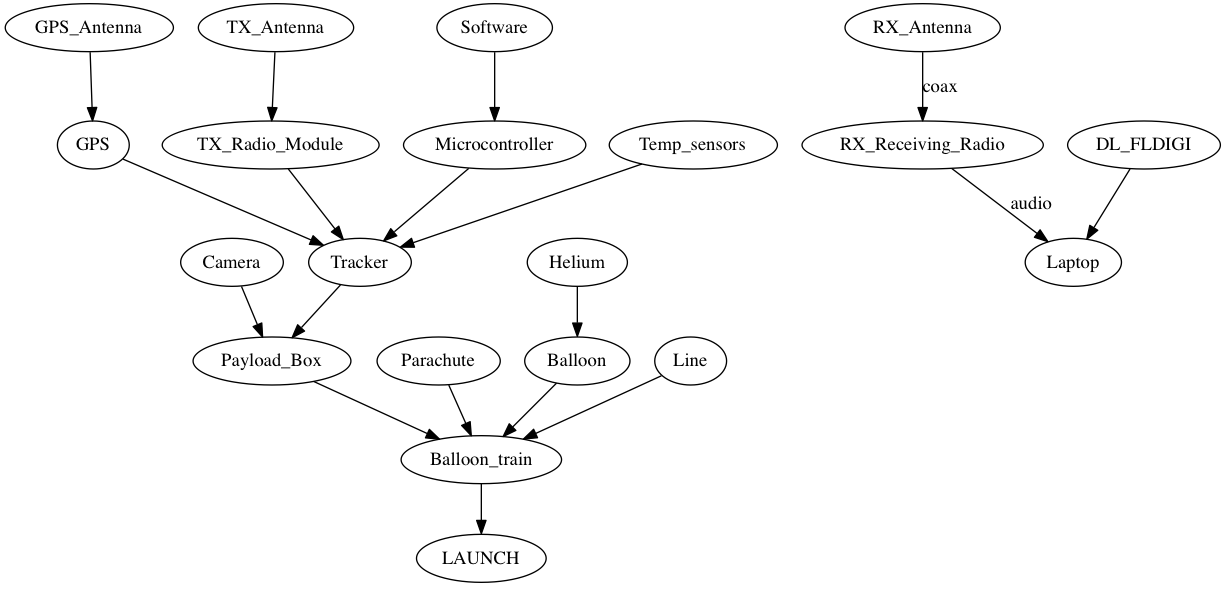
Landing predictor -> <http://predict.habhub.org/>



ACKERMAN CODE

CODE - > <http://www.daveakerman.com/?p=2059>

1. During ascent, it splits the vertical range into 100 metres sections, into which it stores the latitude and longitude deltas as degrees per second.
2. Every few seconds, it runs a prediction of the landing position based on the current position, the data in that array, and an estimated descent profile that uses a simple atmospheric model (from Steve) plus default values for payload weight and parachute effectiveness.
3. During descent, the parachute effectiveness is measured, and the actual figure is used in the above calculation in (2).

So, basically, for each vertical 100m band, the software calculates the estimated time to fall through that band, and applies that to the latitude/longitude deltas measured during ascent.  It then sums all the resulting deltas for descent to 100m (typical landing altitude), adds them to the current position, and emits the result in the telemetry as the predicted landing position.

Although the habhub online map does its own landing prediction, an onboard prediction has some advantages:

* It has more descent data to work with, so can more accurately profile the parachute performance
* It is using more recent wind data, measured during ascent
* Ground chase crews can see the landing prediction without having internet access

There are disadvantages too.  Because it uses wind data from the ascent, if the wind has changed (due to the landing being in a different area, or because the wind is changing with time) then those factors will introduce errors.

Also, I have a suspicion that the live map consistently overestimates the horizontal distance travelled by a descending flight.  This can be seen by watching its landing prediction which, as the flight descends, will move back towards the actual flight position.

So I was keen to see how well the onboard prediction fairs against the habhub prediction.  Steve Randall was also interested in this, and was kind enough to record the descent on his screen.  He has sped up and annotated the video which you can see here:

Noise at low altitudes is less important, as it’s being applied to a short remaining distance to fall, but the noise higher – between say 5000 and 15,000m – is more important.

For my next flight, I’ll apply some filtering to hopefully make the prediction more consistently accurate.  I have all the GPS data from this flight and I can run that back into the tracker code to test how well it would have worked on that last flight.

QUESTIONS TO ASK:

**-over how great a distance can one receive telemetry from the balloon? How far does the balloon normally land from the launch site in general?**

-maximum range is ~200 km beyond which you would terminate flight

**-What, if anything, can affect the quality of the signal?**

-out-of-band interference. Low potential for interference.

**-How often does the balloon transmit data and in what format is it normally encoded?**

RTTY – ASCII format. 300 chars per second. Checksum type algorithm making suremessages are valid.

**-What algorithm is used to predict the landing site? -Just apply kinematic equations and model wind using data from ascent.**

Layers of wind

predict.habhub.org

rivers ses act

tracker.habhub.org

NOTES ON TRACKING

Carrier frequency moving issue – frequency changes over time – find way to address this.

SDR – software defined radio – uSB thing

Tuner software -gqrx (on MAC) – 434.25 MHz – USB modulation

To join tuner and decoding software – virtual patch cable. jack pilot

Decoding software – dl-fldigi - translates into RTTY

1. Start Jack Pilot. click start.

2. Open gqrx – type in 434.25 and USB modulation

3. Click the IO thing. Input device: FUNcube dongle . Set output device to Jack Router

4. Press Play. You should be hearing raipd, intermittent beeping

5. Zoom in until the grey passband is over the two train tracks

6. Open dl-fldigi. Go back to jackpilot and click on routing

7. click on gqrx on send column. Double click on dl-fldigi in receive. If nothing is printing, align the train tracks to the yellow cursor things.

8. DL-Client – configure. ALL PAYLOADS (TESTING). Select YERRA\_TEST\_11 or latest version

-green flash at the top represents a valid message.

LITERATURE REVIEW

-typical landing height of 100-200 metres seems reasonable

<http://gmaps.geoscience.nsw.gov.au/Statewide/DEM/>

ADVANTAGES OF ACKERMAN’s TECHNIQUE

-able to modify parachute model mid flight using descent data

-

DISADVANTAGES OF ACKERMAN’s TECHNIQUE

PYTHON SYNTAX

print("%3d %0.2f" % (year, principal))

To improve readability, you can continue any statement to the next line by using a backslash (\) at the end of a line as shown. If you do this, the normal indentation rules don’t apply to the next line, so you are free to format the continued lines as you wish.

-yield command

-dir() -> lists all methods

del – deletes

Normally, all of the methods defined within a class apply only to instances of that class (that is, the objects that are created). However, different kinds of methods can be defined such as static methods familiar to C++ and Java programmers. For example: class EventHandler(object): @staticmethod def dispatcherThread(): while (1): # Wait for requests ... EventHandler.dispatcherThread() # Call method like a function

You can write functions into files just like in MATLAB and call them if you have an import statement at the top of your file e.g. div.py -> import div, div.divide(a,b)

alternatively , from div import divide

help(‘modulename’)

print(issubclass.\_\_doc\_\_)

-don’t use tabs ☹

@ -> decorator

s.count(x) Counts occurrences of x in s.

s.index(x [,start [,stop]]) Returns the smallest i where s[i]==x. start and stop optionally specify the starting and ending index for the search. s.insert(i,x) Inserts x at index i. s.pop([i]) Returns the element i and removes it from the list. If i is omitted, the last element is returned. s.remove(x) Searches for x and removes it from s. s.reverse() Reverses items of s in place. s.sort([key [, reverse]]) Sorts items of s in place. key is a key function. reverse is a flag that sorts the list in reverse order. key and reverse should always be specified as keyword arguments.

s.capitalize() Capitalizes the first character. s.center(width [, pad]) Centers the string in a field of length width. pad is a padding character. s.count(sub [,start [,end]]) Counts occurrences of the specified substring sub. s.decode([encoding [,errors]]) Decodes a string and returns a Unicode string (byte strings only). s.encode([encoding [,errors]]) Returns an encoded version of the string (unicode strings only). s.endswith(suffix [,start [,end]]) Checks the end of the string for a suffix. s.expandtabs([tabsize]) Replaces tabs with spaces. s.find(sub [, start [,end]]) Finds the first occurrence of the specified substring sub or returns -1. F h Lib f L B d ff Built-in Types for Representing Data 43 Table 3.5 Continued Method Description s.format(\*args, \*\*kwargs) Formats s. s.index(sub [, start [,end]]) Finds the first occurrence of the specified substring sub or raises an error. s.isalnum() Checks whether all characters are alphanumeric. s.isalpha() Checks whether all characters are alphabetic. s.isdigit() Checks whether all characters are digits. s.islower() Checks whether all characters are lowercase. s.isspace() Checks whether all characters are whitespace. s.istitle() Checks whether the string is a titlecased string (first letter of each word capitalized). s.isupper() Checks whether all characters are uppercase. s.join(t) Joins the strings in sequence t with s as a separator. s.ljust(width [, fill]) Left-aligns s in a string of size width. s.lower() Converts to lowercase. s.lstrip([chrs]) Removes leading whitespace or characters supplied in chrs. s.partition(sep) Partitions a string based on a separator string sep. Returns a tuple (head,sep,tail) or (s, "","") if sep isn’t found. s.replace(old, new [,maxreplace]) Replaces a substring. s.rfind(sub [,start [,end]]) Finds the last occurrence of a substring. s.rindex(sub [,start [,end]]) Finds the last occurrence or raises an error. s.rjust(width [, fill]) Right-aligns s in a string of length width. s.rpartition(sep) Partitions s based on a separator sep, but searches from the end of the string. s.rsplit([sep [,maxsplit]]) Splits a string from the end of the string using sep as a delimiter. maxsplit is the maximum number of splits to perform. If maxsplit is omitted, the result is identical to the split() method. s.rstrip([chrs]) Removes trailing whitespace or characters supplied in chrs. s.split([sep [,maxsplit]]) Splits a string using sep as a delimiter. maxsplit is the maximum number of splits to perform. F h Lib f L B d ff 44 Chapter 3 Types and Objects Table 3.5 Continued s.splitlines([keepends]) Splits a string into a list of lines. If keepends is 1, trailing newlines are preserved. s.startswith(prefix [,start [,end]]) Checks whether a string starts with prefix. s.strip([chrs]) Removes leading and trailing whitespace or characters supplied in chrs. s.swapcase() Converts uppercase to lowercase, and vice versa. s.title() Returns a title-cased version of the string. s.translate(table [,deletechars]) Translates a string using a character translation table table, removing characters in deletechars. s.upper() Converts a string to uppercase. s.zfill(width) Pads a string with zeros on the left up to the specified width.

Read again ~p48-63

For calculations involving heavy use of methods or module lookups, it is almost always better to eliminate the attribute lookup by putting the operation you want to perform into a local variable first. For example, if you were performing a lot of square root operations, it is faster to use 'from math import sqrt' and 'sqrt(x)' rather than typing 'math.sqrt(x)'. In the first part of this section, we saw that this approach resulted in speedup of about 1.4. Obviously you should not try to eliminate attribute lookups everywhere in your program because it will make your code very difficult to read. However, for performance-critical sections, this is a useful technique.

if \_\_name\_\_ == ‘\_\_main\_\_’

The purpose of this command is that this code is only run when you run the file directly, not when you import it.

with open as f

we have access to f after the with block, but the thing is properly closed.