Jada Cumberland
Oct 17, 2025

Weird Machines in Familiar Architectures

For this write-up, I will focus on the Wi-Fi-enabled Zumi cars (https://learn.robolink.com/product/zumi/) utilized this summer during ODU's Cyber Summer Camp. Campers programmed the cars with Python to navigate around obstacles and perform various tasks. Each Zumi hosted its own hotspot, to which the campers connected to upload their code. Each car also had several sensors, allowing it to determine proximity to an object and direction relevant to the initial direction. Several issues arose with the cars, from loss of signal to loss of physical direction.

The most relevant of these errors was the infinite action loops. Although no infinite loops were explicitly programmed, the cars sometimes performed the same action repeatedly (e.g., turning). Given the short amount of time we had, our solution to these problems was always rebooting, but it is an excellent example of an unintended state. This problem was especially prevalent when the Zumis attempted to navigate around an obstacle, which can be considered a form of input in the hardware sense. It was not reliably replicable, nor were we trying to replicate the issue at the time.

With the knowledge gained from this preliminary research, one could consider this infinite turning to be a "weird state", triggered by the input of the sensors identifying an obstacle and trying to turn. A property of the initial state eventually results in a turn that starts and never stops. This reinforces the concept of weird machines being highly initial state-dependent, as seen in *Programming the Weird Machine* (https://gwern.net/doc/cs/security/2011-dullien.pdf). The issue causing this state could be the underlying translation of Python to physical movement, signal loss, or some other unforeseen culprit.

A crucial conclusion to draw from this situation is that the immediate programmer's code is not the only factor that matters. Any cyber-physical system (CPS) must consider the full stack of hardware, firmware, and software that collaborates to perform functions. All aspects of a CPS can lead to vulnerable initial states and "weird machines", even if they were not implemented by the end-programmer/owner of the device. In some ways, an example of this was Stuxnet (https://web-assets.esetstatic.com/wls/2012/11/Stuxnet_Under_the_Microscope.pdf); although the centrifuges themselves were entirely air-gapped (i.e., isolated from other networks), the control software itself was vulnerable and exploited through infected USB drives. The operators and sensors did not cause the centrifuges to be destroyed through direct input, just as our summer camp students did not cause the infinite loop.