Goal

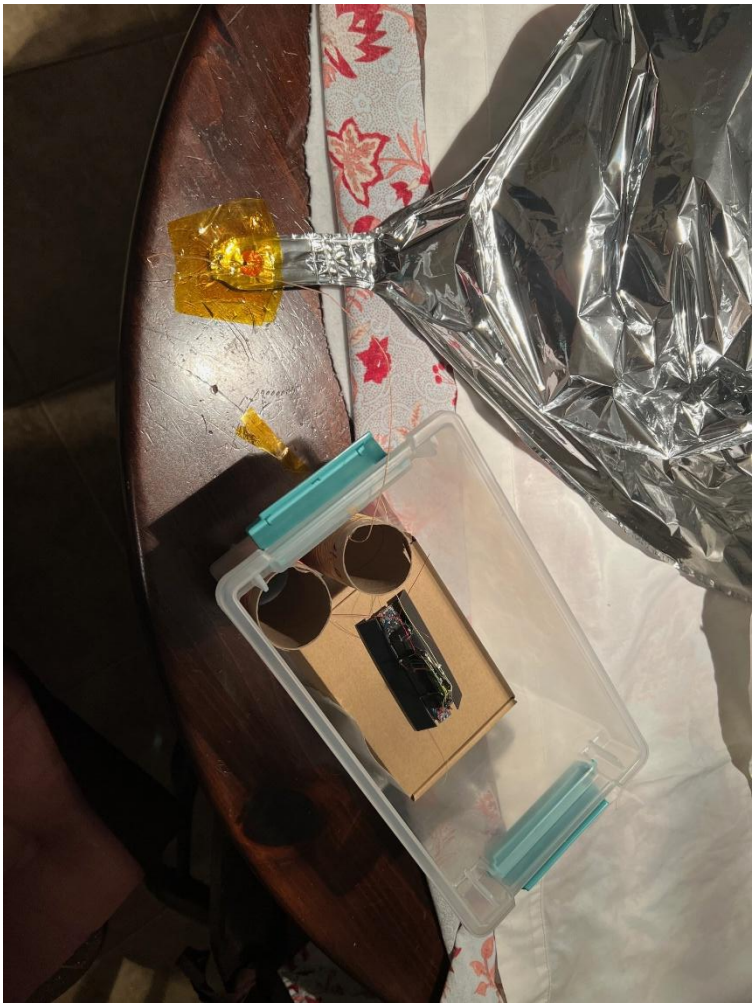
To design a Pico-Balloon that could fly at or above 40,000’ with a payload of 40 – 50 grams, operate 24 hours a day and do Science.

Design

After doing research on the groups.io pico-ballooning site and other mixed media, it became evident that to be successful, it would require a battery that could handle the cold temperatures encountered at high altitudes. After a lot of searching, I found and selected a Tadiran Lithium battery TLI-1550ES. Though not perfect, it provides the best operating temperature range, capacity and mass of any I found. Additionally, I would use a similar platform to what was used for ALP40-G that circled the globe.

ALP40-H

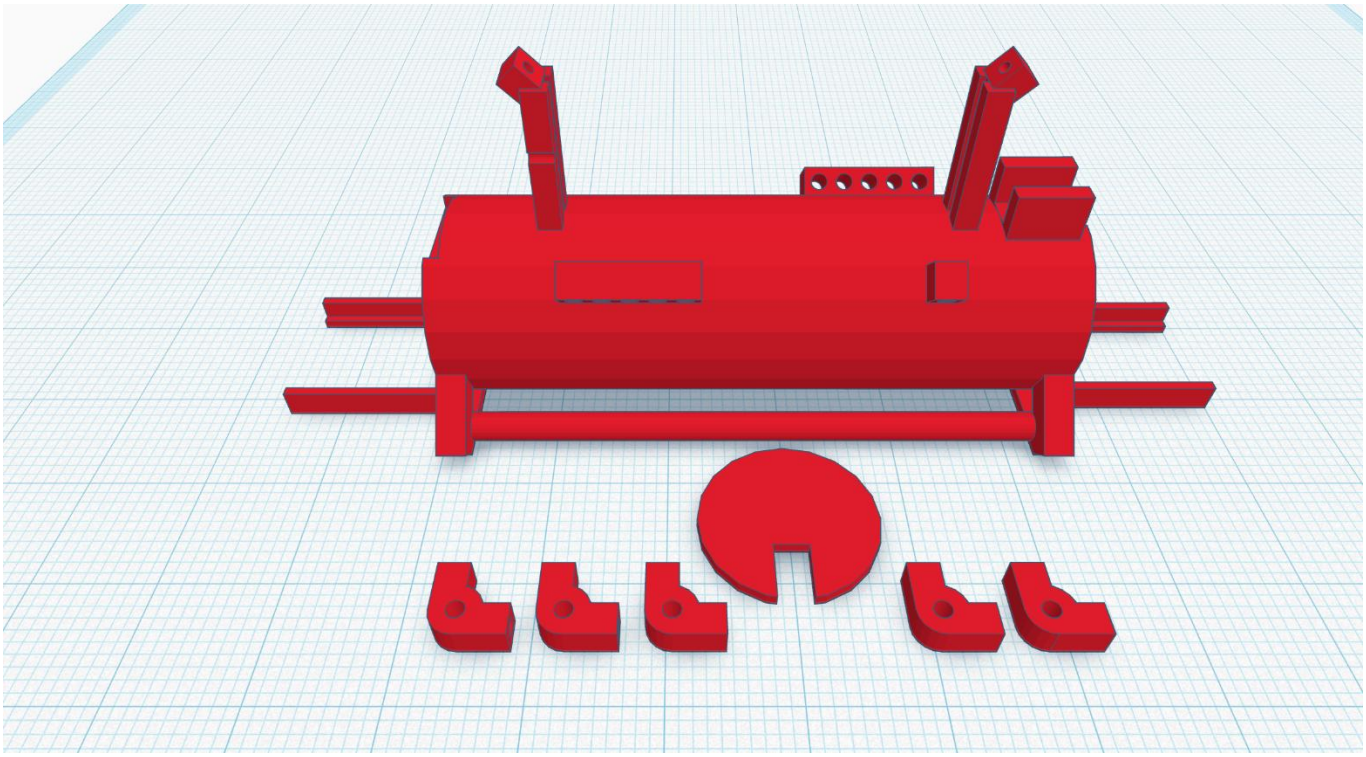
After the overnight success of ALP40-D and the two-week success of ALP40-G, I used the same design, but with a sealed battery compartment; a battery compartment temperature sensor, Innovative Sensor Technology P10K.520.6W.B.010.D; IR temperature sensor, TBP-I2C-H04 (4° field of view); a 3.7 4.2 MPPT Solar controller for a Lithium battery and a higher power solar panel, Anysolar IXOLAR High Efficiency Solar MD SM141K08TF. I flew a Beta version of the U4B firmware, which added extended telemetry and modified I2C command word. The U4B BASIC flight software was modified to transmit alternate IR sensor temperature and battery compartment temperature. A slot 3 WSPR extended telemetry transmission provided IR sensor temperature, battery compartment temperature and solar panel voltage. I used a standard GPS and HF (20 m) dipole antennas. A pre-launch photo is shown below.



A Chinese 60 (Name not diameter) balloon was used and stretched to 116”, the payload had a mass 41.4 grams, with a float of 5 grams and inflated with Hydrogen.

Construction

The process begins with building the frame for all the components. I use a Creality Ender 3v2 3D printer, but any model would work. It is printed using black ABS plastic, which has the desired properties. I use **TinkerCAD** to design the frame and export it as an “*. STL” file found **here**.

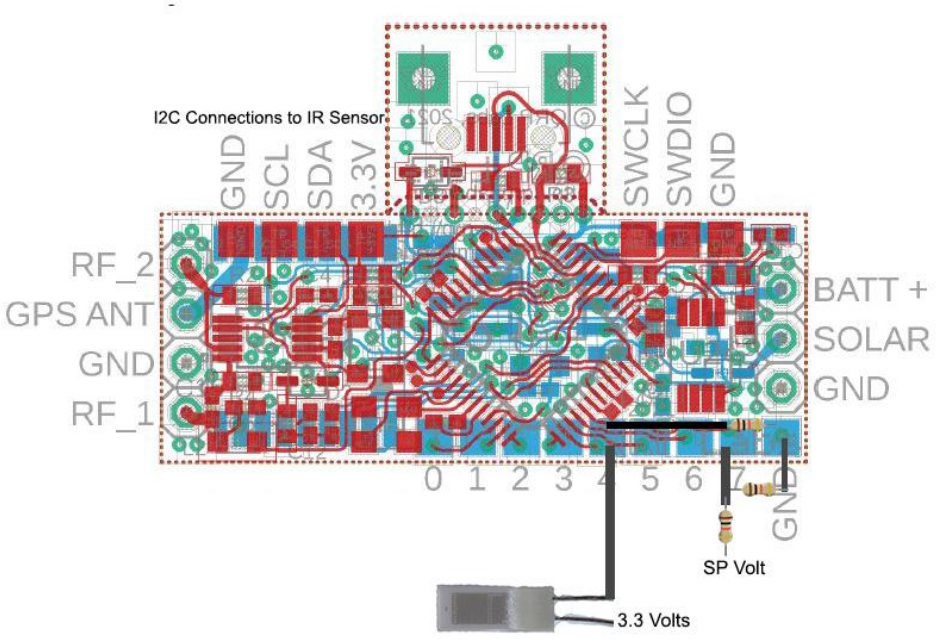


After printing, you get something that looks like this below.

Next, using 1mm **carbon fiber rod**, super glue and **Starbond Accelerator**, I complete frame construction, see below. A small **1mm drill bit** may be needed to open up the holes in the plastic for the CF rods to fit properly.

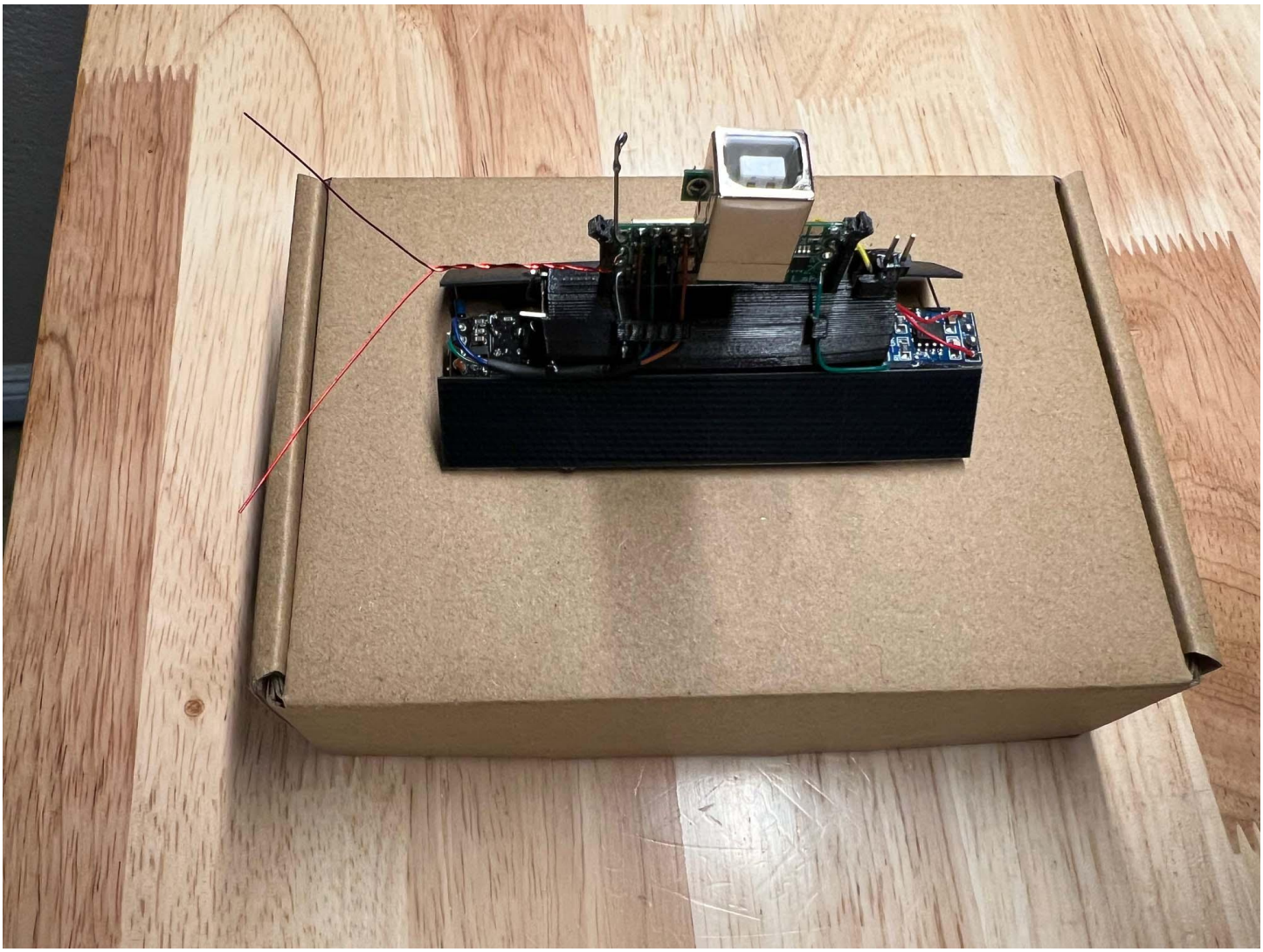
The solar panels will then need to be super glued to the frame and then wired in parallel with Schottky diodes in place on the positive side. SM141K08TF panels are longer and thinner than shown, but you get the idea.

The U4B will then need to be wired with the additional resistor networks for the ADC. These provide the necessary signals to determine the solar panel voltage and read the external temperature sensor. Also, the wires from the I2C bus will need to be connected to the IR sensor. Finally, the solar controller needs to be wired to the solar panels and battery.

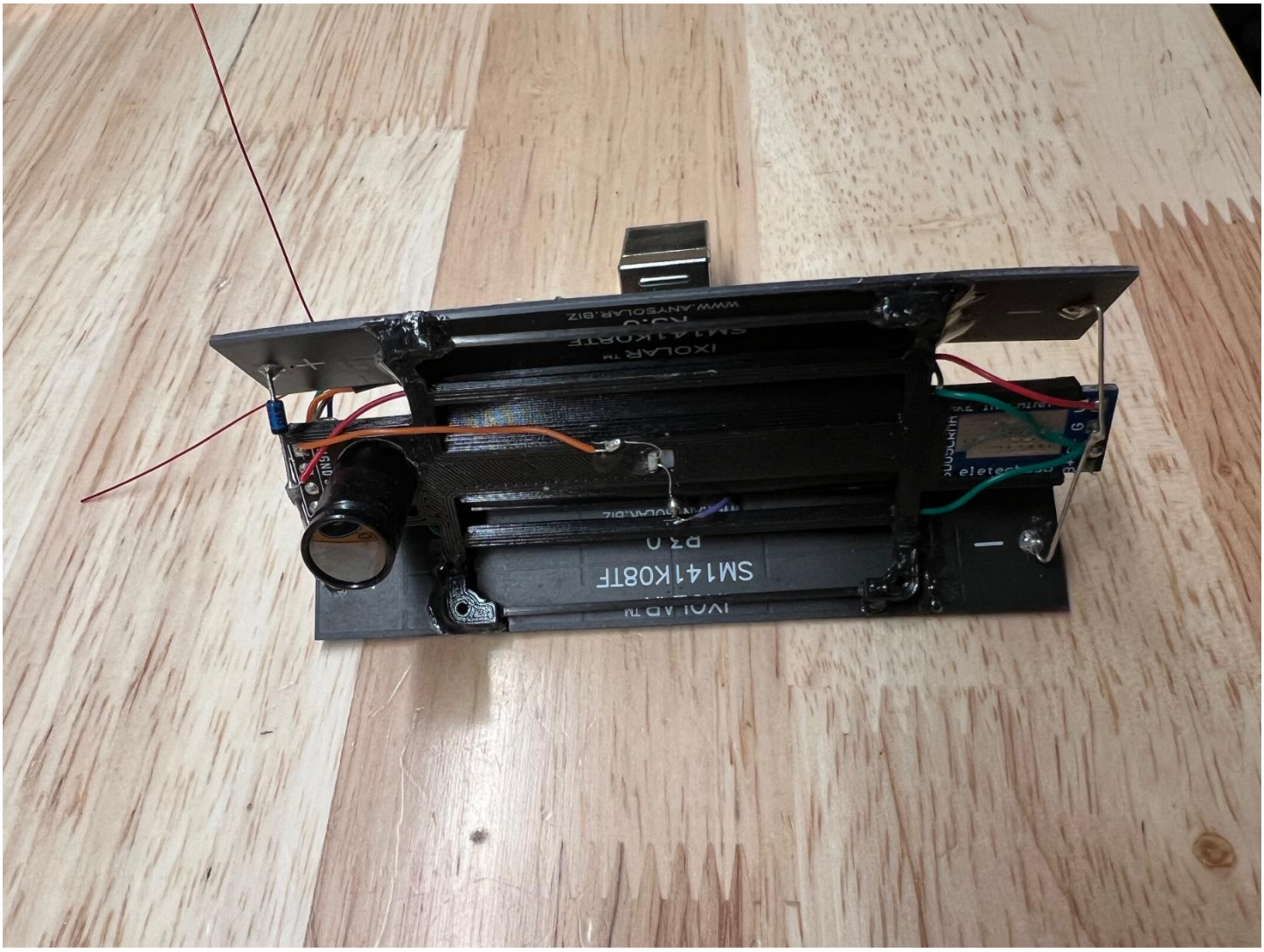


For ALP40-H the external temperature sensor uses ADC channel 4 as shown above, which will be reflected in the software. Since the external temperature sensor is 10K the other resistor in that network is also 10K. Finally, 6.8K resistors were used for the network to read the solar panel voltage ADC channel 7.

The battery is added, and the final wiring is completed, see below.



Top view of the completed payload for ALP40-H, which still has the USB Type B connector attached.



Bottom view of ALP40-H, the IR Sensor is on the left and the 3.7 4.2 MPPT Solar Controller is on the right.

The finished product can be seen at the beginning of the article.

Decode Extended Telemetry

TALARC ALP40-H extended telemetry is decoded from “Big Number” as follows:

As an Example Let Big Number = 384299

Solar Panel Voltage (6 bits)						Battery Temperature (7 bits)							IR Sensor Temperature (7 bits)						
0	1	0	1	1	1	0	1	1	1	0	1	0	0	1	0	1	0	1	1
23						58							43						
384299																			

IR Sensor Temperature (Kelvin)=MOD(BigNumber,128) + 200 = 243K
Battery Temperature (Kelvin) = MOD(INT(BigNumber / 128),128) + 200 = 258K
Solar Panel Voltage (Volts) = INT(INT(BigNumber / 128) / 128) = 23 → Lookup in Table = 3.95 Volts
Note: Temperature range 200K (-73°C or -99°F) to 327K (54°C or 129°F)

Value	Volts	Value	Volts
0	2.80	32	4.40
1	2.85	33	4.45
2	2.90	34	4.50
3	2.95	35	4.55
4	3.00	36	4.60
5	3.05	37	4.65
6	3.10	38	4.70
7	3.15	39	4.75
8	3.20	40	4.80
9	3.25	41	4.85
10	3.30	42	4.90
11	3.35	43	4.95
12	3.40	44	5.00
13	3.45	45	5.05
14	3.50	46	5.10
15	3.55	47	5.15
16	3.60	48	5.20
17	3.65	49	5.25
18	3.70	50	5.30
19	3.75	51	5.35
20	3.80	52	5.40
21	3.85	53	5.45
22	3.90	54	5.50
23	3.95	55	5.55
24	4.00	56	5.60
25	4.05	57	5.65
26	4.10	58	5.70
27	4.15	59	5.75
28	4.20	60	5.80
29	4.25	61	5.85
30	4.30	62	5.90
31	4.35	63	5.95

Software

Standard QRP Labs U4B BASIC programming was used. Program follows use at your own risk.

TALARC Balloon AP40-G Flight Code

Configuration

KB7HTA-15 – 20 Meters – CH 031 – WSPR 14.097140 – CW 14.022000 – Start Minute 00/02 – 1st & 3rd 0-1

Variables

Letter	Initial	Description
S	1	Battery Set Point S = 6 (> 3.9 V), S = 3 (3.7 – 3.9 V), S = 2 (3.4 – 3.7 V) S = 1 (< 3.4 V)
H	3900	This is the battery “High” set point in millivolts.
L	3700	This is the battery “Low” set point in millivolts.
V	3400	This is the battery “Very Low” set point in millivolts.
O	273	This holds the battery compartment temperature in Kelvin after the “BCT” subroutine is RUN

F	0	If this flag is true (1), then the next TELE sends the external temperature vs PCB temperature.
M	4300	Below this altitude in meters, send a CW report with latitude, longitude and battery voltage
Y	0	This variable holds the result of reading the solar panel voltage after the diodes but before the Lithium battery charging circuit. There is a 6.8K Ohm divider network to cut the voltage in half to keep the voltage within the voltage range of the ADC. The full scale 12- bit ADC reading would be 4096 representing 3.3 Volts. The software multiplies this number by 2 to restore the full range of voltages 0 – 6.6, lost due to the voltage divider.
P	0	This holds the 29.180 Bit Big Number
Z	0	
A	?	Temporary Variable
B	?	Temporary Variable
C	?	Temporary Variable
D	?	Temporary Variable

```
LET HP = 0
LET S = 1
LET P = 0
LET Z = 0
LET H = 3900
LET L = 3700
LET V = 3400
LET O = 273
LET F = 0
LET M = 4300
GPS 300
RUN “QLOCK”
10 RUN “BSP”
RUN “QSND”
GPS 300
OUT 19 0
SLEEP 10 9
OUT 19 1
RUN “QBCT”
RUN “PACK”
TELE
TELEX 0 P
RUN “QCW”
RUN “QDWN”
GOTO 10
END
~
```

BSP – Determine the Battery Set Point S = 6 (> 4.0 V), S = 3 (3.7 – 4.0 V), S = 2 (3.4 – 3.7 V) S = 1 (< 3.4 V). The variable S specifies the number of times a WSPR transmission is made each hour.

```
LET S = 1
```

```
IF BT > V
   LET S = 2
ENDIF
IF BT > L
   LET S = 3
ENDIF
IF BT > H
   LET S = 6
ENDIF
~
```

QSND – Use the Battery Set Point to determine how often to send WSPR transmissions by introducing CPU sleep cycles.

```
IF S = 1
   LET A = 5
   LET HP = 0
   RUN “SLPX”
ENDIF
IF S = 2
   LET A = 2
   LET HP = 1
   RUN “SLPX”
ENDIF
IF S = 3
   LET A = 1
   LET HP = 1
   RUN “SLPX”
ENDIF
IF S = 6
   LET HP = 1
ENDIF
~
```

SLPX – Sleep for the number of transmission cycle(s) determined by the variable A.

```
OUT 19 0
FOR X = 1 TO A
   SLEEP 10 9
NEXT X
OUT 19 1
~
```

QBCT – Determine If **B**attery **C**ompartment **T**emperature or IR Sensor Temperature Needs to be Read and Substituted for U4B Temperature. A value of 2 K was added to ‘O’ and ‘Z’ the value stored in the system variable TT to take care of round error between the encoding done by PACK and the encoding done on system variable.

```
IF F > 0
```

```
   RUN “BCT”
   LET TT = O + 2
   LET F = 0
ELSE
   RUN “RIRT”
   LET TT = Z + 2
   LET F = 1
ENDIF
~
```

BCT – Read and Convert the **B**attery **C**ompartment **T**emperature to Kelvin. The resulting temperatures seem to be 7 K higher than they should be, so 7 is subtracted from the value before leaving the subroutine

```
LET A = INA 4
DELAY 1
LET A = INA 4
LET B = A * 806
LET C = 3300000 – B
LET D = C / ( B / 10000 )
LET O = ( D / 38 ) + ( D / 39 )
LET O = ( O / 2 ) - 7
~
```

QCW – This program is run to determine, depending on battery voltage, whether to send a CW transmission.

```
IF S = 6
CW 0 14022000 12 0 “CQ KB7HTA KB7HTA BALLOON #M6 #AT #BT”
ENDIF
~
```

FLASH – This program is used to send the number of light flashes specified by the variable A.

```
FOR X = 1 TO A
OUT 19 0
DELAY 1000
OUT 19 1
DELAY 1000
NEXT
~
```

QLOCK – This program determines if the balloon has a GPS lock. If there is a lock it turns off the LED, otherwise it flashes the LED 10 times and leaves it lit.

```
IF GL = 0
LET A = 10
RUN “FLASH”
OUT 19 1
ELSE
OUT 19 0
```

```
ENDIF
~
```

QDWN – This program determines if the balloon is possibly going down. It sends CW Latitude Longitude and battery voltage if the balloon goes below the value of variable M, which is usually 4300 meters or 14,000'.

```
IF AT < M
CW 0 146585000 12 0 "KB7HTA #LT #LN #LT #LN #BT"
ENDIF
~
```

RIRT – Read the IR Temperature Sensor

```
LET Z = 0
FOR X = 1 TO 5
   LET D = I2CR 117 7 3
   LET D = D % 65536
   LET Z = Z + D
NEXT
LET Z = 2 * Z / 500
~
```

RSPV – Read the solar panel voltage after the diodes.

```
FOR X = 1 TO 5
LET Y = Y + INA 7
DELAY 10
NEXT
LET A = 0
LET Y = Y / 5
LET Y = Y * 1000 / 4096
LET Y = Y * 3300 / 10000
LET Y = Y * 2
IF Y > 280
   LET Y = (Y – 280) / 5
ELSE
   LET Y = 0
ENDIF
~
```

PACK – Package telemetry, IR sensor temperature (K), battery compartment temperature (K) and solar panel voltage index (0 – 63) representing 2.80 to 5.95 Volts with a resolution of .05 Volts.

```
LET P = 0
RUN "RSPV"
LET P = ( P + Y ) * 128
RUN "BCT"
LET P = ( P + ( O – 200 ) ) * 128
RUN "RIRT"
```

LET P = P + (Z – 200)
~

Balloon Mass Analysis

3D ABS Print:	3.66 grams	
Solar Battery Charger:	0.79 grams	
IR Temperature Sensor:	2.36 grams	
U4B Tracker:	1.80 grams	
Solar Panels x 2:	6.03 grams	
Battery:	21.70 grams	
Miscellaneous:	2.36 grams	
Payload Mass:	38.75 grams	
Dipole Antenna Top:	1.60 grams	--- #34 gauge copper wire 16’ 7.2” with 6# fishing line and silicone
Dipole Antenna Bot:	1.00 grams	--- #34 gauge copper wire 16’ 7.2”
Total Payload:	41.35 grams	
Float:	5.00 grams	
Neck Weight:	46.35 grams	