

Secret Battleship: A Two-FPGA Acoustic Battleship Game over Binary FSK Communication

Fares Pasuni (fipasuni), Muath Alghamdi (muath)

Abstract—Secret Battleship is a two-board FPGA project that implements the classic Battleship game played entirely over an acoustic communication link. Each FPGA transmits and receives fixed-length packets encoded using binary frequency-shift keying (FSK), where 4 kHz and 7.68 kHz tones represent logic 0 and logic 1, respectively. The physical layer design includes an I²S audio interface, two parallel FIR-based frequency discriminators, and a majority-vote symbol classifier. Each bit is transmitted using 600 tone samples followed by 200 silence samples, yielding a data rate of 40 bps at a 32 kHz sample rate. To compensate for the omission of a Cyclic Redundancy Check (CRC), a simple time-out and explicit ACK/NACK-based reliability layer was implemented. This report details the system design, the implementation of the custom link-layer protocol, quantitative evaluation of throughput and latency, and analysis of the project’s high DSP resource consumption, which was a critical design trade-off for the real-time signal processing requirements.

Index Terms—FPGA, I²S, FSK, Acoustic Communication, Digital Signal Processing, Battleship, Reliability Layer.

I. INTRODUCTION

This project addresses the non-trivial challenge of building a reliable, bidirectional, packet-based communication link using only audible sound. Secret Battleship is a complete implementation of the classic board game [1] played between two FPGAs, utilizing a speaker and MAX98357A amplifier [2] and an ICS-43434 digital microphone [3] on each board for all data exchange.

The central technical challenge lies in designing a custom physical layer entirely in hardware, capable of robustly decoding frequency-shift keying (FSK) symbols from a high-noise, high-latency acoustic channel. This required overcoming several hurdles: stable timing for the I²S interface, implementing high-selectivity Finite Impulse Response (FIR) filters for frequency discrimination, and developing a symbol debouncing and packet synchronization scheme to mitigate noise-induced errors.

II. PROBLEM SUMMARY AND REQUIREMENTS

The design goals centered on achieving a functional, reliable game loop.

A. Technical Challenges

The acoustic channel introduces several practical challenges. Reflections and small variations in the environment can distort

the received tones, and without a shared clock the two boards must rely on timing structure in the signal for synchronization. Background noise also affects the clarity of the tones, which forces the filtering stage to be more selective to avoid misclassification. In addition, the limited usable audio bandwidth restricts the data rate, so the communication protocol must remain simple and efficient.

B. Project Checklist Requirements

- Minimum: Working I²S audio receive and transmit chains; FSK-based differentiation of 4 kHz/7.68 kHz tones; a playable Battleship demo with a fixed communication channel (met).
- Ideal: Complete Battleship game with both boards fully synchronized; a full packet send/receive system using a time-out and explicit ACK/NACK reliability layer; functional, full-game play over the acoustic link (met).
- Stretch: Dynamic ship placement (achieved using controller input); Sound effects and UI polish (achieved with BRAM sprites).

The project successfully achieved the ideal goals, resulting in a complete and interactive acoustic game system. Reliability was handled through a combination of aggressive symbol debouncing and a lightweight ACK/NACK exchange, which together provided sufficient robustness for turn-based gameplay.

III. HIGH-LEVEL SYSTEM ARCHITECTURE

A. Clocking and Timing

The FPGA’s 100 MHz reference clock was used to derive two main clock domains. The game logic, HDMI, and peripheral FSMs operate on a high-speed clock near 100 MHz. To ensure stable timing for the audio interface, the Xilinx Clocking Wizard [5] generated a separate 102.4 MHz clock from the 100 MHz reference. This 102.4 MHz clock enables precise generation of the I²S Serial Clock (SCLK) of 2.048MHz (102.4MHz/50), which is required to achieve the 32 kHz audio Word Select (WS) rate (2.048MHz/64). This dual-clock approach minimizes jitter on the audio data streams.

B. Protocol Stack and Packet Format

The protocol is a symmetric, fixed-length 32-bit packet format:

Packet = password[15 : 0] | row[3 : 0] | col[3 : 0] | flags[7 : 0]

Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA, USA.

This work was performed as a final project for the MIT course 6.205 Digital Systems Laboratory (F25).

- **Password (16 bits):** A fixed preamble for basic packet boundary detection.
- **Coordinates (8 bits):** Encodes the target row and column (0-9).
- **Flags (8 bits):** Contains command opcode (MOVE, RESULT, ACK, NACK), and status bits (hit/miss).

C. Memory and IP Cores

The design is optimized for resource usage, with state stored primarily in LUTs and graphical assets in Block RAMs:

- **Board State LUTs:** Each FPGA stores a 10×10 grid of cells, with 2 bits per cell (ship/hit/miss/no-info), totaling approximately 200 bits of game state stored in distributed registers (LUTs).
- **Sprite BRAM:** A small BRAM holds ship sprites for the HDMI display.
- **ASCII BRAM:** A small BRAM stores bitmaps for a minimal ASCII set used in status messages and on-screen text.

Vendor IP cores include the clocking wizard [5], BRAM generators, and the existing video timing and TMDS encoder [4].

IV. ACOUSTIC COMMUNICATION LINK IMPLEMENTATION

Audio Input Chain (Receiver)

The receiver's primary task is to transform noisy 24-bit analog-to-digital (ADC) samples into reliable, time-aligned bits.

1) *FIR Bandpass Filters:* Two dedicated FIR filters were implemented: one centered at $f_0 = 4$ kHz (logic 0) and one at $f_1 = 7.68$ kHz (logic 1). The design for these filters, which required a large number of multiply-accumulate operations, led to the high consumption of the dedicated DSP blocks on the FPGA (see Evaluation). The filter output magnitude for the two bins, $|Y_{4\text{kHz}}|$ and $|Y_{7.68\text{kHz}}|$, is continuously compared.

2) *Symbol Classification and Debouncing:* Each symbol is represented by 800 samples:

$$600 \text{ tone samples} + 200 \text{ silence samples}$$

To stabilize symbol detection, the system uses a majority-choice filter that confirms a bit only if the same decision appears across several consecutive raw samples (Majority-of-5). This helps reject short noise spikes and leads to more reliable packet reconstruction.

3) *Packet Framer:* The framer consumes the debounced symbol decisions generated by the majority-choice filter. Whenever the classifier outputs a valid 0 or 1, that bit is shifted into a 32-bit register. The 200-sample gap after each 600-sample tone appears as an *idle* region at the classifier output. The framer watches for a predefined pattern (the 16-bit password) and the mandatory quiet gap, which marks the end-of-packet and asserts a packet-valid signal.

Audio Output Chain (Transmitter)

4) *Packet Encoder:* Packets contain:

password[15 : 0], row[3 : 0], col[3 : 0], flags[7 : 0]

5) *FSK Tone Generator:* The transmission of FSK symbols relies on directly outputting pre-calculated sinusoidal samples to the I²S amplifier. The samples representing the tones for logic 0 (4 kHz) and logic 1 (7.68 kHz) are stored as fixed constants, typically implemented using SystemVerilog `localparam` within the tone generator module.

- **Logic 0 (4 kHz):** The tone is represented by 8 unique samples. This number is precisely determined by the 32 kHz sampling frequency (f_s). Since 4 kHz is exactly 1/8th of f_s , using 8 samples ensures a perfect, continuous repeating sine waveform without truncation errors ($32 \text{ kHz} / 4 \text{ kHz} = 8$).
- **Logic 1 (7.68 kHz):** The tone is represented by 5 unique samples. This frequency (7680 Hz) was specifically chosen to be 1/5th of an integer multiple of the sample rate ($32 \text{ kHz} \times 1.2 = 38.4 \text{ kHz}$) to provide clean separation from the 4 kHz tone while maintaining an integer number of samples per cycle for simplified implementation.

The tone generator enforces the symbol structure by streaming 600 samples of the appropriate tone, followed by 200 samples of zero (silence), before moving to the next bit. This sample-split yields the 40 bps bitrate and provides the necessary quiet gap for packet framing.

V. GAME ENGINE AND FSM

The FSM handles the full match lifecycle. The critical reliability logic is embedded in the `Playing` state:

Algorithm 1 ACK/NACK and Retransmission Protocol

- 1: Player FSM sends MOVE packet.
 - 2: FSM starts `TIMEOUT_COUNTER` $\leftarrow T_{\text{max}}$.
 - 3: Remote FSM receives packet.
 - 4: **if** Remote FSM decodes valid MOVE **then**
 - 5: Remote FSM sends RESULT (Hit/Miss) or ACK.
 - 6: **else if** Remote FSM decodes invalid packet (e.g., timing/format error) **then**
 - 7: Remote FSM sends NACK.
 - 8: **end if**
 - 9: **if** Local FSM receives NACK **OR** `TIMEOUT_COUNTER = 0` **then**
 - 10: Retransmit MOVE packet (up to 3 times).
 - 11: **else**
 - 12: Update board and transition turns.
 - 13: **end if**
-

This high-level retransmission protocol compensates for the absence of per-packet CRC checking, ensuring game state synchronization despite occasional bit errors.

TABLE I
FPGA RESOURCE UTILIZATION (XC7S50CSGA324-1)

Resource	Used	Available	Utilization
Slice LUTs	7,348	32,600	22.54%
Slice Registers	2,623	65,200	4.02%
DSPs	114	120	95.00%
Block RAM Tile	19	75	25.33%

VI. EVALUATION

A. Performance: Latency and Throughput

The low symbol rate dictated by the acoustic channel and required filtering causes significant latency overheads.

- Symbol Time (T_{sym}): $T_{\text{sym}} = 800 \text{ samples}/32 \text{ kHz} = 25 \text{ ms/bit}$.
- Bitrate (R): $R = 1/T_{\text{sym}} = 40 \text{ bps}$.
- Packet Transmission Time (T_{pkt}): $T_{\text{pkt}} = 32 \text{ bits} \times 25 \text{ ms/bit} = 0.8 \text{ s}$.

A full turn (Move $\rightarrow$ Result) requires a minimum of $2 \times T_{\text{pkt}} \approx 1.6$ seconds, representing the minimum latency achievable under these physical layer constraints.

B. Resource Utilization and Timing

Table I presents the final resource consumption derived directly from the Vivado implementation report.

The design is dominated by the 114 DSP blocks used (95% utilization). This extreme utilization stems from implementing the two FIR filters entirely using DSP slices for maximum throughput and precision, confirming that the real-time signal processing requirements of the acoustic link were the primary consumer of specialized hardware. The 19 Block RAM Tiles were primarily used for the video frame buffer (tiles) and sprite storage. Timing constraints were met on all clocks, demonstrating robust Clock Domain Crossing (CDC) logic.

C. Acoustic Robustness and Reliability

The reliability of the acoustic link depended heavily on the physical distance between the two boards. At close range (approximately 2–5 cm), the received tones were strong and well-defined, resulting in consistent bit classification and stable packet reconstruction. As the separation increased, however, the ICS-43434 microphone struggled to capture the transmitted tones with sufficient clarity. Beyond roughly 25–30 cm, the signal-to-noise ratio dropped enough that the classifier began to misidentify symbols, leading to a noticeable decline in overall link performance. These distance-related limitations stem primarily from the microphone’s reduced sensitivity to higher-frequency tones at greater distances and the naturally high attenuation of acoustic energy in air.

VII. DISCUSSION AND IMPLEMENTATION INSIGHTS

A. DSP Utilization as a Design Trade-off

The near-total use of the FPGA’s DSP resources was unexpected, but ultimately necessary for the project’s core

signal-processing requirements. Although we initially estimated lower usage, the dual FIR filters—running continuously at a 32 kHz sample rate—required a large number of multiply-accumulate operations. Implementing these operations with dedicated DSP hardware avoided timing violations and prevented an even larger consumption of general-purpose LUTs and flip-flops. In practice, this validated our design choice: prioritizing real-time audio filtering performance at the cost of most of the device’s DSP budget.

B. Alternative to CRC

The biggest insight was successfully relying on the Majority-of-5 debouncer to clean the incoming bitstream and the ACK/NACK protocol to ensure link-layer integrity. This simplified the FPGA logic by removing the computationally expensive CRC LFSR or LUT implementation, but required a strict FSM timeout logic to manage retransmission. This architectural choice proved viable for the low-speed, high-redundancy acoustic channel.

VIII. CONCLUSION

Secret Battleship successfully demonstrates a fully functional two-way acoustic communication system built entirely on FPGA hardware, integrating real-time audio signal processing with a complete turn-based game engine. The system achieves a reliable end-to-end communication channel over sound, proving that low-speed, custom protocol development is viable on constrained hardware, even when critical components like CRC are deliberately omitted and replaced with higher-layer reliability mechanisms.

AUTHOR CONTRIBUTIONS

Muath Alghamdi: Designed and implemented the full Audio Chain (I²S, FIR filters using DSPs, symbol classifier, FSK tone generator), executed all quantitative latency and PER testing, and co-authored the report.

Fares Pasuni: Designed and implemented the Game FSM (including the ACK/NACK logic), the HDMI/Screen Manager subsystem (sprite and tile rendering), managed system integration and clock domain crossing logic, and co-authored the report.

Acknowledgments: The authors thank the MIT 6.205 staff for providing the base FIFO, HDMI infrastructure, and initial I²S controller references.

REFERENCES

- [1] Hasbro, “Battleship Game,” Accessed: Nov. 2025.
- [2] Analog Devices, “MAX98357A/MAX98357B Amplifier,” 2016.
- [3] TDK InvenSense, “ICS-43434 Microphone,” 2017.
- [4] MIT EECS, “6.205 HDMI Test Infrastructure,” 2025.
- [5] AMD, “Clocking Wizard v6.0,” 2024.

Audio Input

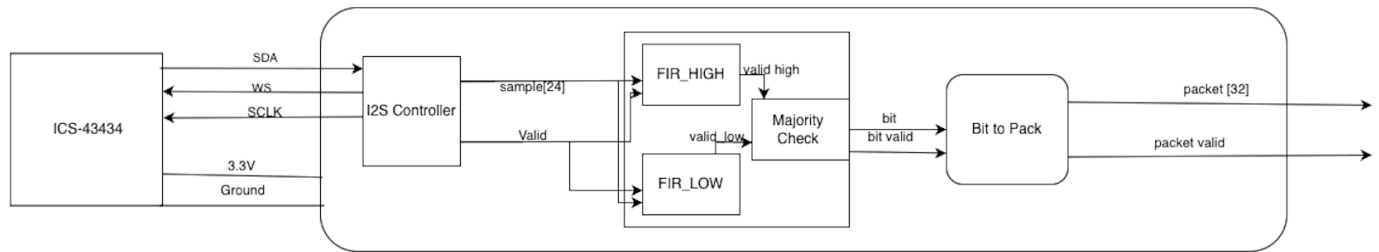


Fig. 1. System-level block diagram of the audio receive chain, illustrating the flow from the I²S microphone interface through the FIR-based frequency discriminators, the majority-vote symbol classifier, and finally the packet framer. This portion of the design performs all tone detection and bit reconstruction required for reliable acoustic communication.

Speaker

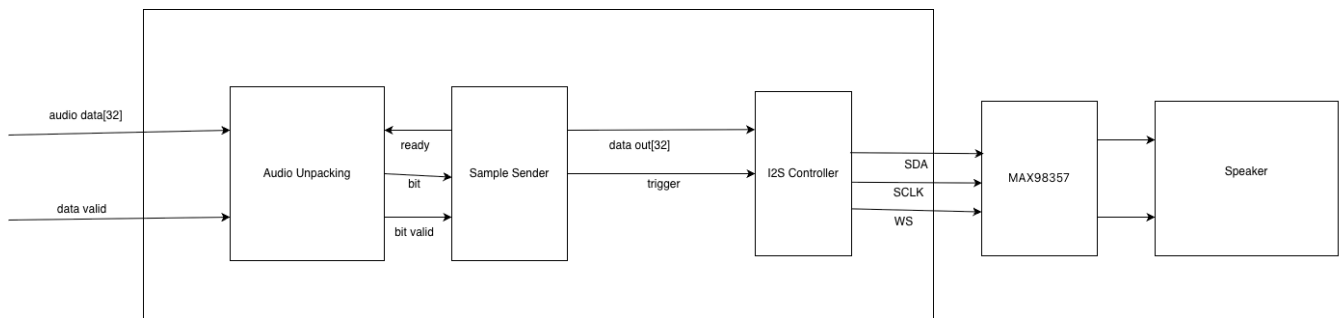


Fig. 2. Block diagram of the audio transmit chain, showing the flow from the packet encoder through the FSK tone generator and I²S transmit interface. This module produces the 4 kHz and 7.68 kHz tones used for symbol transmission and enforces the 600/200 sample structure.

Game Logic

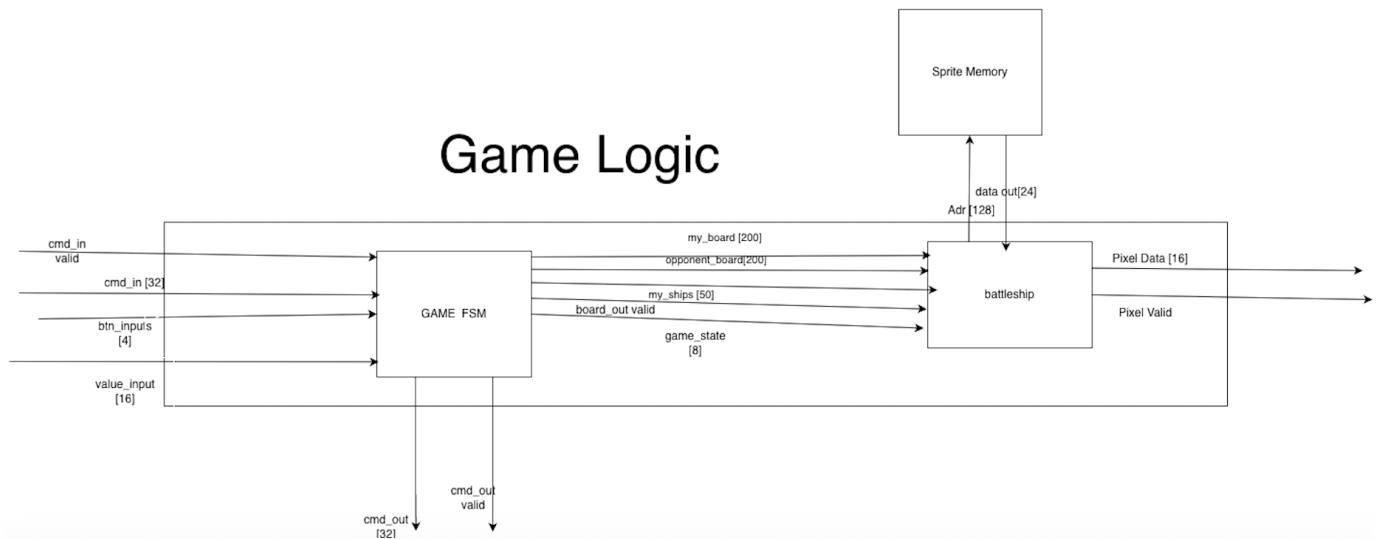


Fig. 3. High-level structure of the game engine, including the Game FSM, board state registers, UI control logic, and the HDMI rendering pipeline. This diagram highlights the interaction between gameplay logic and the display subsystem.

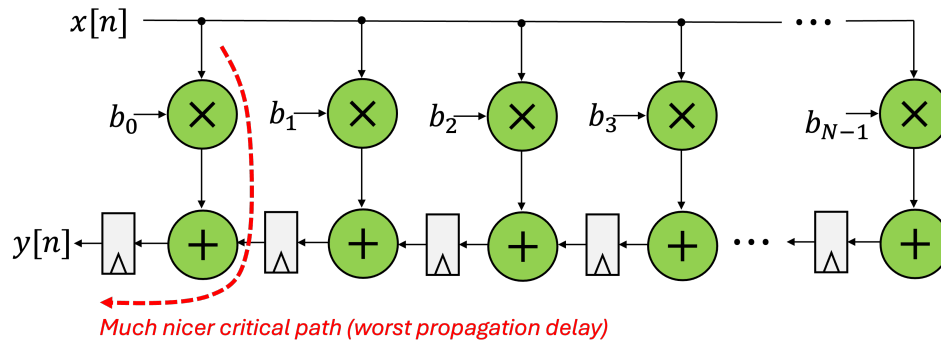


Fig. 4. Architecture of the FIR filter used in the audio receive chain. The heavy arithmetic load from the dual bandpass filters contributed significantly to DSP slice usage. Adapted from 6.S965 (https://rfsoc.mit.edu/6S965/F25/assignments/week03/fir_design).

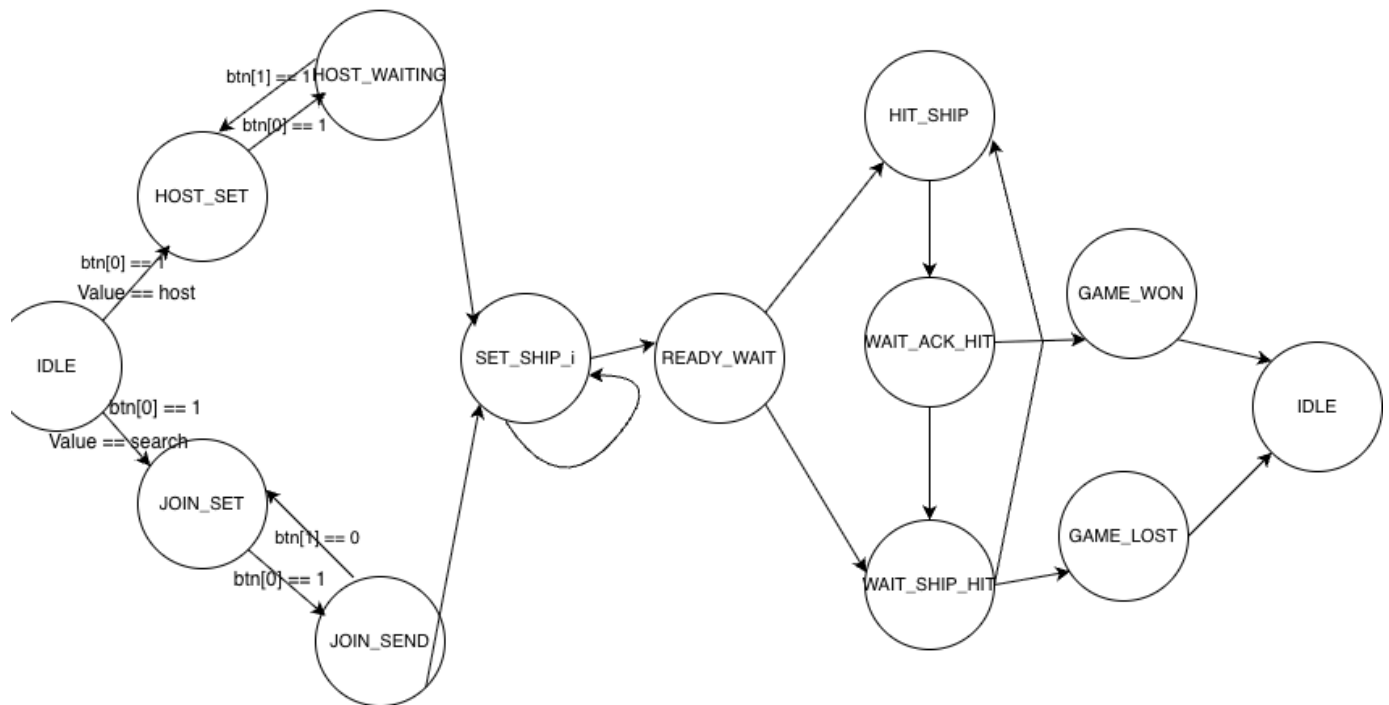


Fig. 5. State Diagram of the Game FSM, governing the host/join states, ship placement, and the core turn-taking/reliability loop enabled by the ACK/NACK protocol.

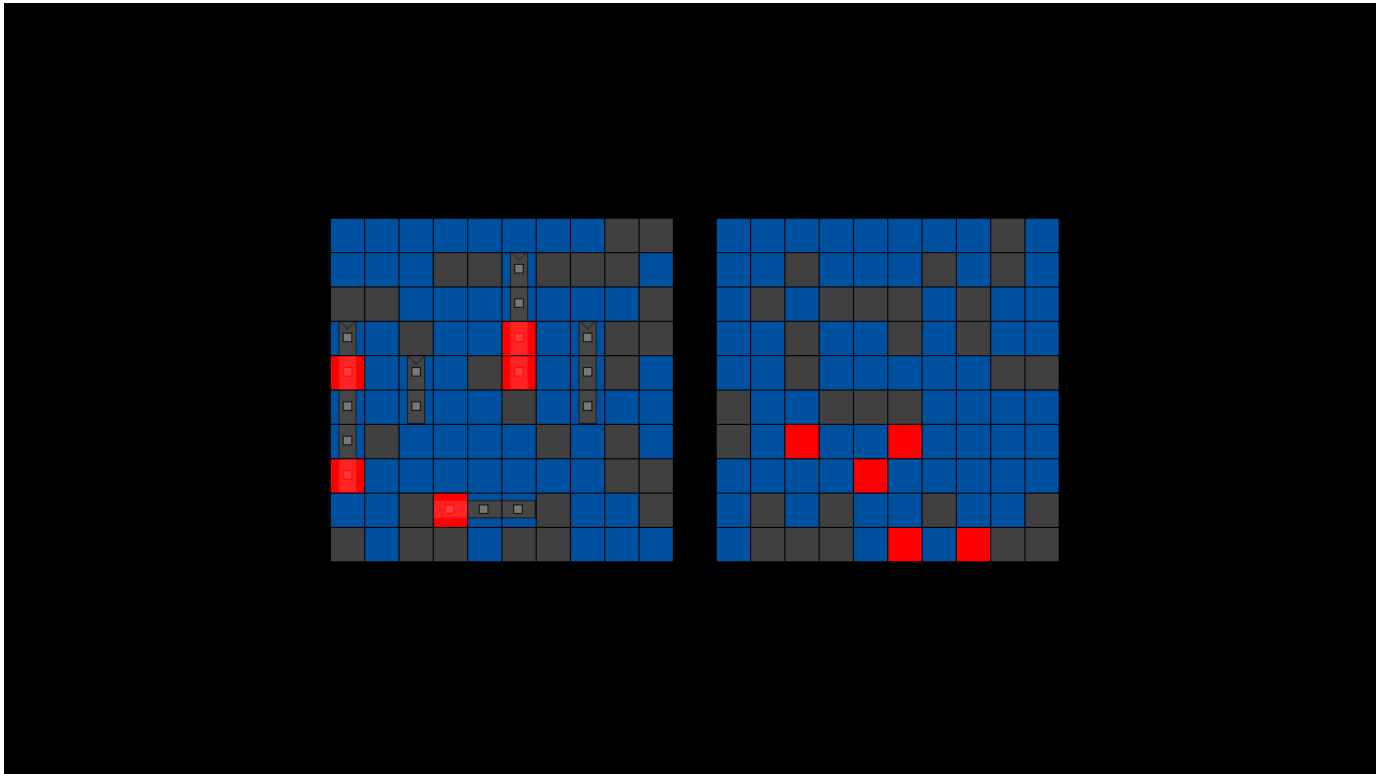


Fig. 6. Screenshot of the Secret Battleship HDMI output taken mid-game, showing the dual-board view (local ship placement and remote targeting board) and status text.