A Cost-effective Automated Weather Reporting System AWRS for The Canadian Remote Northern Air Operators

Houssam Hammoudi
CEO, TradeSec Corp
Calgary, T3N 1T4, Canada

Abstract

Air transportation is essential for Canada’s and US northern communities. It is the leading lifeline supplying fresh food, medicine, and other goods; providing Health care services; medical emergency evacuation; and supporting travel outside of the communities. In addition, air transportation is the only reliable year-round mode of transportation. However, the Canadian north and Alaska present significant operational challenges mainly due to inhospitable terrain, harsh conditions, extreme cold. The challenges are financial as well due to low passenger volumes and high operations costs. This research shows how northern Air operators can enhance flight safety using WX-Ready as an AWRS “Automated Weather Information System” to get vital weather information without investing in expensive commercially available ACARS systems (Government of Canada, 2017).

Keywords: Software Engineering, ACARS, Python, Linux, METAR, TAF, Aviation, Air Operators.

1. INTRODUCTION

Aviation is an essential lifeline of the world’s economy; it supports more than 65.5 million jobs worldwide and enables $2.7 trillion in global GDP (Government of Canada, 2017). The aviation sector in Canada generates more than 632000 jobs and $49 billion to contribute to the Canadian GDP. Although Canada has 500 airports, only about half of these airports have scheduled commercial flights. However, most of these airports are small. From these 500 airports, only 117 are remote northern airports scattered across the vast Canadian north. In addition, many northern communities operate ice strips in the arctic region, which are not certified nor listed as airports by Transport Canada. (Government of Canada, 2017).

2. OBJECTIVES

The study aims to develop a Python/Linux Weather retrieval and display system that replaces the expensive commercially available ACARS systems and enhances flight safety without a significant investment. Small operators cannot afford a half-million dollars ACARS solution. Our objective is to fill this need and fix this operational problem in the industry by providing an affordable AWRS system that can enhance flight safety at minimal costs. In addition, we will leverage the GSM/Cellular network and its SMS capability to send and retrieve messages. We will also set a concept for future development using Iridium Satellite Short Burst Data SBD messaging.

3. LITERATURE REVIEW

Reviewing similar research papers, we found that most of the research papers focused on operations within commercial airports (Benjamin & Moninger, 2016). We, however, focused on northern operators operating in and from ice strips and small northern airports. We realized that most of the literature is only aimed at established airlines with big commercial jets; we couldn’t find anything related to small operators, charter or commuter operators with aircrafts less than 19 passengers. Another observation was that none of the studies covered Medivac operators that are vital in Canada (International Civil Aviation Organization, 2010).
3.1 The Importance of Aviation Weather
The aviation weather systems have evolved tremendously since the 1980s; in addition to fixed weather observation stations, aircraft with their onboard sensors take atmospheric measurements and share that information with other aircraft and ground stations via the Aircraft Meteorological Data Relay (AMDAR). (Anaman et al., 2017; Federal Aviation Administration, 2009).

3.2 Important Weather Readings
Aviation weather information comes in the form of forecasts (Federal Aviation Administration, 2009) observations, and notices. We are going to focus only on the following:
- METAR – Meteorological Terminal Air Report
- TAF – Terminal Aerodrome Forecast
- NOTAM – Notice to airmen

4. PROJECT DESIGN AND METHODOLOGY
The methodology used in this study follows the software prototyping development model (Susanto et al., 2019) This model allows building, testing, and refining the software prototype based on end-user feedback until a good working version is accomplished.

4.1 The Requirements Gathering and Analysis Phase
The requirement gathering and analysis phase consisted of being on the field gathering all the observations made by the pilots, the dispatchers, and the company management. We witnessed the lack of weather information during the study during ferry flights over remote northern areas and transoceanic flights.

4.2 The Quick Design Phase
This section presents a quick and straightforward framework showing how the user interface will appear.

4.3 The Build Phase
This section produces a quick and simple design that we call an MVP a “Minimum Viable Product.” This version of the apparatus included only four features. This phase allowed us to gather valuable feedback about the product

4.4 Field Evaluation Phase
In this section, we describe how we used the apparatus in the field during transoceanic flights and flights conducted in the Arctic region. This phase was crucial to gathering valuable feedback from the pilots, allowing us to determine its strengths and weaknesses and the must-have features.

4.5 Refining The Prototype Phase
In this section, we measured the impact of the apparatus, we determined the nice-to-have features and the must-have features, and we were able to have an approved final prototype.

5. HARDWARE COMPONENTS
This section will describe in detail the different hardware components used in this project and explain their roles.

5.1 Single Board Computer
This apparatus combines software and a single board computer that needs to bring a small size factor, versatility, and upgrade either by expansion boards or hardware attached on top boards (HAT) and cost-effective. We found all these characteristics in the Raspberry Pi 3 Model B+ shown in figure 1 below.
5.2 SIM800 GSM/GPRS HAT Expansion Board
This HAT board is explicitly made for the Raspberry Pi. It attaches on top of the Pi by connecting to the 40 GPIO pins shown in figure 2.

5.3 TFT 3.5-inch LCD Touch Screen
The 3.5-inch LCD touch screen displays the weather data and allows users to make selections and input airport codes. The LCD screen connects to the raspberry pi via the 40 GPIO pins of the SIM800 board shown in figure 3.
6. USER INTERFACE AND NAVIGATION

We chose to develop the Linux Debian-like OS called Raspbian because it provides a cost-effective, stable, and robust platform. We also decided to write the UI using the Tkinter Python framework due to its various GUI elements and ease of use.

6.1 Airport Selection

6.1.1 ICAO CODES UI Component

This section of the UI allows the pilot to enter all airports they wish to retrieve weather info. Commas can separate the airport codes. The crew members can select which information they want to receive by selecting the METAR, TAF, NOTAM, and GFAs as depicted in figure 4.

FIGURE 3: TFT 3.5-inch LCD Touch Screen.

FIGURE 4: The airport, METAR, TAF, and Notam selection.
6.1.2 METAR Weather Info Display UI Component
This screen depicted in figure 5 below shows the METAR information of every airport in the airport list.

![METAR Information](image)

**FIGURE 5:** METAR information displayed.

6.1.3 TAF Weather Info Display UI Component
The figure 6 below shows the METAR information of every airport in the airport list.

![TAF Information](image)

**FIGURE 6:** TAF information displayed.
6.1.4 NOTAM Weather Info Display UI Component
The figure 7 below shows the NOTAM information of every airport in the airport list.

![NOTAM Information Display](image)

**FIGURE 7:** NOTAM information displayed.

7. SYSTEM ARCHITECTURE
The system architecture depicted in figure 8 is comprised of different components and services.

- Webserver containing the web service listener
- SMS 3rd party service API “Twilio.”
- SMS 3rd party webhook service “Twilio.”
- SIM800 receiving and sending SMS messages

![System Architecture Diagram](image)

**FIGURE 8:** Systems architecture diagram.
7.1 Webserver
The web server is an Apache server running on a Debian Linux OS. We chose Apache for its ease of use and stability. The webserver is hosted publicly and responds only to HTTPS calls for added security.

7.2 Twilio Messaging Service
Twilio is a cloud communication platform allowing developers to send calls, SMS, and MMS messages using their many web service APIs. Moreover, the inbound and outbound SMS message prices are minimal, costing only $0.0075 per message. This price point makes the WX-Ready very attractive and advantageous for northern operators who cannot afford a hundred thousand dollars ACARS system.

7.3 Twilio Webhook Service
Twilio is a cloud communication platform allowing developers to send calls, SMS, and MMS messages using their many web service API’s.

7.4 Twilio Webhook Service
Twilio is a cloud communication platform allowing developers to send calls, SMS, and MMS messages using their many web service API’s.

8. WEATHER INFORMATION RETRIEVAL BACKEND
8.1 Backend METAR Info Retrieval “GET request.”
The backend component is the code that runs to retrieve the weather observations of the specified airports. This backend code uses API calls to an aviation Weather REST API service. The METAR has its specific endpoint shown in figure 9 and figure 10.

https://avwx.rest/api/metar/cyzs

FIGURE 9: METAR API Endpoint

FIGURE 10: Python Code used to request weather data in.

The airport list contains all the airports the crew inputs into the UI. We then authenticate to the endpoint using the bearer token method. After the authentication, we modify the server response to a JSON format.

8.2 Backend METAR Info Retrieval “JSON data consumption.”
Now that our response is in JSON format as shown in figure 11, we place each weather observation in its variable, as shown in figure 12.
FIGURE 11: METAR weather data in JSON format.

```python
# METAR Altimeter information
METAR_Altimeter_results = json_data['altimeter']
altimeter = METAR_Altimeter_results['repr']

# METAR Clouds
cloud_coverage_list = []
METAR_Clouds_results = json_data['clouds']
for cloud_results in METAR_Clouds_results:
    cloud_coverage = cloud_results['repr']
    cloud_coverage_list.append(cloud_results)

# METAR Wind direction
METAR_wind_direction = json_data['wind_direction']
wind_direction = METAR_wind_direction['repr']

# METAR Wind Speed
METAR_wind_speed = json_data['wind_speed']
wind_speed = METAR_wind_speed['repr']

# METAR Visibility
METAR_visibility = json_data['visibility']
visibility = METAR_visibility['repr']

# Flight Rules
METAR_Flight_rules = json_data['flight_rules']
```

FIGURE 12: Python Code to dissect JSON data into variables.
8.3 Backend Weather Info Display In METAR Format
We can display the information gathered to the crew by printing the variables while respecting the International Civil Aviation Organization ICAO METAR format clearly shown in figure 13.

![METAR weather info displayed in ICAO format.](image13)

8.4 Backend TAF Info Retrieval “GET request.”
This section of the backend code is the same as the METAR section. The only difference is that the TAF section uses its endpoint, as shown in figure 14.

https://avwx.rest/api/taf/cyzs

![TAF API Endpoint.](image14)

8.5 Backend Weather Info Display In TAF Format
In figure 15 We display the TAF information retrieved to the crew by printing the variables while respecting the ICAO TAF format.

![TAF weather info displayed in ICAO format.](image15)

8.6 Backend Sending Weather Data through Cellular Network
Once we have gathered all the weather data, the Twilio messaging service will reply to the same phone number that requested the info shown in figure 16.

![Weather data transmission via SMS using Twilio API.](image16)
9. THE WORKING PROTOTYPE
This section shows how WX-Ready is operating in figure 17, figure 18, figure 19 and figure 20.

FIGURE 17: WX-Ready main screen.

FIGURE 18: WX-Ready METAR screen.
10. FUTURE ENHANCEMENTS
WX-Ready is eventually intended to operate in the most remote areas in the world. To reach this goal, the apparatus needs to send the requests via the GSM network and through the Iridium
sateellite network when the GSM network is not available. Since it is based on the Raspberry PI hardware platform, the WX-Ready Apparatus can easily be connected to a Rockblock Mk2 Iridium modem shown in figure 21. This modem can send SMS messages through the Iridium Satellite Constellation via the SBD Short Burst Data communication protocol. Adding this capability to WX-Ready does not change the systems’ architecture discussed above because all we do is sending the SMS message through a different network. The backend servers and processes remain the same.

FIGURE 21: RockBlock Satellite Modem.

11. CONCLUSION
The developed prototype performed its job correctly. In the future, we should add additional features to the software and upgrades to the hardware. Currently, the system can operate anywhere in Canada, provided it’s within Cellular coverage. To have WX-Ready work virtually anywhere globally, we can add an Iridium transceiver that can communicate with the Iridium satellite constellation using the Short Burst Data protocol (SBD). Currently, we are getting the power from a USB connection; however, we can also add a battery to provide power to the device. WX-Ready costs less than 40$ and has an operating cost of 10$ per month. This apparatus is the cheapest way to retrieve aviation weather information in remote areas versus commercially available ACARS solutions with prices ranging from $200.000 to $400.000.

WX-Ready benefits airline operators in the Arctic and remote regions like Africa, South America, and Asia.

12. REFERENCES


