

  if (!client.connected()) {
    reconnect();
  }
  client.loop();

  static char buffer[80];
  if (readline(Serial.read(), buffer, 80) > 0) {
    Serial.print("You entered: >");
    Serial.print(buffer);
    Serial.println("<");
    client.publish("test/hello", buffer);
  }
}
```

And we use MQTT protocol to publish our data, so all the clients subscribed to our services will get update.



COM10

```
You entered: >33.670805,-117.829824<
You entered: >33.670805,-117.829824<
You entered: >33.670805,-117.829824<
You entered: >33.670805,-117.829824<
You entered: >33.670805,-117.829823<
You entered: >33.670805,-117.829823<
You entered: >33.670804,-117.829823<
You entered: >33.670804,-117.829823<
You entered: >33.670804,-117.829824<
You entered: >33.670804,-117.829824<
You entered: >33.670804,-117.829825<
You entered: >33.670804,-117.829825<
You entered: >33.670804,-117.829825<
You entered: >33.670804,-117.829825<
```

Received messages

Topic	Message
test/hello	33.670803,-1133.670804,-117.829826
test/hello	33.670804,-117.829826
test/hello	33.670804,-117.829827
test/hello	33.670804,-117.829827
test/hello	33.670805,-117.829828
test/hello	33.670805,-117.829828
test/hello	33.670805,-117.829827
test/hello	33.670805,-117.829827
test/hello	33.670805,-117.829826
test/hello	33.670805,-117.829826
test/hello	33.670805,-117.829826
test/hello	33.670805,-117.829826
test/hello	33.670805,-117.829825
test/hello	33.670805,-117.829825
test/hello	33.670805,-117.829824
test/hello	33.670805,-117.829824
test/hello	33.670804,-117.829823
test/hello	33.670804,-117.829823

The cloudMQTT is a free IoT cloud services which we use to upload our data.

5.3 VS1053b

We use this board as our sound analysis board. To achieve this we connected a external microphone to get better recording.
VS1053b doesn’t have spectrum analyzer built-in. Instead they use a patch to enable this function. Also if we want to use our external microphone instead of the built-in, we need another patch called admoneq.053. After we load this patch, we need to load the spectrum analyzer and set the frequencies it extracts.

```
// Function modified from user 'CheeseBurger' from vsdsp-forum.com
void SetBands(const short int frequency[10], int nBands) {
  if(!frequency || nBands > 23) return;

  //Set band frequencies
  int i;
  MP3player.Mp3WriteRegister(SCI_WRAMADDR, BASE + 58);
  for(i = 0; i < nBands; i++) {
    MP3player.Mp3WriteRegister(SCI_WRAM, frequency[i]);
  }
// Register(SCI_WRAM, 25000); // Terminate partial frequency lists with 25000

  //Reactivate Analyzer
  MP3player.Mp3WriteRegister(SCI_WRAMADDR, BASE + 1);
  MP3player.Mp3WriteRegister(SCI_WRAM, 0);
}
```

Then we can retrieve our data from specific registers.

```
// Function modified from user 'CheeseBurger' from vsdsp-forum.com
void GetAnalysis(unsigned int data[10],const short int frequency[10]) {
  if(!frequency) return;
  delay(100);

  Serial.println("loading");
  //Get nBands
  MP3player.Mp3WriteRegister(SCI_WRAMADDR, BASE + 2);

  Serial.println("loading1");
  int nBands = MP3player.Mp3ReadRegister(SCI_WRAM); //direct integer write of address

  Serial.print("There are ");
  Serial.print(nBands,DEC);
  Serial.println(" bands\n");
  //nBands = 14; // Ignore incorrect nBands

  //Read analysis
  MP3player.Mp3WriteRegister(SCI_WRAMADDR, BASE + 4);
  for(int i = 0; i < nBands; i++) {
    //data[i] = MP3player.Mp3ReadRegister(6);
    data[i] = MP3player.Mp3ReadRegister(6) & 31; //Current value is in bits 0-5
    Serial.print("Spectrum Data: ");
    Serial.println(data[i]);
  }
```

Chapter 6. Development Plan

6.1 Status

The comparator is not stable, so the data frame is usually crapped and hard to extract bits from it. don’t have a user interface for information inquire and management.

6.2 Roles

Time	Major roles
Week 1-4	ADS-B decoder
Week 5	ADS-B receiver (expect comparator) implement & testing
Week 6	All: GPS implement
Week 7	All: VS1053 implement
Week 8	All: ESP8266 implement
Week 9	debug GPS and ADS-B decoder & testing
Week 10	poster; All: video

6.3 Future Work

Because the demonstration is achieved via arduino, but the final project uses FPGA, the decoders and implements currently written in arduino needed to be transferred into FPGA using verilog. The connections are similar for two boards.

The project is a single node implementation, a large scale with multiple nodes would be needed for practical usage.

References

[1] Stephen A Stansfeld and Mark P Matheson (2003). *Noise pollution: non-auditory effects on health*. [online] Available at: http://www.kensingtonassociation.org.au/wp-content/uploads/2013/10/Noise+Pollution_non-auditory+effects+on+health.pdf [Accessed 12 Dec. 2016].

[1] En.wikipedia.org. (2016). *Automatic dependent surveillance – broadcast*. [online] Available at: https://en.wikipedia.org/wiki/Automatic_dependent_surveillance_%E2%80%93_broadcast [Accessed 6 Dec. 2016].

[2] Faa.gov. (2016). *ADS-B – Frequently Asked Questions (FAQs)*. [online] Available at: https://www.faa.gov/nextgen/programs/adsb/faq/ [Accessed 6 Dec. 2016].

[3] A Brief History of ADS-B. (2011, December 17). Retrieved June 02, 2017, from http://adsbforgeneralaviation.com/a-brief-history-of-ads-b/

[4] Wiki.modesbeast.com. (2016). *Mode-S Beast:About the Mode-S Beast - Beast Wiki*. [online] Available at: http://wiki.modesbeast.com/Mode-S_Beast:About_the_Mode-S_Beast [Accessed 6 Dec. 2016].

[5] Cardew, E. (2016). *Very Simple ADSB receiver*. [online] Lll.lu. Available at: http://www.lll.lu/~edward/edward/adsb/VerySimpleADSBreceiver.html [Accessed 6 Dec. 2016].

[6] Orlando, Vincent A. "The mode s beacon radar system." *The Lincoln Laboratory Journal* 2.3 (1989): 345-362.