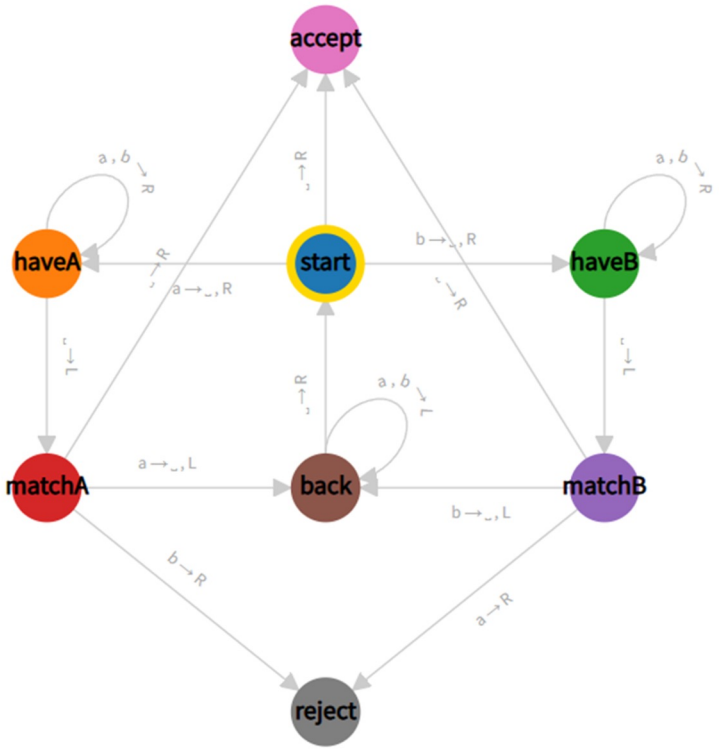
Detecting and Exploiting Emergent Security Behaviors in Cyber-Physical System Architectures

A Weird Machines Approach

Jada Cumberland, Brianne Dunn, Samuel Jackson

Turing Machines



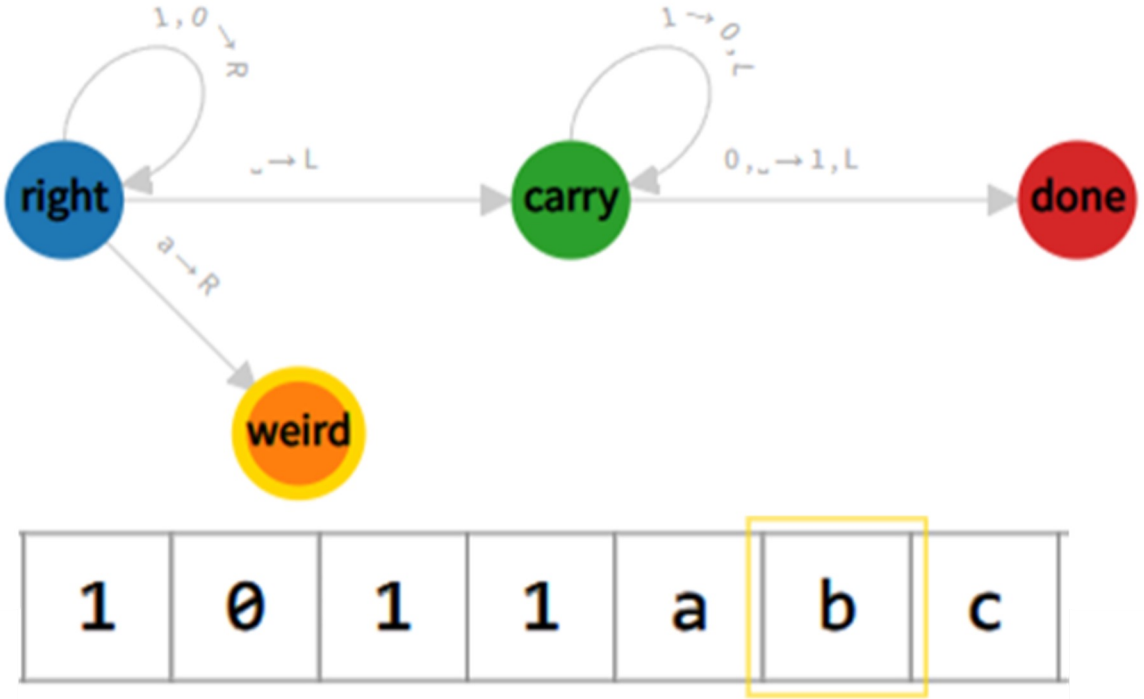
Represent decidable problems

Accepts input and conditionally moves through/edits memory tape

Programs are **implementations** of TMs

What is a weird machine?

- A computational artifact where code execution can happen outside the original specification of the program due to unexpected input



Weird Machine Reading List

- Dullien, T. Weird Machines, Exploitability, and Provable Unexploitability
- Bratus, et al. Exploit Programming: From Buffer Overflows to “Weird Machines” and Theory of Computation
- Bratus, et al. Hacking & Computer Science
- Pavlovic, Seidel. Security Science (SecSci), Basic Concepts and Mathematical Foundations
- Dullien, T. Programming the Weird Machine
- Riley, Karen, et al. Crema
- Benjamin, Thomas S., et al. Computing with Time: Microarchitectural Weird Machines

Weird Machine Timeline

Year	Event	Abstraction Level	Significance
1970	Apollo 13 Mailbox [9]	Cyber-Physical	Early documented beneficial cyber-physical weird machine constructed under crisis
1974	Lamport & Palais Glitch [22]	Hardware (Theoretical)	Established mathematical foundation for metastability and unavoidable computational ambiguity
1976	Harrison-Ruzzo-Ullman [7]	Theoretical	Proved Turing completeness in access control matrices; safety undecidable
1988	Morris Worm [32]	Software (Network)	First major buffer overflow exploitation; infected 10% of Internet
1991	Anderson & Gouda [23]	Hardware (Discrete)	Extended glitch phenomenon to digital/discrete domain
1997	Solar Designer return-into-libc [33]	Software (Memory)	First return-oriented attack; bypassed non-executable stack
2007	Shacham ROP [34]	Software (Code Reuse)	Proved ROP is Turing-complete; formalized gadget chains
2010	Stuxnet [17]	Cyber-Physical (Malicious)	First weaponized CPS attack; destroyed 1,000 Iranian centrifuges
2011	Xbox 360 Reset Glitch [35]	Hardware (Fault Injection)	Practical voltage glitching with under $100 equipment; bypassed secure boot
2011	Bratus et al. [1]	Software (Conceptual)	Formalized “weird machine” terminology and theory
2013	Domas MOVfuscator [2]	Software (Instruction-level)	Demonstrated Turing completeness using only MOV instructions
2018	TA505 LOTL Campaign [26]	Software + Tools	Sophisticated living-off-the-land attacks against financial sector
2025	This Work	Cyber-Physical (Dual-Use)	First demonstration of beneficial and malicious CPS weird machines on operational hardware

Thinking about previous experiences and weird machines

- Arista EOS (Extensible Operating System) Logging as a Weird Machine
  - **Intended Machine:**
    - Local logging and remote logging are independent
    - Local logs persist even if remote forwarding fails
  - **Weird Machine:**
    - An emergent control path where the TCP reachability of a remote syslog server controls whether local device logging executes.
- Robolink Zumi Programmable Cars
  - **Intended Machine:**
    - Car executes code as entered, then stops
  - **Weird Machine:**
    - Car catches on specific parts of code and loops
    - Sensor data as input to unintended state

Thinking about previous experiences and weird machines (cont.)

- Swapping Movesets in Tekken 8
    - Users were able to find a bug which allowed male characters to use an unintended slap move on female characters
    - Found through conditionals that incorporated stance changes, specific timing, and a reaction from the opposing character.
    - This allowed for even deeper exploitation, allowing for discoverable data.
- **Intended Machine**
    - A fighting game with rigid movesets, some moves being gender locked to certain characters.
- **Weird machine**
    - Everyone was able to respond to slap prompt, movesets became variables that allowed users to find new data.

Weird Machine Gadgets

What is a gadget?

Set of architectural artifacts that can be composed to construct a program

Architectural artifacts can be:

- Software
- Protocols
- Physical components

Example: Video Game Save File Corruption

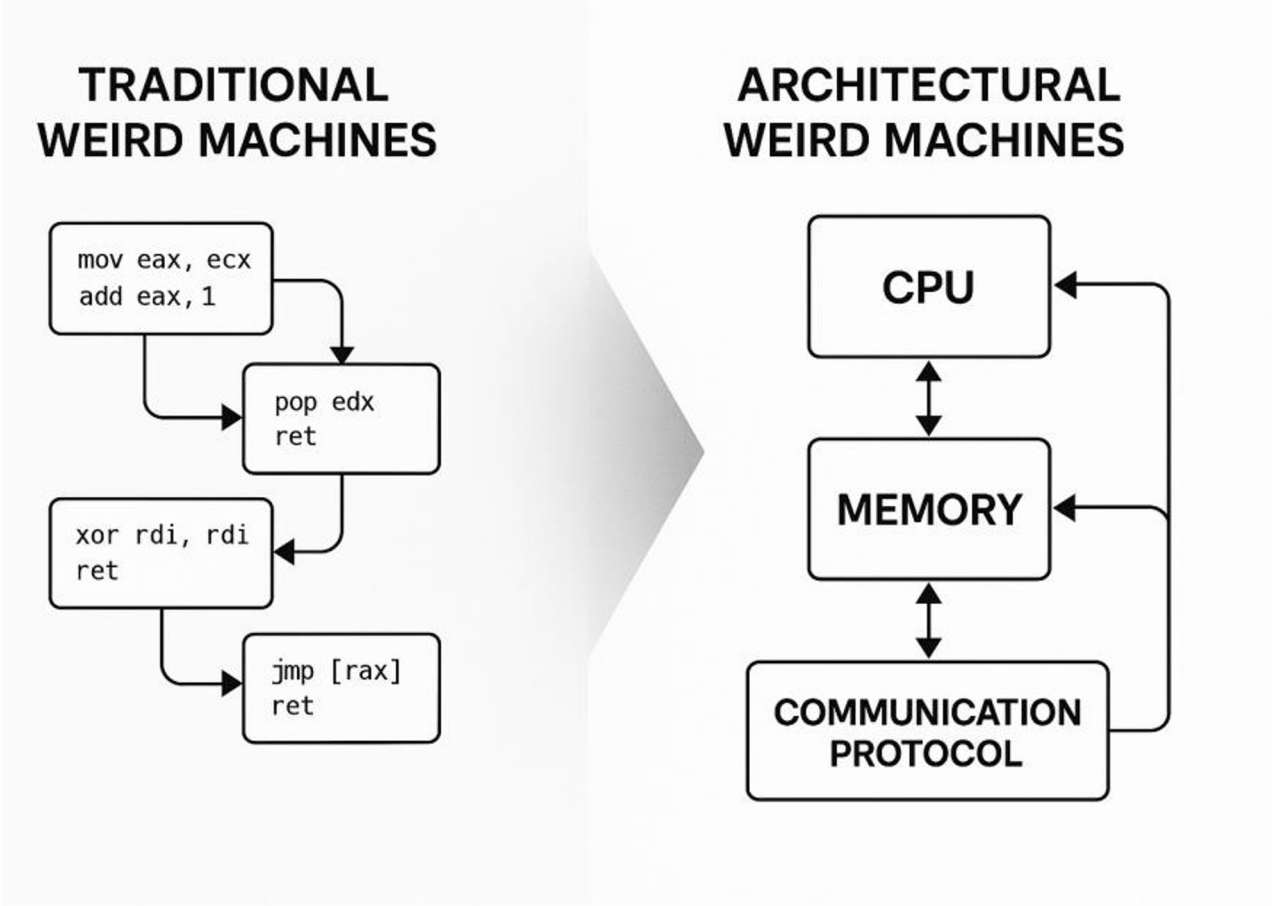
Gadgets: Save file format, game state parser, memory layout

Composition: Carefully corrupt specific bytes in the Final Fantasy VII save file to gain extra items

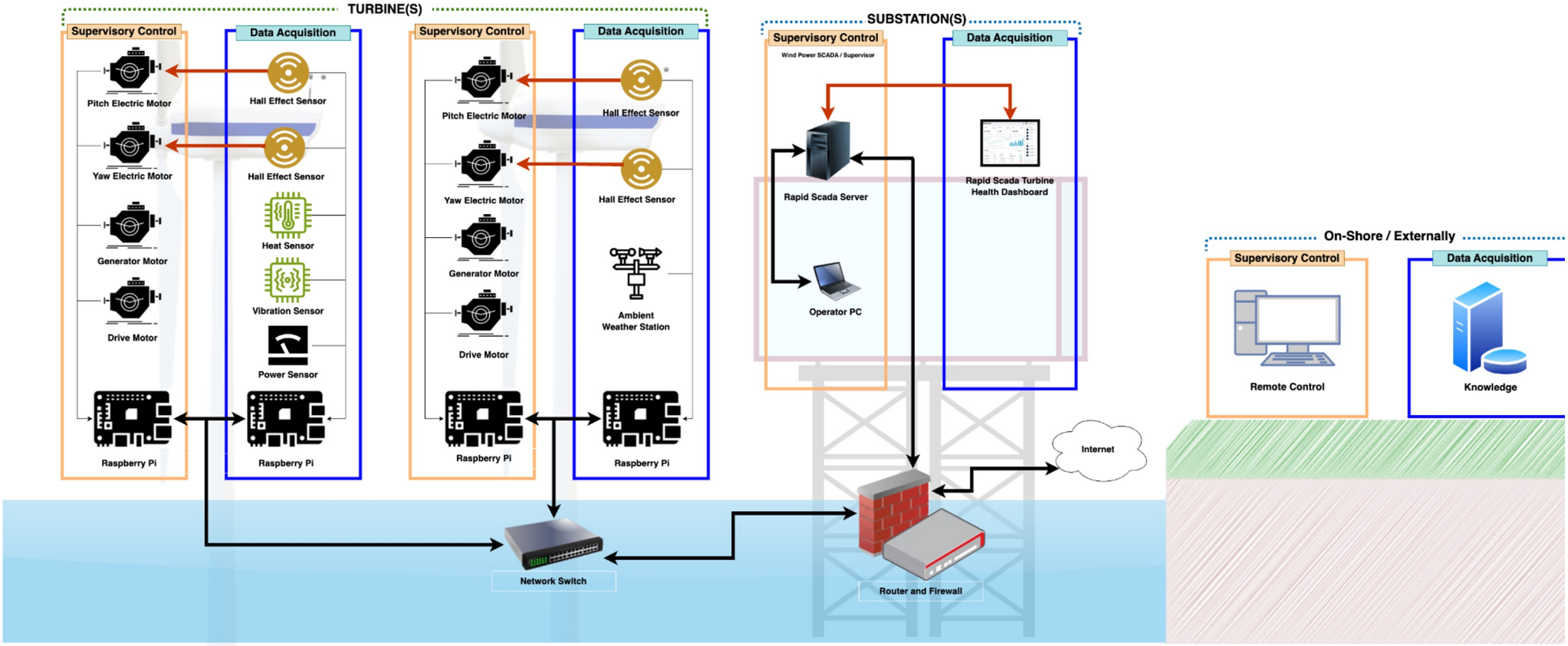
The Final Fantasy VII save file is just data, but when the parser crashes on malformed input, it becomes a lever for control.

Architectural Weird Machines

•Unlike traditional weird machines which use code-level gadgets, architectural gadgets exploit the interactions across system boundaries, communication channels, and operational constraints.



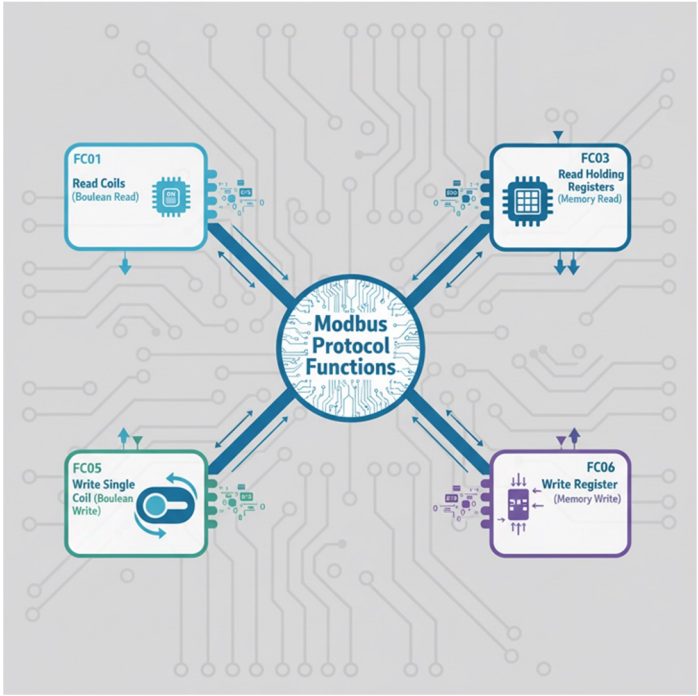
Our Wind Energy Testbed



Wind Energy Testbed Weird Gadgets: Modbus

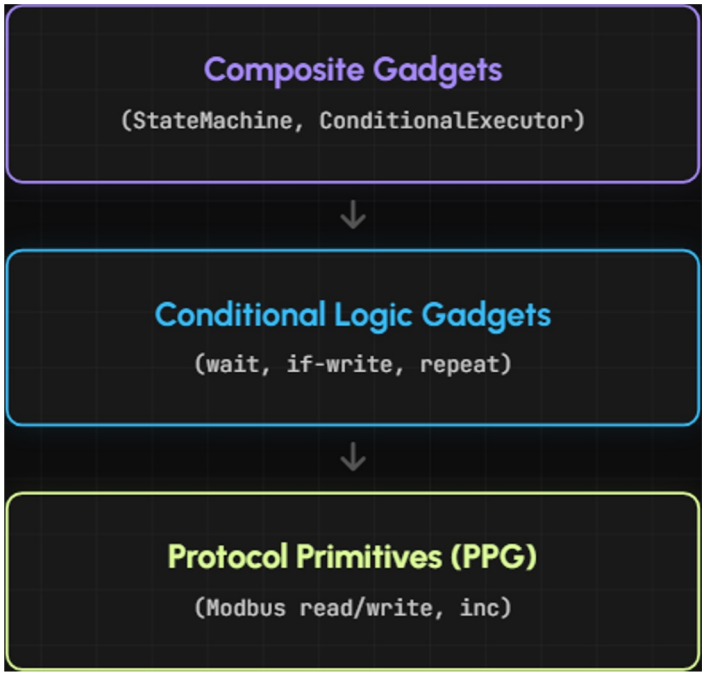
- The testbed uses 16 function code (FC01-FC16) that operate like opcodes in a computer's instruction set. The most significant ones include:

- FC01 (Read Coils): Boolean read operations
- FC03 (Read Holding Registers): Memory read operations
- FC05 (Write Single Coil): Boolean write operations
- FC06 (Write Single Register): Memory write operations



Wind Energy Testbed Weird Gadgets

- Core Idea: Using the Modbus Protocol “opcodes” we can create computational behavior through composition.
- Gadget Stack (3 layers):
  - **PPG (Protocol Processing Gadgets)**
    - Provide register/coil read+write and increment/decrement primitives
  - **CLG (Conditional Logic Gadget)**
    - CLGs provide emergent control-flow-like behavior using only protocol interactions such as while-loops, if-statements, and do-while loop
  - **Composite Gadgets**
    - Higher-level logic like a State Machine or a Conditional Executor



Building weird machines can be hard when you have a software mindset

Software mindset assumes:

- Clear APIs
- Deterministic behavior
- Separation between control and execution

Cyber-physical reality breaks those assumptions:

- Software decisions directly affect physical state
- Hardware timing, sensors, and actuators introduce tangible side effects

Weird machines Emerge:

- From composition, side effects, and misused assumptions

Python gadget libraries helped bridge this gap by:

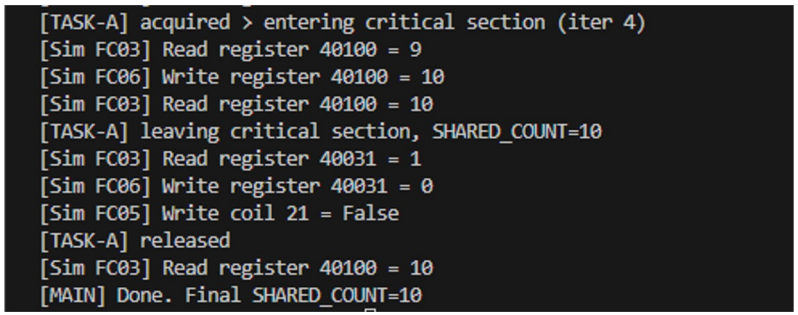
- Abstracting low-level hardware and protocol details
- Exposing simpler, composable primitives
- Enabling rapid experimentation with unintended behaviors
- Result: easier exploration of emergent control and computation

GitHub Repository: https://github.com/rossgore/weird_machine_gadgets

Wind Energy Testbed Weird Machines

- Here are the Weird Machines we composed using our manufactured *Weird Gadgets:*

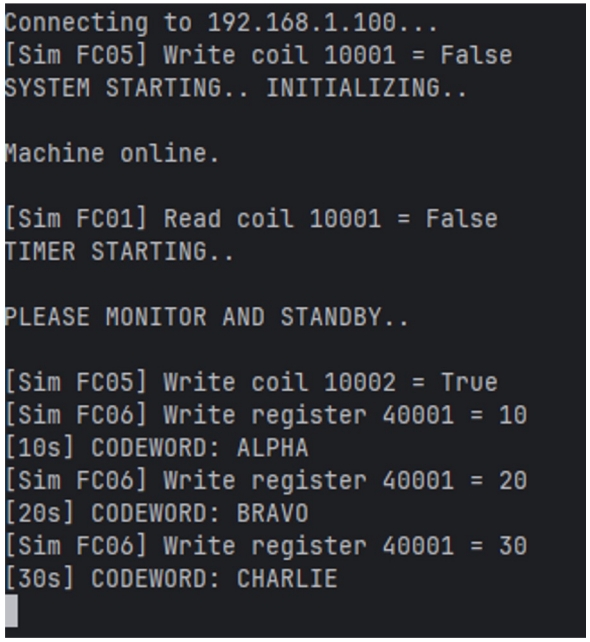
- Binary Semaphore Weird Machine
  - Emergent synchronization primitive built from protocol & control gadgets
  - Enforces mutual exclusion over a shared resource



Wind Energy Testbed Weird Machines (cont.)

Weird Machines composed using our manufactured *Weird Gadgets*

- Code Signaling Weird Machine
  - Utilized modbus coils and registries
  - Machine stored conditional flags to coils to signal whether machine was vulnerable
  - Sent code words from registry,
  - After countdown value would modify coil to display vulnerability
  - **Implications**
    - Allows monitoring of outside factors to determine vulnerable state



Wind Energy Testbed Weird Machines (cont.)

- Remote Mailbox
  - Segmented: Send a message one character at a time in server registries
  - Unsegmented: Save and retrieve messages in server registries
  - Utilizes: Conditionals, loops, read/write to registers/foils

```
[Sim FC06] Write register 40010 = 84
[Sim FC05] Write coil 10010 = True
[Sim FC01] Read coil 10010 = True
[Sim FC03] Read register 40010 = 84
Character detected: T
[Sim FC05] Write coil 10010 = False
```

What is Turing Completeness?

- Programming Language with full capabilities of a Turing Machine:
  - Sequence: Read/write to arbitrary memory
  - Selection: Make conditional decisions
  - Iteration: Move/jump through memory, loops

Are the Wind Energy Testbed Gadgets Turing Complete?

- Yes, they satisfy the necessary qualities of Turing Completeness
  - Sequence: Capability of reading/writing to registers and foils with “get” and “set”
  - Selection: Conditional writes simulate if-then-else
  - Iteration: Wait or repeat until condition is satisfied with ControlGadget
- Weird Machines => Turing Complete => Solve any “decidable problem”

This work argues Turing completeness by construction and analogy; a formal proof is out of scope

What is coming next?

- **White Paper & Research Artifacts**
  - Document our findings, methodology, and lessons learned in a formal white paper. Publication and public release will occur only after sponsor review and approval.
- **Research Workflow & Tooling Experience**
  - This research project provides hands-on practice in technical writing, version control, and reproducible research workflows utilizing tools such as Overleaf, Git, and LaTeX.
- **Scaling Weird-Machine Discovery with LLMs**
  - Explore a large-language-model consortium approach to assist in identifying potential weird gadgets within complex system architectures such as our Wind Energy Testbed.

LLM Consortium For Identifying Weird Gadgets from System Documentation

Multiple fine-tuned llms independently generate responses, creating redundancy and reducing bias / hallucination

Dedicated reasoning model compares, validates, and synthesizes these outputs into a single trusted result