Presented By:
Thomas, Darragh & JJ

TECH

FW

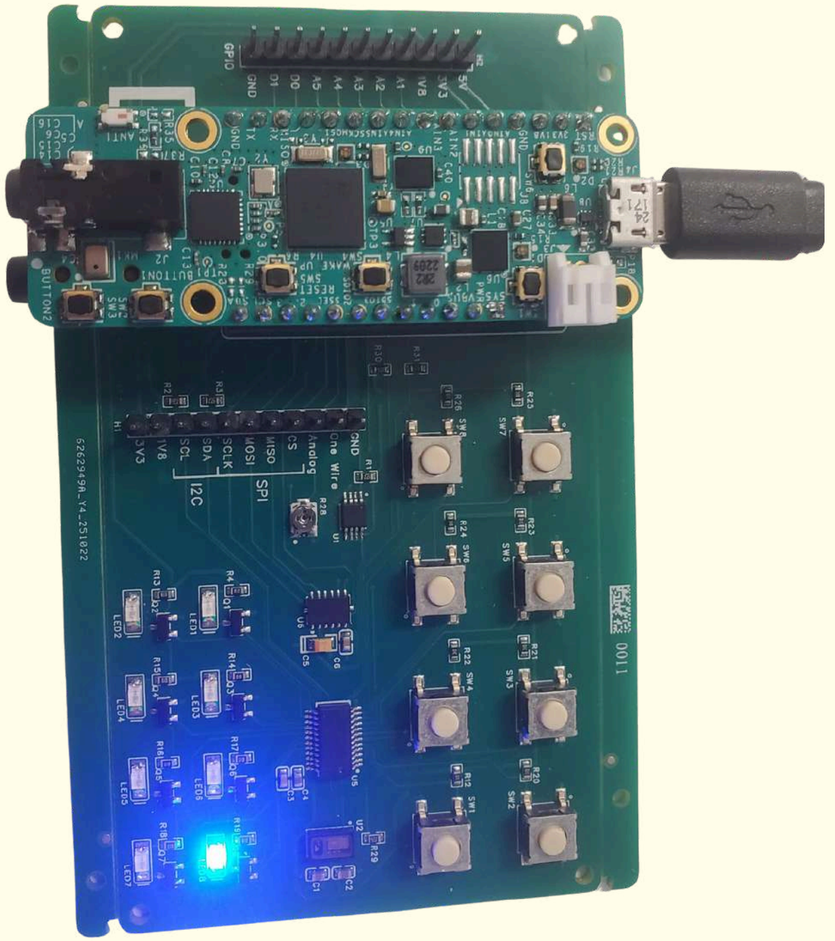
- C
- Dependencies: FreeRTOS, MaximSDK

HW

- MAX32655FTHR EVKit for MAX32655 (ISE I/O Board)

Key Peripherals

- 8 LEDs
- 8 corresponding buttons

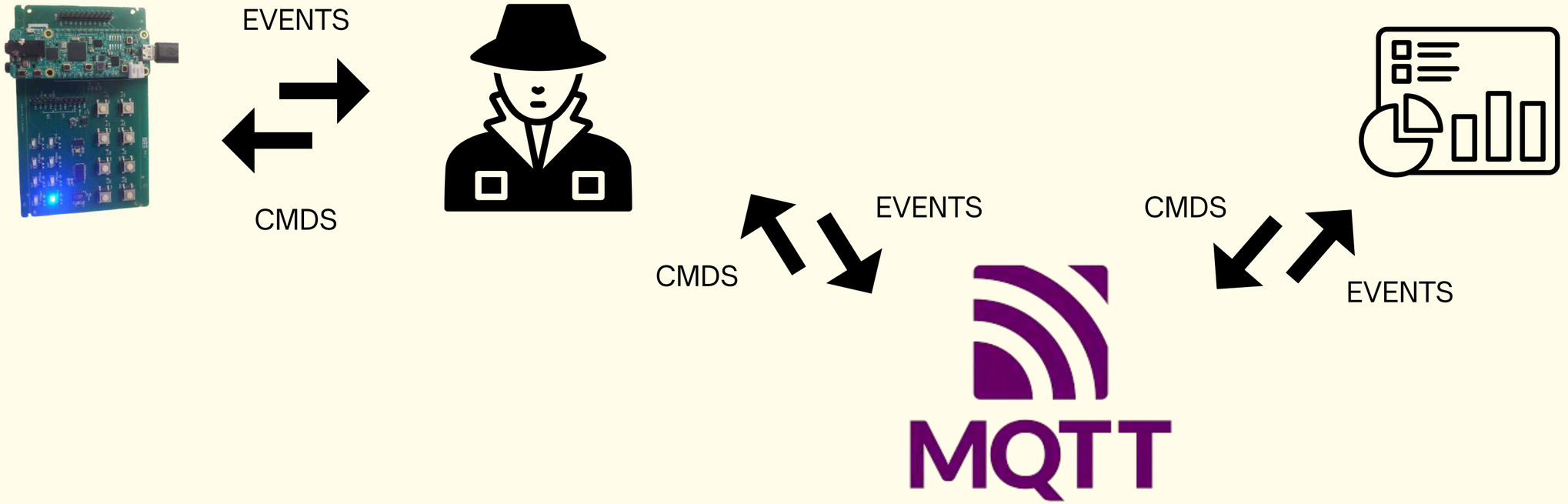


THE GAME

- Press the button matching the lit LED before it turns off
- 8 levels (1500ms → 275ms pop duration)
- 5 lives
- Score = points × level × speed bonus

SYSTEM OVERVIEW

Device / Agent / Dashboard



“whac/{device_id | all}/{state | commands | game_events}”

FREERTOS TASK ARCHITECTURE

MULTI-THREADED WITH PRIORITY LEVELS

3 Tasks with priorities:

- Pause Task (Priority 3) - Immediate preemption
- Game Task (Priority 2) - Deterministic 5ms polling
- Agent Task (Priority 1) - UART comms, non-blocking

```
#define PAUSE_TASK_P (IDLE_TASK_P + 3) // Priority 3 (Highest for immediate preemption)
#define GAME_TASK_P  (IDLE_TASK_P + 2) // Priority 2 (Game logic should not be blocked)
#define AGENT_TASK_P (IDLE_TASK_P + 1) // Priority 1 (Low priority comms)

#define IDLE_TASK_P 0 // Priority 0 (System idle)
```

BI-DIRECTIONAL QUEUING

Event Queue: Device → Agent (Optionally buffered)

- Pop results, level complete, session events

Command Queue: Agent → Device

- Start, reset, level change

```
QueueHandle_t event_queue;   // Game → Agent (32 events)
QueueHandle_t cmd_queue;     // Agent → Game (8 commands)
```

Queue 1: Event Queue (Game → Agent)

- Producer: Game task generates events (pop results, level complete, etc.)
- *Consumer*: Agent task drains events and sends via UART
- Thread-safe FIFO with automatic blocking

Queue 2: Command Queue (Agent → Game)

- *Producer*: UART ISR receives commands from the dashboard
- *Consumer*: Game task processes commands (level changes, reset, start)
- Bidirectional control from the remote dashboard

REAL-TIME CONTROL

5ms Button Polling Loop

- Deterministic timing: vTaskDelay() provides tick-accurate delays
- Low latency: Maximum 5ms detection delay
- Priority protection: Higher priority prevents preemption by communication tasks
- Millisecond precision: Reaction time measurement accurate to 1ms

COMMUNICATION PROTOCOLS

JSONL / SINGLE ASCII BYTE

AGENT → DEVICE (COMMANDS)

```
// Single-byte commands for efficiency
UART_Handler() {
    if (c == 'P') // Pause toggle
    if (c == 'R') // Reset
    if (c == 'S') // Start
    if (c >= '1' && c <= '8') // Level 1-8
}
```

DEVICE → AGENT (EVENTS)

```
{"event_type":"pop_result","mole_id":3,"outcome":"hit","reaction_ms":245,"lives":4,"lvl":2}
{"event_type":"session_start"}
{"event_type":"lvl_complete","lvl":3}
{"event_type":"session_end","win":true}
```

AGENT

- UART-MQTT relay
- Publishes device status (online/offline)
- Forwards events to dashboard
- Forwards commands to device
- QoS 2 for reliable delivery

```
~/projects/emb-whacamole/agent [main] > agent -s /dev/ttyACM0 -l DBG

[DBG] (01:43:50) :: Connecting to serial port /dev/ttyACM0 (115200 baud)
[INF] (01:43:50) :: Connected to /dev/ttyACM0
[DBG] (01:43:50) :: Requesting device ID
[INF] (01:43:50) :: Device ID received: 030d024afd
[DBG] (01:43:50) :: Last will set to offline
[DBG] (01:43:50) :: Connecting to MQTT broker alderaan.software-engineering.ie:1883
[INF] (01:43:50) :: Connected to alderaan.software-engineering.ie:1883
[DBG] (01:43:50) :: [Agent -> MQTT] {'device_id': '030d024afd', 'ts': 1766022230112, 'status': 'online'}
[DBG] (01:43:50) :: Subscribing to topic: whac/030d024afd/commands
[INF] (01:43:50) :: Subscribed to topic: whac/030d024afd/commands
[DBG] (01:43:50) :: Subscribing to topic: whac/all/commands
[INF] (01:43:50) :: Subscribed to topic: whac/all/commands
[DBG] (01:43:50) :: Listening for events
[DBG] (01:43:54) :: [Device -> MQTT] {'event_type': 'session_start', 'device_id': '030d024afd', 'ts': 1766022234695}
```

SYSTEM ROBUSTNESS

```python
def _read_events(self) -> None:
    """Read and process events from serial port."""

    self._log.debug("Listening for events")

    last_heartbeat = time.monotonic()
    while True:
        try:
            jsonl = self._serial_read_jsonl()

        except SerialException:
            if not self._device_connected():
                self._log.critical("Device unplugged, exiting")
            elif self._wait_for_reconnect():
                self._mqtt.publish_state("online").wait_for_publish()
                continue
            else:
                self._mqtt.publish_state("serial_error").wait_for_publish()
            return

        except (UnicodeDecodeError, JSONDecodeError):
            pass

        else:
            if jsonl is not None:
                self._mqtt.publish_event(jsonl)

        now = time.monotonic()
        if now - last_heartbeat >= HEARTBEAT_INTERVAL:
            self._mqtt.publish_state("online")
            last_heartbeat = now
```

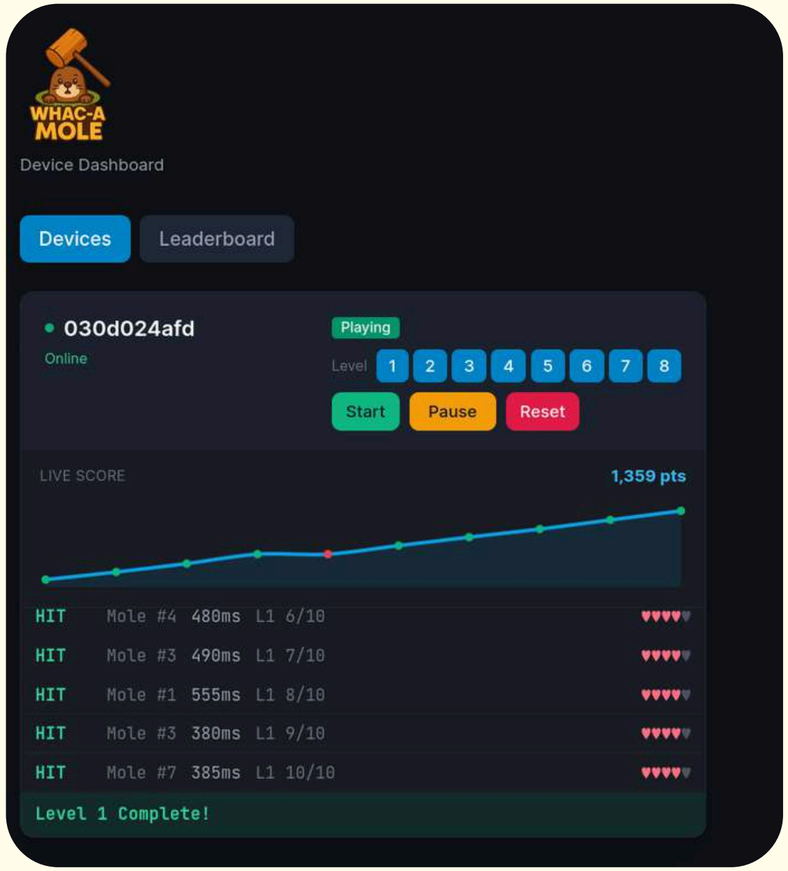
Device operates independently of agent connection:

- Game logic never blocks on communication
- Events queued when agent disconnected (100-event ring buffer)
- Automatic drain when agent reconnects
- No data loss during brief disconnections

Gracefully handles serial errors & attempts reconnections.

DASHBOARD

- lore
- ipsum
- lorem
- ipsum



LIVE DEMO!

THANK YOU!

Questions?