# ZenAI Community of Robots ch7

**www.ZENMCU.com**

Draft 1 2023-10-18

Copyright © 2023 by Trenton Henry, All rights reserved

**zenai-0.0a #7**
As always, I have had numerous non-hobby related activities to attend to so I have not made great strides forward recently.

If you followed along with chapter 6, the round PCB rykor1 board, you are aware that I made several physical mistakes that hampered my ability to bring up the robot.

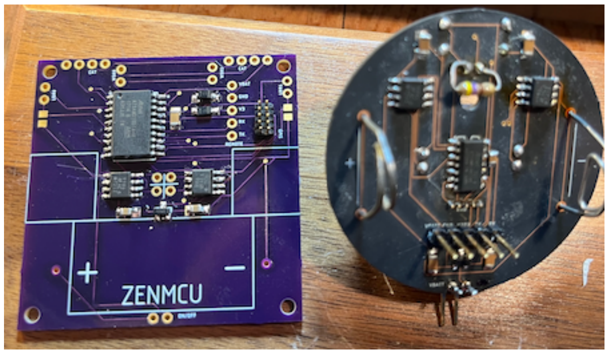
- The board radius was a tiny bit to large to fit the tube I had planned for it, which was needed to make room for the battery and hold up the dome.
- The SWD pins were pretty much right under the motors making it difficult to connect the debugger, necessitating inverting the robot, etc.
- I used 2.75 mm headers for the SWD connector and the mating connectors just don't have enough grip to stay connected while holding/manipulating the robot.
- Etc.

I did manage to get some mileage from that design, however, including some LDR experiments, and differential drive speed matching. None of it is perfect, but its progress.

This time I decided to do something a little less constrained. I decided on a 2x2 inch chassis / PCB with enough space to comfortably mount everything comfortably. This board has a SAMD10D (20 pin) part, UART host interface (so as to be ready to mount a Kaldane), and is powered by a battery.

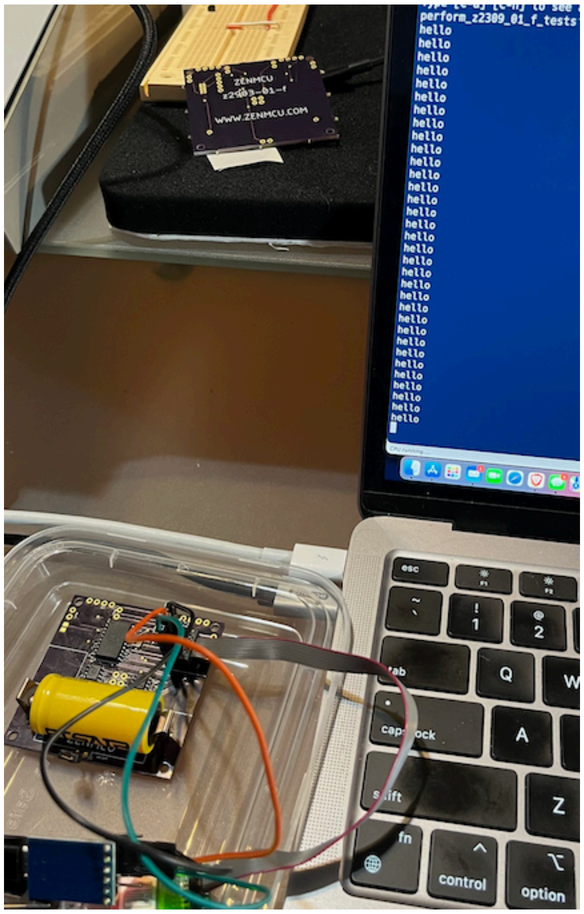
- 2 LDRS for light sensing.
- 2 RGB LEDs for emitting color, sensing color, and possibly transmitting and receiving data.
- 2 Whisker/switch inputs (on SWC/SWD so not useable while debugging).
- There is a 5 pin HIF with VBATT, GND, V3P3, RX, TX to facilitate development by connecting it to a PC using a CP2102 with picocom, and later to support a Kaldane as the host.
- There are two DC gear motors from Didel.
- The power supply is a 16340 Battery (2800 mAh Li-ion 3.7 Rechargeable CR123A Cell).
- The on/off switch is just a jumper for now.

## In reality it isn't that much larger, but its a square, so it has more surface area.

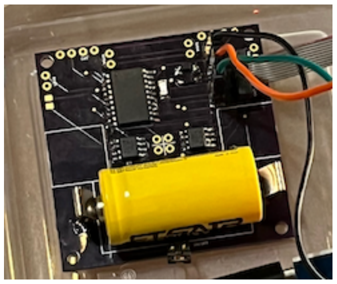


## Bringup

First I soldered the LDOs, resistors, capacitors.
Then I soldered the ICs, the SWD header, and the HIF header.
Then I soldered the battery connectors.
Then I soldered the LDRs and the RGB LEDs.

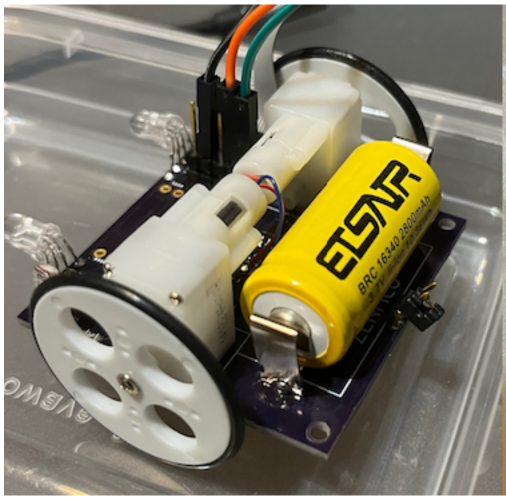
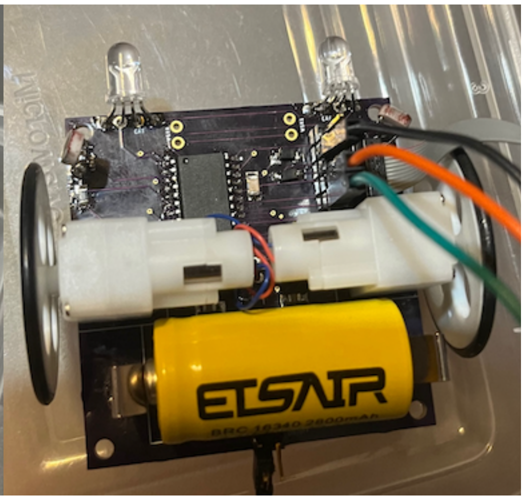


```
perform_z2309_01_f_tests
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
```



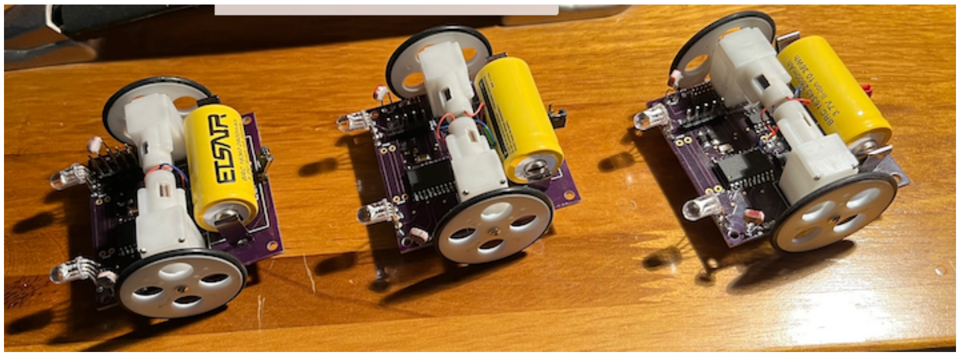
Then I wrote some test code to print over UART, blink LEDs, and read LDRs.
Everything works fine, although there are (as usual) some issues.

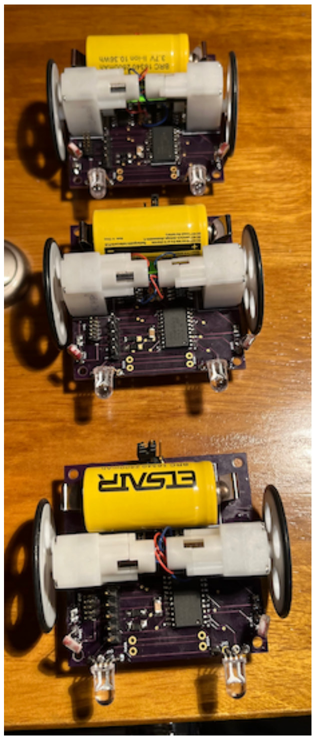
- The holes for the battery spring leaves are too small making it a lot of work to install them.
- I chose the wrong resistance for the LDR voltage dividers and ended up having to set the gain really high on the ADC to get useable readings. My LDRs are like 30 years or more old, and I have no idea what their dark resistance is. I did try several times to measure them, but my multimeters are also old, not high quality, and not terribly trustworthy. Also they all seem to use stupid hard to find batteries.
- So I bought a new digital meter, and a new analog meter. They aren't terribly expensive, but they use normal batteries. Hopefully they are of sufficient quality as to be reliable. I also ordered a bunch of 200K ohm (dark) CdS LDRs and a big variety pack of 0805 resistors, including of course 200K ohm. When those arrive the boards will get an upgrade.
- Aaand ... they arrived and the 200K LDRs with 200K resistors didn't behave as expected. The seem to be like 50K? So I did some experiments swapping out the fixed and settled on 2K which seem to work pretty well.
- Also, I managed to locate the SWD connector in a difficult to access area, again. It is quite close to one of the motors, as well as one of the LDRs. But it has just enough room. And I was at least smart enough to use the 10 pin header so it stays on solidly. The cable, of course, rubs the wheel so it has to be held clear.
- I also put the host interface connector in a kind of random spot. Its ok for now, but it probably won't stack well with a PCB on top. I probably should have used a QUIIC style connector (my QUUIC UART variant) with another connector for VBATT. But I think its ok for now.

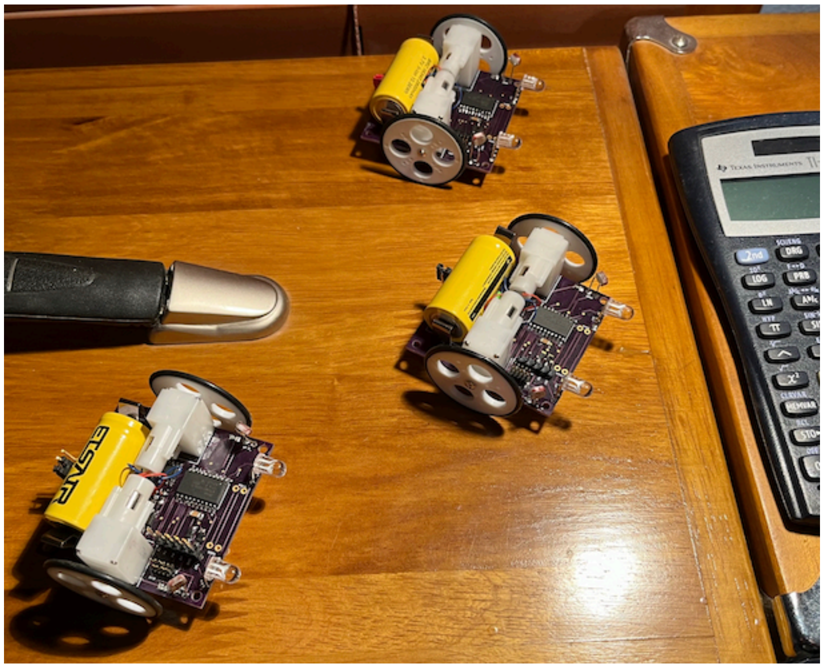



Things I still need to do include but aren't limited to:

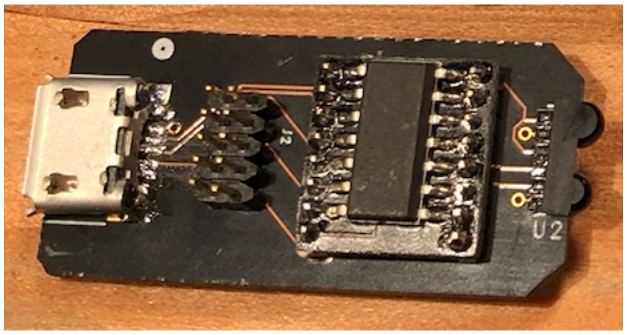
- Mount the motors and get the differential drive code smoothed out. Done 10/11/23.
- Get the light sensing using RGB LED (common cathode) with ADC code in place. Done 10/22/23.
- Build up all three boards. Done 10/22/23, except only one board has motors so far.
- Add motors to boards 2 & 3. Done 10/23/23.
- Lots of experiments. This is likely to involve dumping a lot of data over the UART. It may also involve capturing 'watch' data to view in zplot.



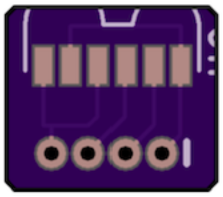




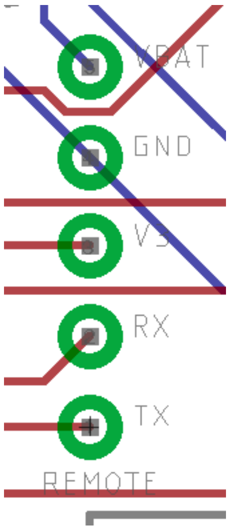
**10/23/23**
Once the robots have motors and are able to move around the wired host interface, and obviously the SWD, become major impediments. It is getting close to time to mount up some I/R cards and break out the Ociter.



I have qty (21) brd0026 that are intended for exactly this purpose.



4:GND, 3:RX, 2:TX, 1:VCC



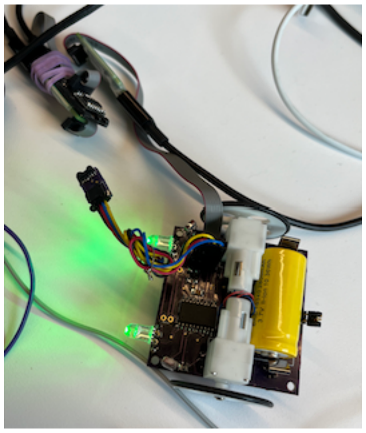
I wasn't smart enough to make the Rykor's HIF connector match up to the I/R board, but I can make this work.

**11/03/23**
I failed to get a large enough block of time to sit down and make this work, so I had to do it in little sessions over many days. But regardless, it now seems to work.  The Rykor is now printing its results to the host exactly as before, but now it is using the I/R link.

Before: [rykor-uart] <--> [cp2102-usb2uart] <--> [picom-host]

After:   [rykor-uart] <--> [i/r-xcvr] <--> [i/r-xcve] <--> [ociter] <--usb--> [ociter-host]

Its kind of tricky because knowing if its actually transmitting or receiving requires a scope to watch the i/r transceiver pads wiggle, which is hard to do with only two hands.

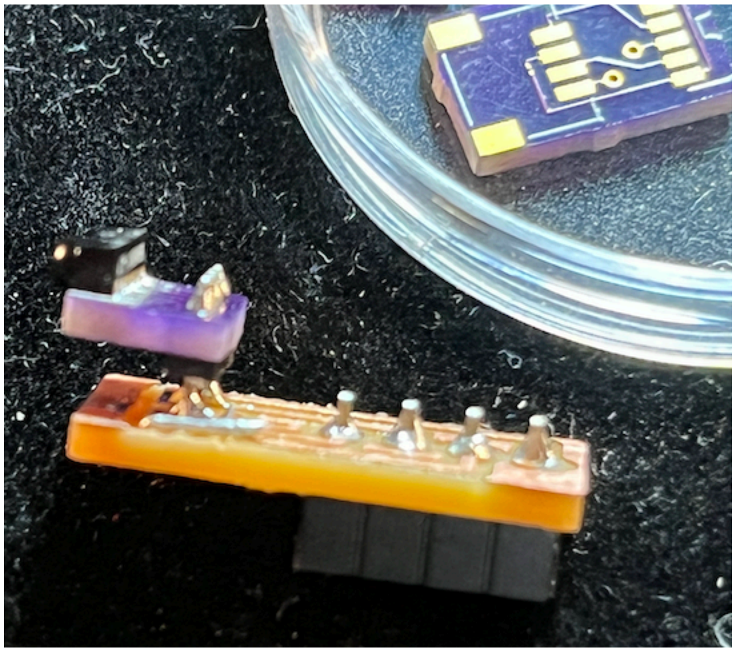


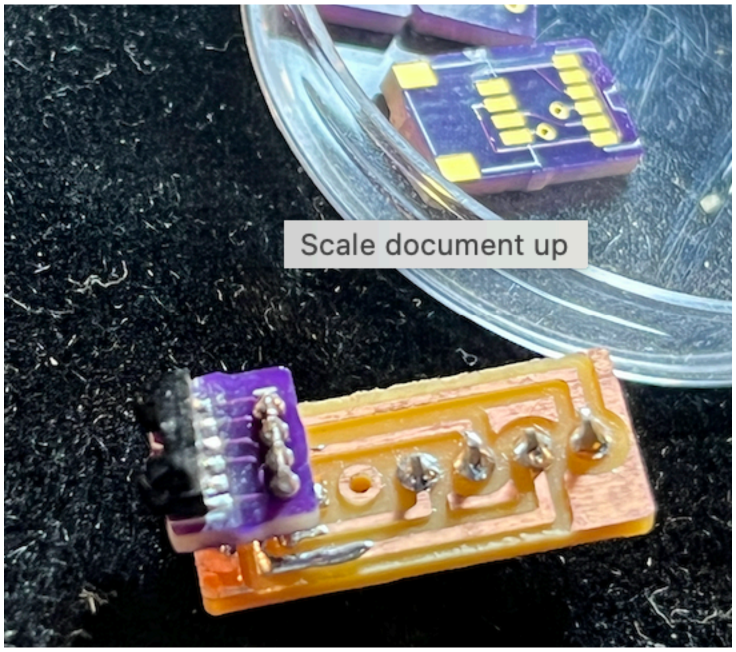
You can see in the photo that the i/r is attached to the Rykor using one of my brd0036 quuc-ir-dapta boards. I actually had those made like a year ago and never got back to using them until now. It was convenient to have them on hand.

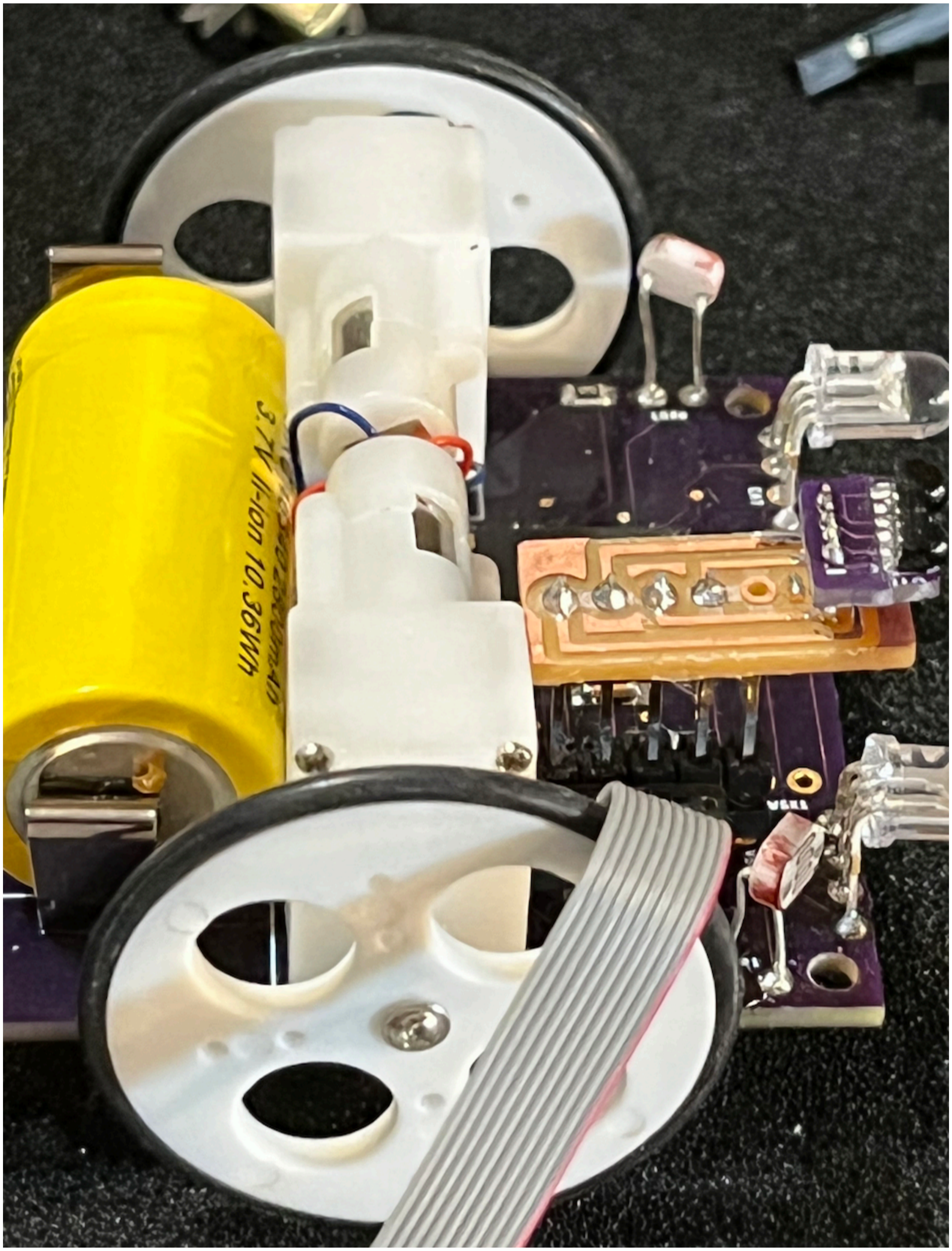
The brd0033 interociter board is even older, though I had used it early on to get initial i/r links working. Of the three that I built way back when, one seems to work fine, one seems to refuse to receive, and the third seems to repeatedly hard-fault. Same code, same everything, it just faults repeatably. Not sure if its a hardware problem, but it may be caused by some bad soldering. I need to analyze it, and the tone that won't receive.

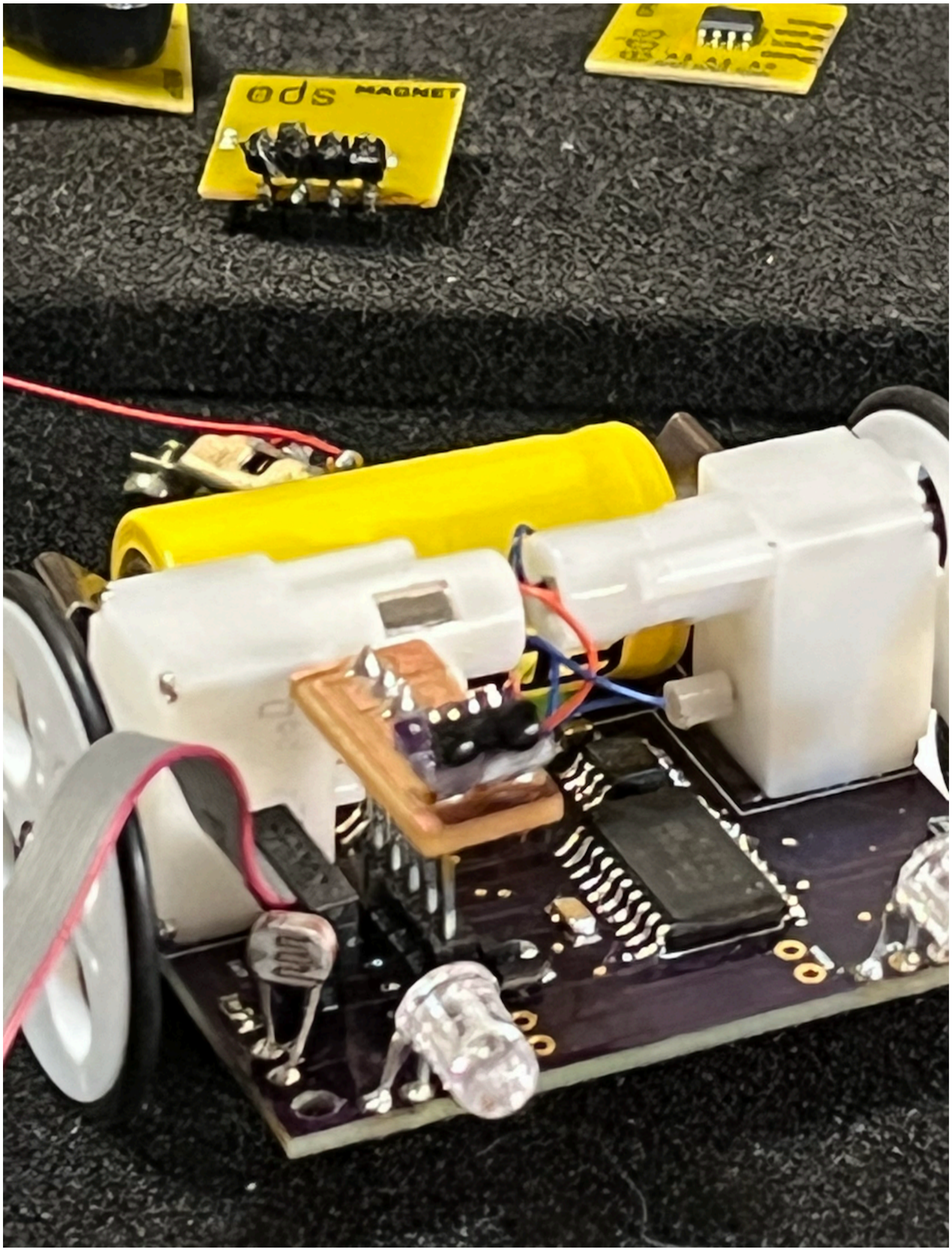
In any case, now I need to get the other Rykors fitted out with i/r. And I need to do it with something much less wombly than the dangly wire I used to prototype it. Then I can start evaluating how well this i/r link is going to work for networking multiple robots. I expect the range to be limited, and the beams to be narrow and directional. I have not yet decided if directionality is an advantage or a disadvantage.

11/05/23
Today I decided to make a small PCB to attach the i/r transceiver more securely. It was fun to break out mu milling machine again and carve out some small boards.






3.7V li-ion 10.36Wh


ods
MAGNET

And, surprisingly, it worked. I used another of my little i/r to 1.27mm dapta boards to mount the transceiver, which made it a little easier to assemble the whole thing. After building the first one, and testing it, I went ahead and miller two more. I still need to solder them together, but that shouldn't be too hard.

Anyhow, my initial experiments indicate that, as I suspected, I can get decent range at 9600 baud (like 6 feet is as far as I have tested), but the beam is quite directional. I'm starting to think maybe that's ok. If you can receive data from another robot then you know that you are more or less facing each other.

I also did more experiments with the LDRs, and the RGB LEDs as sensors. I placed the robot on the floor, and the computer on the desk, displaying the sensor readings beamed up from the robot. I then illuminated the robot with the beam of a flashlight that emits white, red, green, or blue light while sitting in the desk chair. The sensor readings indicated that the LDRs and RGB LEDs do in fact register the difference in intensity of the incoming light. When the room is darkened the sensitivity seems to be higher.

Aside from soldering the two remaining i/r transceivers, the main thing to work on next is a reliable protocol to ensure that the base station can communicate with the robots, and that the robots can communicate with each other. I have implemented multiple protocols over the years, and they are all more complex that I would like. My MCUs don't really have the memory necessary for large packets, and large packets don't work well with the way my Forth implementations parse their input.

I am willing to trade some efficiency for simplicity. There isn't likely to be a vast amount of data flowing at any given time, so I am not overly worried about throughput. I am concerned with reliability though. So I have convinced myself that the simplest, hopefully reasonably reliable, protocol is to employ parity, two stop bits, and to repeat every byte. I.e., when the sender transmits a byte the receiver echos it back.

That, combined with a timeout, gets most of the way there. Ignoring the fact that it is at best 50% efficiency, it handles the case of transmit ... acknowledge, and transmit ... timeout. But it doesn't handle retransmission. The ack may be lost for two reasons; the receiver does not receive the byte, or the sender does not receive the ack sent by the receiver. Therefore the receiver does not know it it should drop a retransmit that it has already acknowledged.

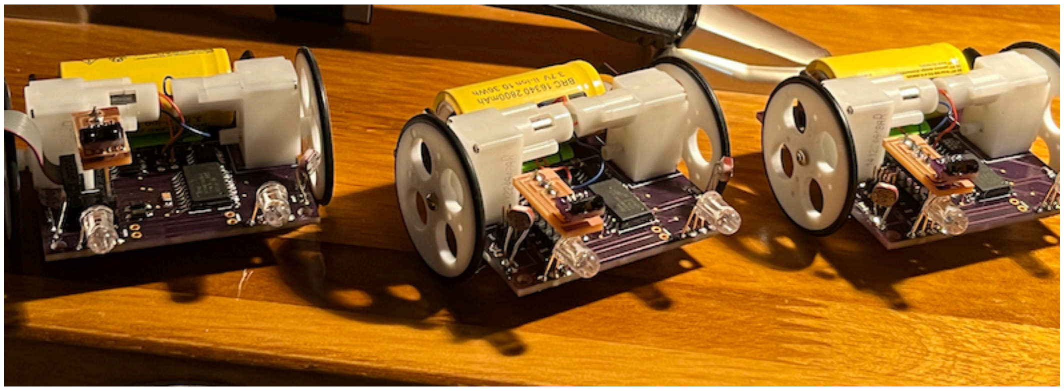
Correct handling or retransmissions requires another bit, which would work as a toggle, or as a retransmit indicator. And there are two ways to get another bit.
1) Put the UART into 9bit mode, or
2) Only transmit 7 bit ASCII and use the 8th bit as the toggle.

The obvious problem with (1) is that not all UARTs support 9 bits,. For instance, termios on MacOSX returns an error if 9 bits are requested. As does picocom. I can work around this because I am building my own USB-to-I./R-UART bridge, so that may be a reasonable choice.

The obvious problem with (2) is that sending only ASCII means transmitting significantly more data to convey numbers etc that transmitting binary values. For example, to send the value 127 takes 6 bytes (because of the ACKs) vs binary's 2 bytes to move the same value across the link. Now for a Forth operator console ASCII is actually what you want. Except for, obviously, downloading compiled code.

Anyhow, I am still working through this in my head. There must be a concept of the selected interlocutor to prevent every robot from answering every byte from the base station. So there must be a higher level protocol for selecting who is allowed to use the air. To be continued.

11/06/23
I built the other two i/r dapta thingies. Now I may be able to issue orders to my army of dorkness. Go me.



**End**