HARP Payload APRS Tracker System Technical Writeup

By: Michael Gentry

ECE 528

# Background and Problem

Mercer University is currently running a student led research program for high altitude research, called HARP. The goal of the High Altitude Research Platform is to implement a low cost solution for parties looking to conduct research tests in the upper atmosphere. This is done by creating a research payload filled with sensors and instrumentation and attaching that payload to a hydrogen filled balloon. The balloon is capable of going up to an altitude of 100,000 feet. At that altitude, the balloon pops and the payload falls to the ground. Telemetry data from the balloon is transmitted in packet form by a radio system called APRS. Data in each packet includes GPS coordinates, altitude, time, and some limited information of the users choosing.  
  
The problem is when the payload falls to the ground. The balloon’s position is tracked by a ground team using its APRS data. The ground team tracking the balloon loses contact with the payload when it falls below a certain height and the terrain blocks the transmissions. When the ground team loses contact, the balloon can still travel for a large distance. This creates a large zone of uncertainty where the fallen payload could be, creating a long and difficult search for the ground team attempting to recover it.   
  
This project is for creating an embedded system that can greatly assist the ground team in locating the fallen payload.

# Proposal

My proposal for this is two fold. The first part is a drone based platform, a drone repeater in a sense, with equipment capable of demodulating and decoding the APRS signals in order to extract the GPS location of the fallen payload. The next portion is navigation unit capable of taking that GPS location and allowing the user to get close to the fallen payload. The drone repeater and handheld unit will communicate with each other by using a pair of radio transceivers using the Lora spread spectrum modulation standard. So the drone repeater will fly over the large zone of uncertainty, picking up the APRS signal from the payload and sending it to the handheld navigation unit so the user can start finding his way to the balloon.

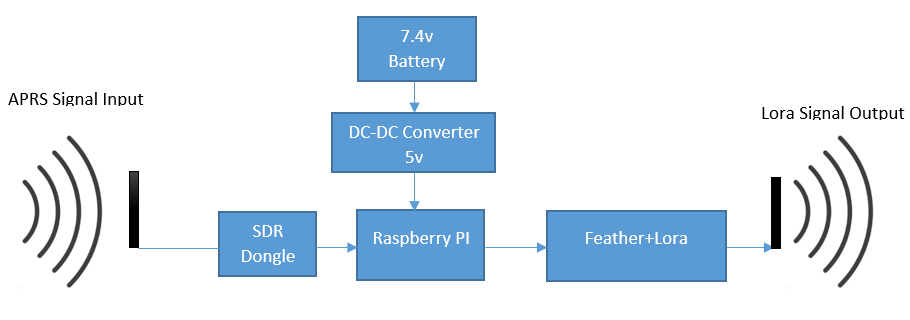
# Prototype

## Drone Portion Repeater

The drone repeater works by having a SDR run by a Raspberry PI demodulating the APRS signal and decoding it with software. The Raspberry Pi filters out the useful information from the full packet then sends the important information to a Feather m0 over serial connected by USB. The Feather m0 then attempts to send the data to the handheld unit using the Lora radio module built in to the Feather m0. The Feather m0 is programmed with a robust communication protocol to ensure the drone repeater sends the data to the handheld unit correctly.

### Hardware

The hardware is relatively simple. The raspberry PI is powered by a 7.4 volt battery run through a 5 volt converter. The SDR USB dongle is connected directly to the raspberry PI and it has a 144 MHz tuned antenna better receive the APRS signals. The Feather m0 is connected to and powered by the PI’s USB port by a type A to micro USB cable. The chassis of the repeater package is made of acrylic sheets. The components are held down by standoff screws.

****

### Software

The software on the raspberry pi consists of three different programs. The OS on the raspberry PI is Raspbian Stretch with desktop version 4.14. The program used to demodulate the RF signals from the SDR is RTL\_fm from git://git.osmocom.org/rtl-sdr.git. This outputs an audio of the baseband signal. The next step is to run the baseband audio into a program to decode the APRS data. The decoding software is used was “multimon-ng” from https://github.com/EliasOenal/multimon-ng/. This will output a string in the terminal of the decoded APRS packet. I created a python program that uses the subprocess library to run the following command in a subprocess:

“ rtl\_fm -f 144.390M -s 22050 -g 100 - | multimon-ng -a AFSK1200 -A -t raw -”

The python program then captures the outputted string, uses string methods to search for and isolate the important gps location, altitude and timestamp. The program then sends that data to the Feather m0 using the serial USB connection.

OS, programs, and libraries:  
Raspbian Stretch with desktop version 4.14  
  
RTL\_fm  
multimon-ng  
Custom python program  
  
using shlex  
using serial   
using subprocess  
  
The Feather m0 has an ARM processor and is programmed using the Arduino IDE. The program on here is dedicated to receiving the data form the raspberry PI serial connection and sending that data to the handheld unit. The program constantly checks for new data from the serial and tries to establish a connection with the Feather m0 on the handheld unit. A robust message handler was implemented to ensure the data arrives at the handheld correctly and error free.

Libraries used on the Feather#include <SPI.h>  
#include <RH\_RF95.h>

## Handheld Unit

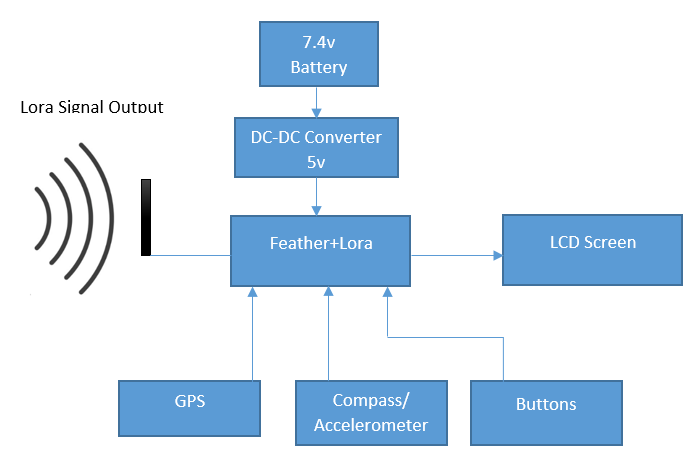
The main use of this is to receive GPS location data from the repeater unit, determines its own GPS location, calculate a bearing between the handheld unit and the payload, use a compass to determine the magnetic bearing of the user, and use that to guide the user to the fallen payload.



### Hardware

#### Connections

The handheld unit is similarly powered by a 7.4volt battery run through a 5 volt output DC-DC converter. This directly powers the feather m0. A power switch is connected in series between the battery and converter. All other components are hooked up with the Feather m0. The compass/accelerometer and LCD screen are connected to and powered the Feather m0 and communicate with I2C. The LCD screen is connected with a 5 volt to 3.3 volt level shifter. The GPS is connected to the feather’s serial pins and has a separate GPS antenna. The buttons wires run to the digital input pins on the Feather.

****

#### Layout

The handheld consists of a box with most of the components on the inside. The buttons and LCD screen are attached to the lid of the box. The power switch is on the attached to the lower portion of the box. The Feather m0, compass/accelerometer, and logic level shifter are soldered onto a proto board. The proto board, battery, DC-DC converter, switch, GPS, and GPS antenna are attached to a sheet of acrylic which is screwed to the bottom of the box. The buttons and LCD screen are attached to the lid of the box. The lid of the box has a hinge on it.

### Software

The program on this feather is rather large and separated into several parts. One part of it is responsible for establishing a radio connection with the drone payload. The main part is dedicated to reading from the GPS and the compass, calculating bearings and magnetic compass directions, and calculating distance between the fallen payload and the handheld unit. Another part of the program is for displaying more detailed information about the current GPS location of the handheld. Another part of the program is for displaying more detailed information about the compass and accelerometer. Another part of the program is for displaying the information about the last received message.

Libraries on the feather  
#include <SPI.h>  
#include <RH\_RF95.h>  
#include <Wire.h>   
#include <LiquidCrystal\_I2C.h>  
#include <LSM303.h>  
#include <TinyGPS++.h>  
#include "math.h"

# Budget

Below are some basic estimated costs for both the Drone Repeater Unit and the Handheld Unit. Prices for small things like screws, nuts, cables, and hinges are not included.

|  |  |  |  |
| --- | --- | --- | --- |
| **Drone Package** | | | |
| Item | Price | Quantity | Cost |
| ABS box | $ 12.00 | 1 | $ 12.00 |
| Raspberry PI 3 | $ 37.00 | 1 | $ 37.00 |
| Feather m0+Lora | $ 33.00 | 1 | $ 33.00 |
| Battery | $ 8.00 | 1 | $ 8.00 |
| DC-DC | $ 6.00 | 1 | $ 6.00 |
| Acrylic | $ 5.00 | 1 | $ 5.00 |
|  |  | Total | $ 101.00 |

|  |  |  |  |
| --- | --- | --- | --- |
| Handheld | | | |
| Item | Price | Quantity | Cost |
| ABS box | $ 12.00 | 1 | $ 12.00 |
| logic shifter | $ 7.50 | 1 | $ 7.50 |
| LCD screen | $ 10.00 | 1 | $ 10.00 |
| Buttons | $ 8.00 | 1 | $ 8.00 |
| Feather m0+lora | $ 33.00 | 1 | $ 33.00 |
| GPS | $ 15.00 | 1 | $ 15.00 |
| Compass | $ 10.00 | 1 | $ 10.00 |
| Dc-DC converter | $ 6.00 | 1 | $ 6.00 |
| Switch | $ 3.00 | 1 | $ 3.00 |
| Battery | $ 12.00 | 1 | $ 12.00 |
|  |  | Total | $ 116.50 |

# Testing

Testing of this was not done with a drone. All tests were done with the antennas less than ten feet to the ground. Using the track soar to send APRS signals, the system was able to fully decode the signals and send the data to the handheld unit. The range for this varied based on terrain. The link between the track soar and the drone package was consistently more than 1 kilometer at ground level with moderate wooded terrain. The link between the Drone package and the handheld unit was less stable at less than 1 kilometer with little trees at ground level.

The ability of the handheld to navigate the user to the payload was mostly successful. Regular calibration of the compass must be done for good results and due to GPS inaccuracies, the handheld unit can narrow down the search area to a zone 20 meters in radius before it starts pointing you in the wrong direction. Sometimes the handheld could navigate the user directly to the payload.

# Conclusion and Recommendations

The project was a success overall even though no full test was done with a drone. The full system can be built for less than $250.00 if appropriate tools and supplies are on hand. Further development of this project is planned in the future and will be done by myself. Some basic recommendations in order to optimize this design are as follows. For the drone repeater portion, a Raspberry PI Zero instead of a Raspberry PI 3 B+ can possibly be used in order to save weight as well as a smaller battery. For the Handheld Unit, a PCB can be used to keep the circuit layout compact instead of using proto boards screwed onto acrylic sheets. Nonvolatile memory could be added in order to keep the payload GPS data in case the power is accidently cut off to the handheld. Status LED’s for power and GPS could also be added.