GitHub - EngineerGuy314/pico-WSPRer: minimalist WSPR tracker for pico-balloons utilizing Raspberry Pi Pico as the RF generator. Uses U4B/Traquito protocol for 6 char grid, voltage and temperature from the RP2040's internal sensor.

https://github.com/EngineerGuy314/pico-WSPRer

The README

pico balloon WSPR tracker

Implements an extremely low-cost WSPR tracker for HAB (high altitude balloons) with barely anything besides a Raspberry Pi Pico. The only other significant component is a cheap GPS module such as a generic ATGM336H as shown (approx $4 as of April 2024). RF power in the 14Mhz (20 meter) band is generated directly by the Pi Pico. Two gpio pins are driven out of phase to feed a dipole antenna (typ 38awg copper).

The RF synthesis and basic WSPR code is based on Roman Piksaykin's excellent work at https://github.com/RPiks/pico-WSPR-tx

This WSPR beacon calculates the altitude and full 6 character maidenhead grid based on the GPS location data and transmits it along with the callsign. Solar array voltage and rp2040 temperature are also sent as telemetry utilizing the U4B/Traquito protocol.

The user's callsign and telemetry encoding details are configurable via the pico's USB port and a simple terminal program (ie Putty).

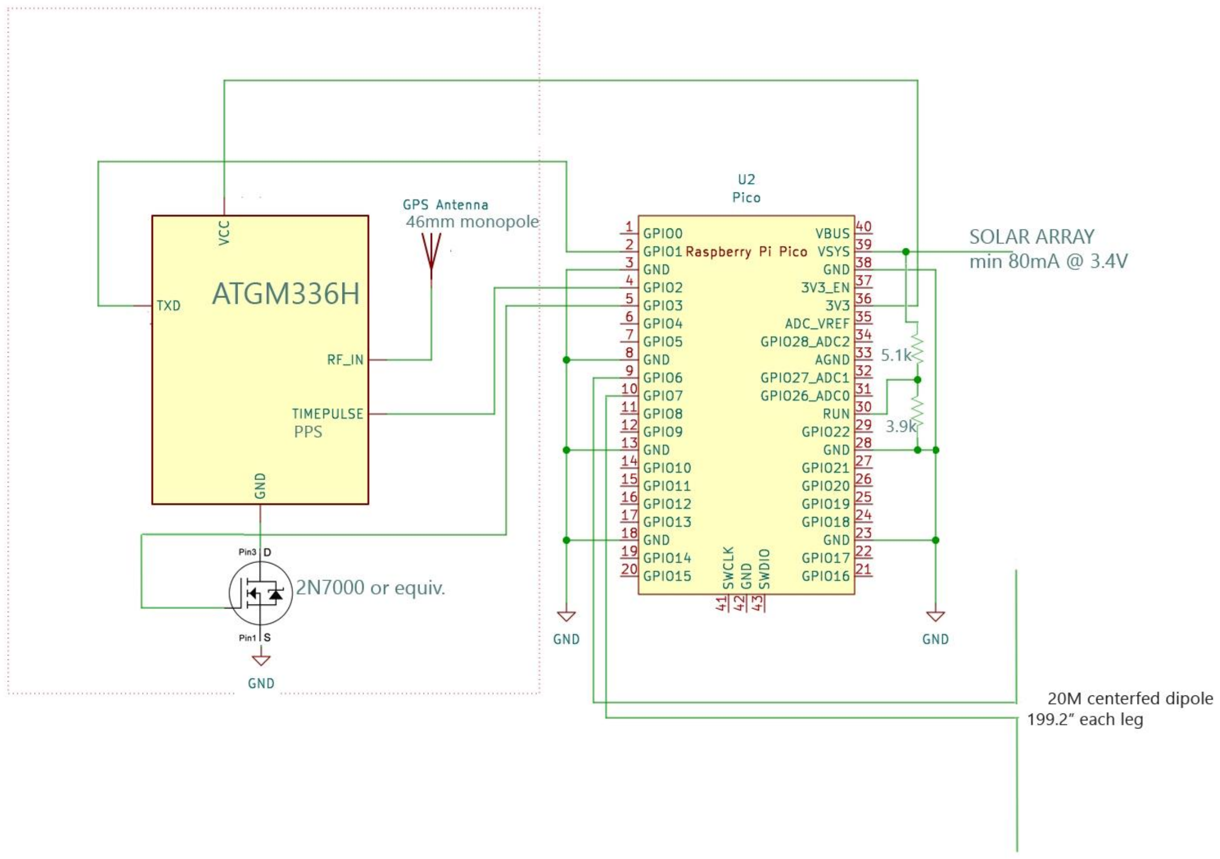
There is an issue with the RP2040 locking up if its input voltage is raised too gradually (as would happen during sunrise). To combat this I have a simple voltage dividor of two resistors across ground and the input voltage. The output if this voltage divider is tied to the RUN input on Pi Pico. The GPS unit stays on during transmission to provide continual frequency shift correction. However, a generic MOSFET or NPN transistor must be included to allow the pico to keep the GPS powered off during initial sunrise boot. See schematic below.

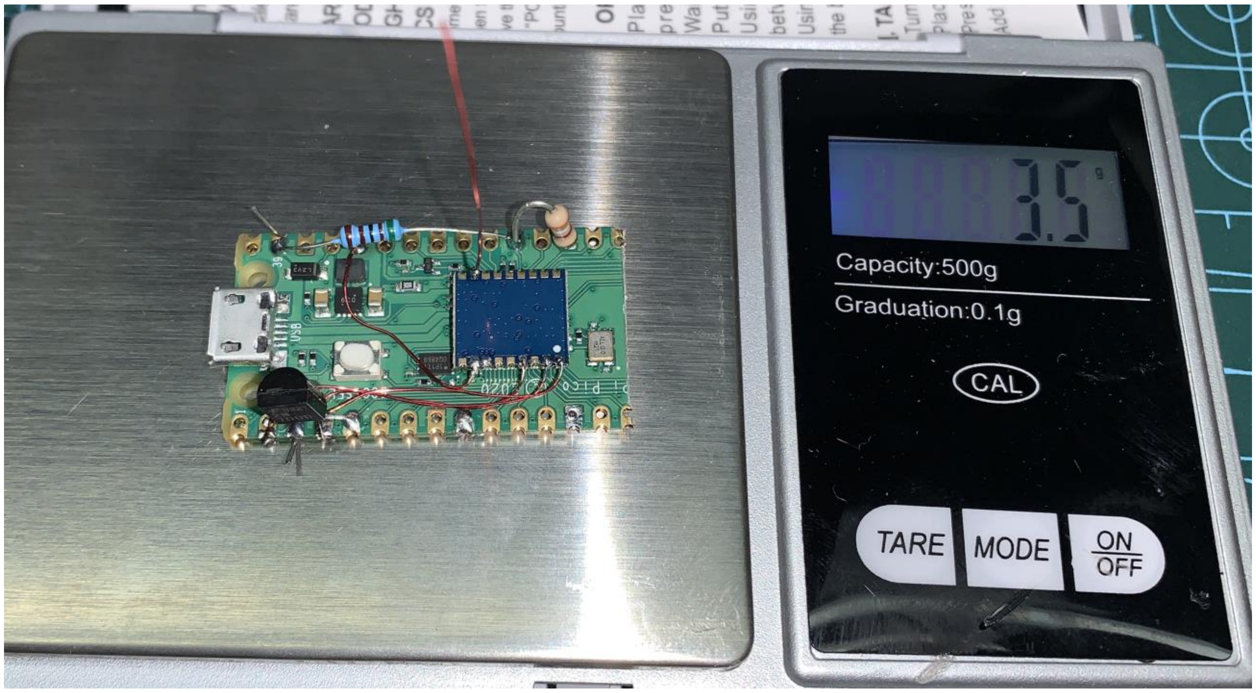
With the original code the Pico was being overclocked to 270Mhz, so the total power draw of the Pico and GPS module was around 100mA at 4 volts. But this version I have the speed down to 135Mhz, which is fine for transmitting on 20M (14Mhz).

Quick-start

1. Download https://github.com/EngineerGuy314/pico-WSPRer/blob/main/build/pico-WSPRer.uf2 and skip to step 6 (or follow steps 1-5 to compile it yourself)

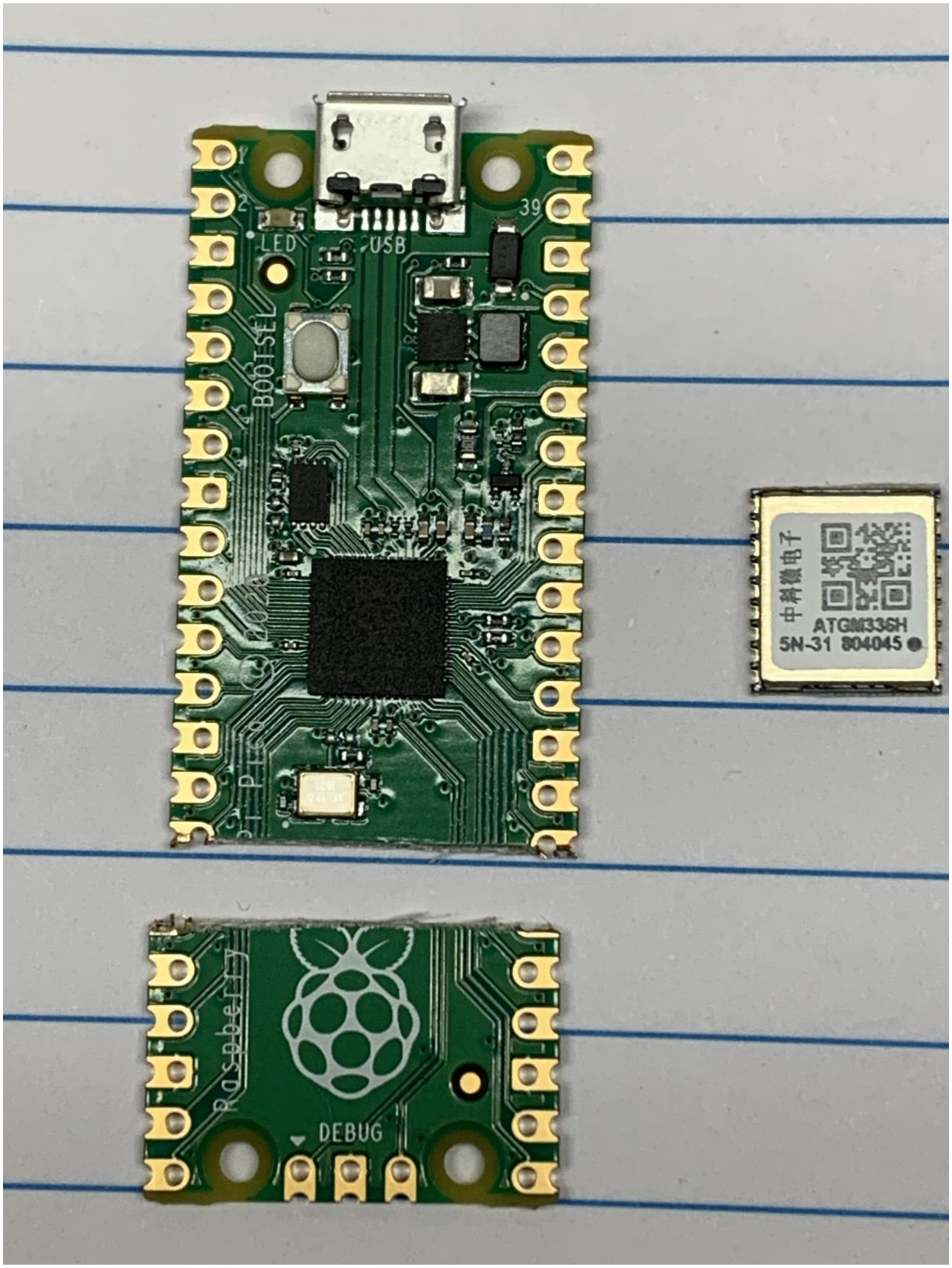
2. Install Raspberry Pi Pico SDK. Configure environment variables. Test whether it is built successfully.
3. git clone https://github.com/EngineerGuy314/pico-WSPRer
4. cd pico-WSPRer5.
5. ./build.sh
6. Check whether output file ./build/pico-WSPRer.uf2 appears.
7. power up pico with BOOTSEL held, copy the .uf2 file into the Pico when it shows up as a jumpdrive.
8. Go to https://traquito.github.io/channelmap/ to find an open channel and make note of id13 (column header), minute and lane (frequency).
9. Connect to pico with a USB cable and a terminal program such as Putty. Hit any key to access setup menu. Configure your callsign and telemetry channel details from step 7.
10. WSPR type-1 messages will be sent every ten minutes followed by the U4B style telemetry with a coded callsign
11. If the pico is plugged into a computer via USB while running it will appear as a COM port and diagnostic messages can be viewed at 115200 baud. Raw $GNRMC messages from GPS unit will be displayed, and every 20 seconds decoded latitude/longitude and grid locator will be shown.



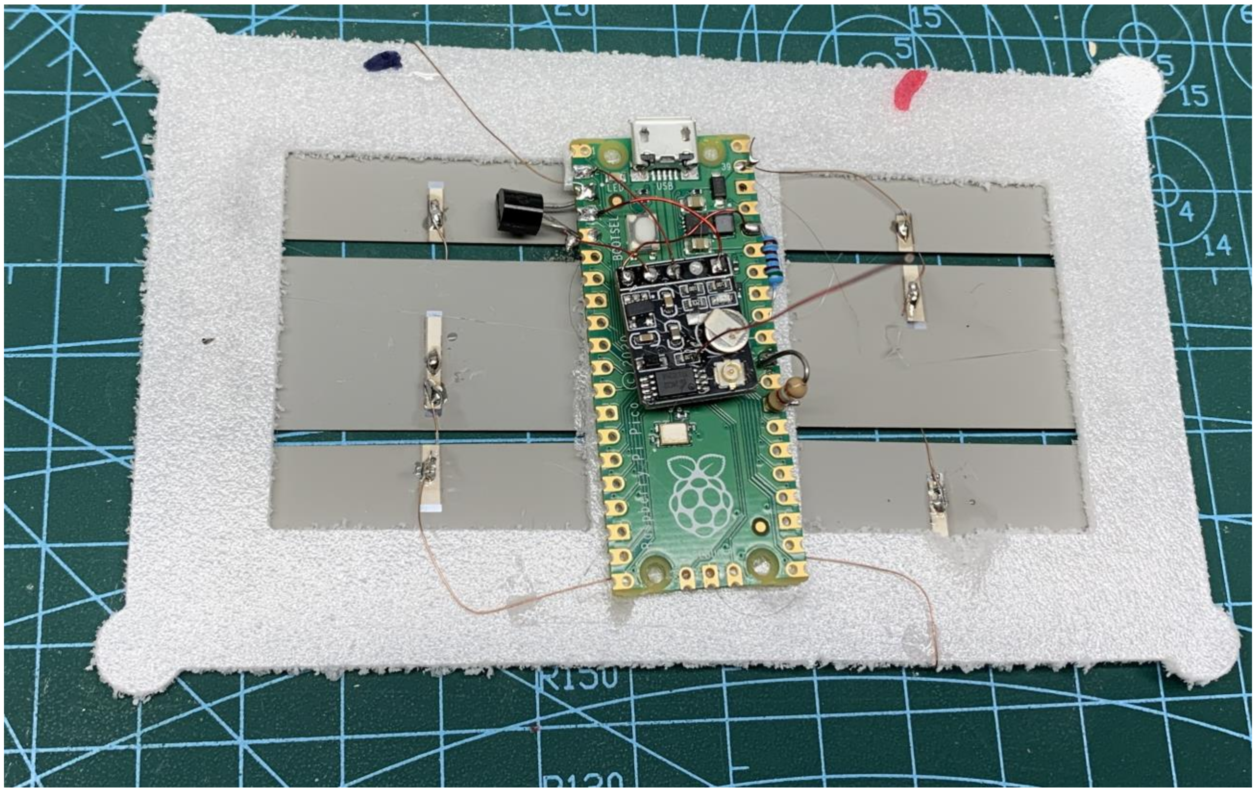


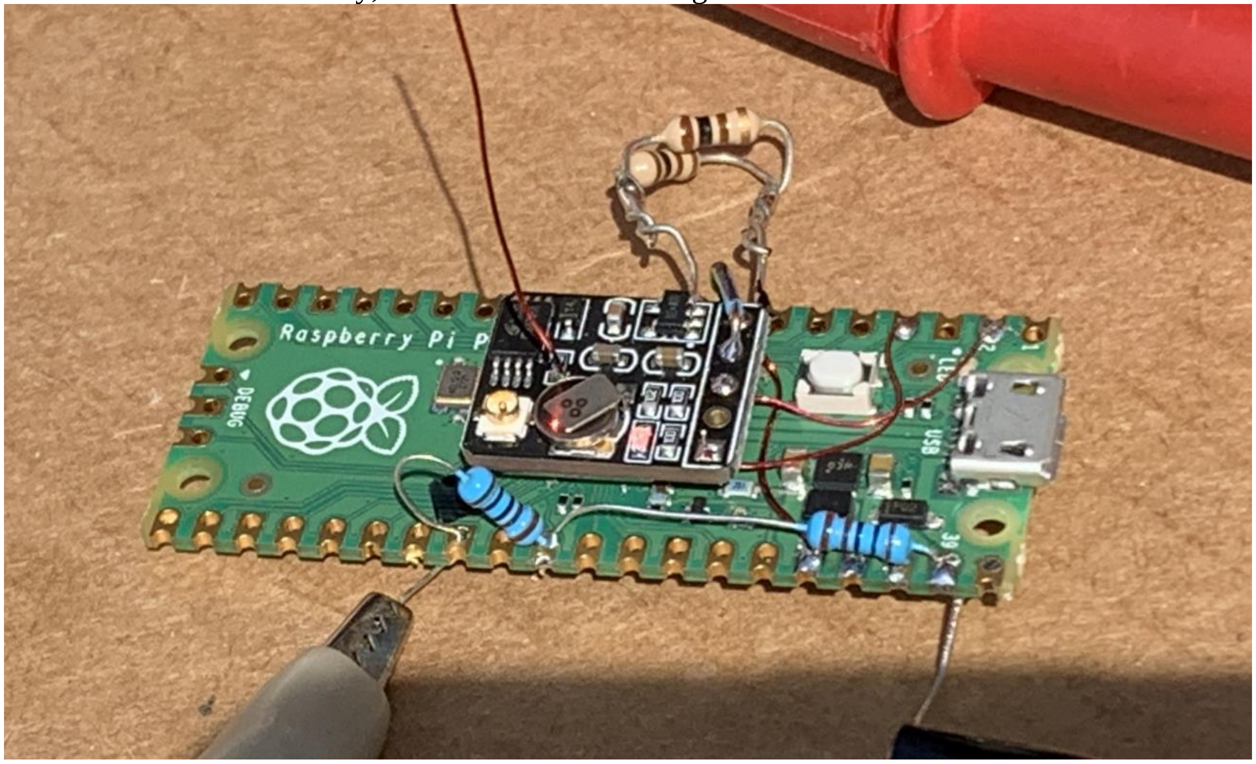
Completed V3 unit before flight. Bare castelated ATGM336H is flipped over and glued to the RP2040. Total weight including GPS antenna 3.5g.

Pico board can be trimmed to save weight

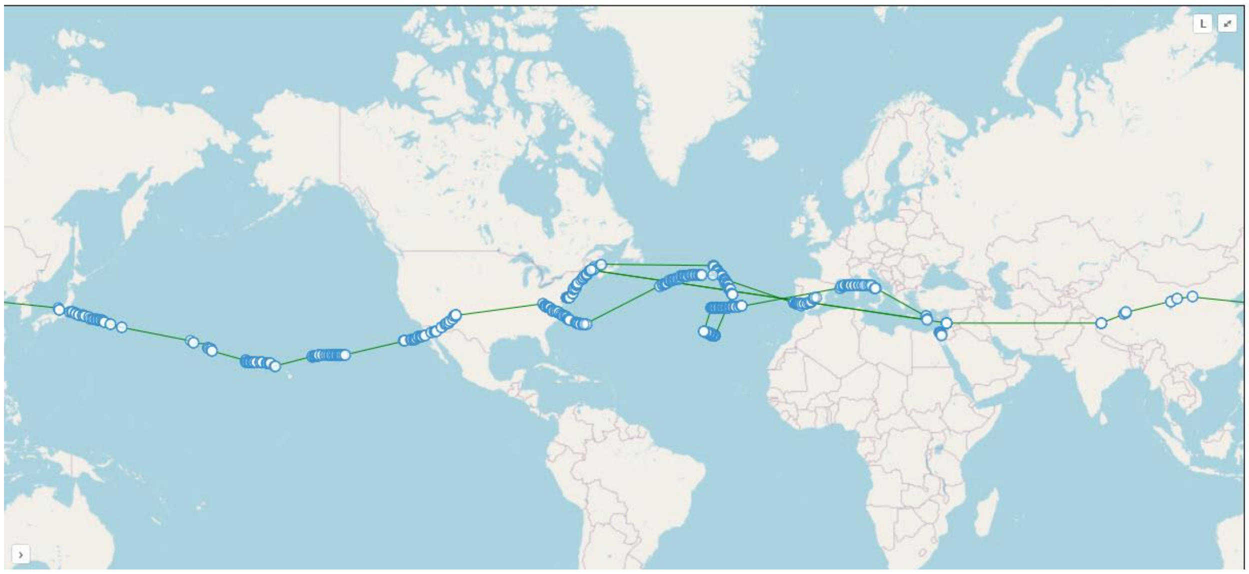


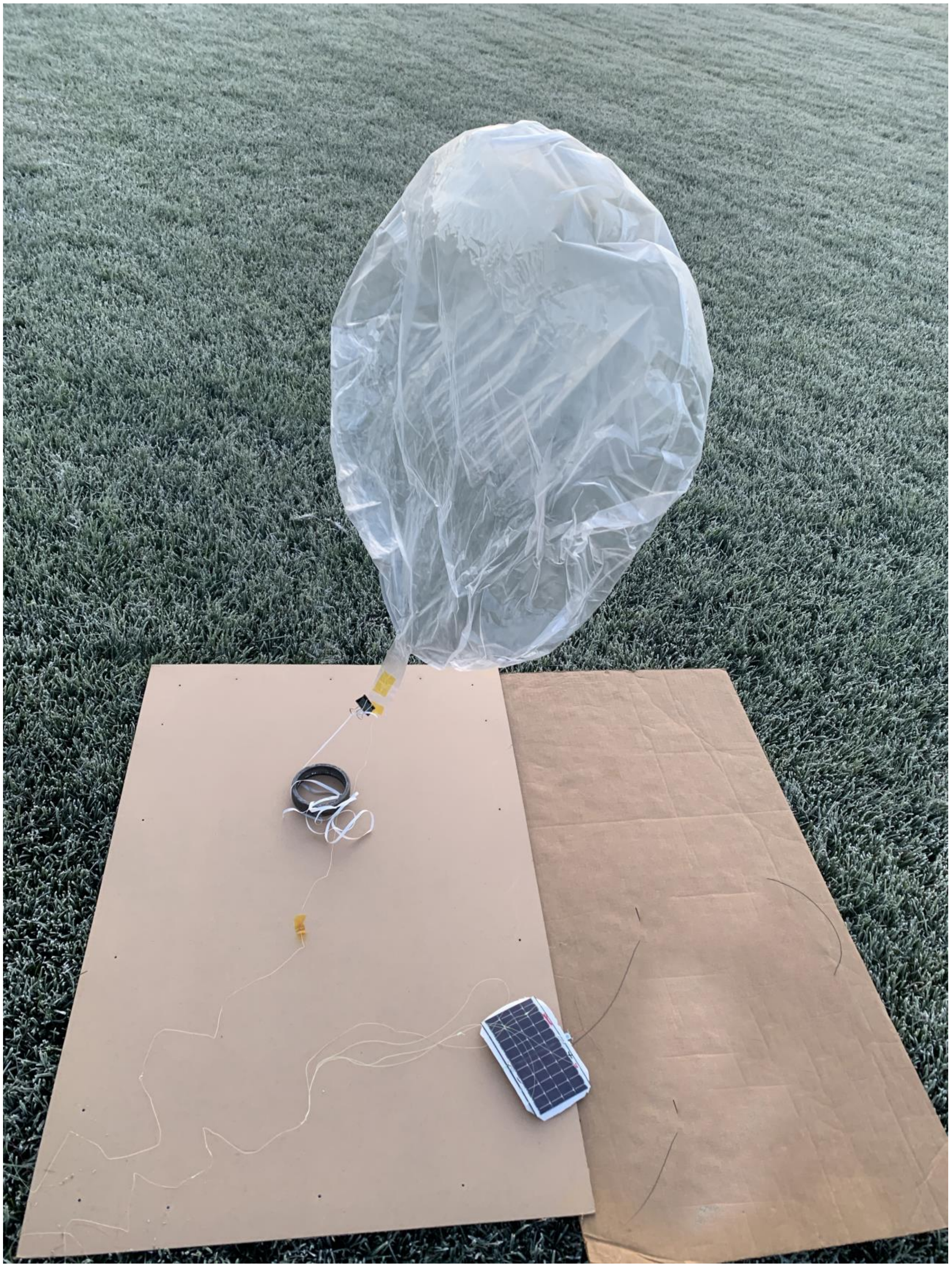
Pico board can be trimmed to save weight


Mounted beneath solar array, before first succesful flight.

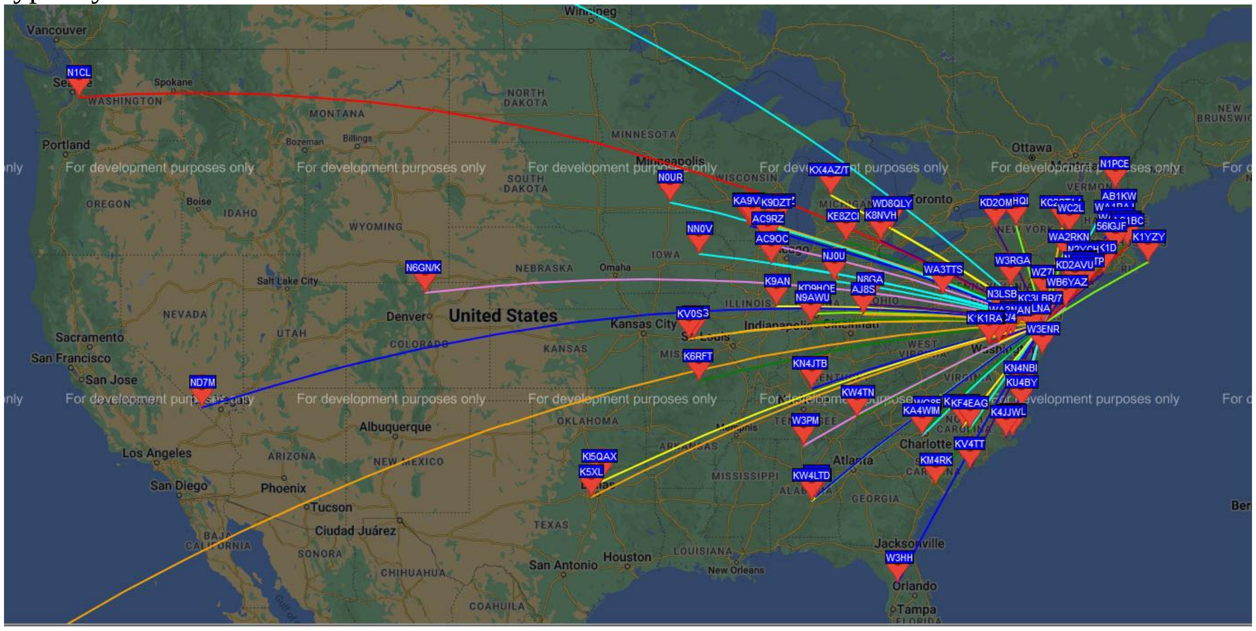


V1 prototype


V2 was launched and circumnavigated the world in 13 days.


Typical solar powered tracker and balloon before launch. Balloon is deliberately under-

filled with hydrogen to allow room for expansion at high altitude. 12-13km altitude is typicaly reached.



Global tracking with such low powered transmissions are possible thanks to the extensive network of WSPR receive stations.

DISCLAIMER: This project transmits on the 20 meter ham radio band: you must have an appropriate amatuer radio license to do so legally. Also, toggling a microcontroller IO pin theoretically generates a square wave, which theoretically has an infinite number of high order harmonics. However, RP2040 io pin circuitry is not particularly efficient at generating very high RF energy. Furthermore, the dipole antenna is trimmed to be resonant at 14.1Mhz and does not do a very good job at radiating anything else. During my testing the amount of RF energy actually emitted outside of the passband was well within limits. You are encouraged to perform your own testing and utilize additionall filtering as needed to meet your local regulations.