Figures

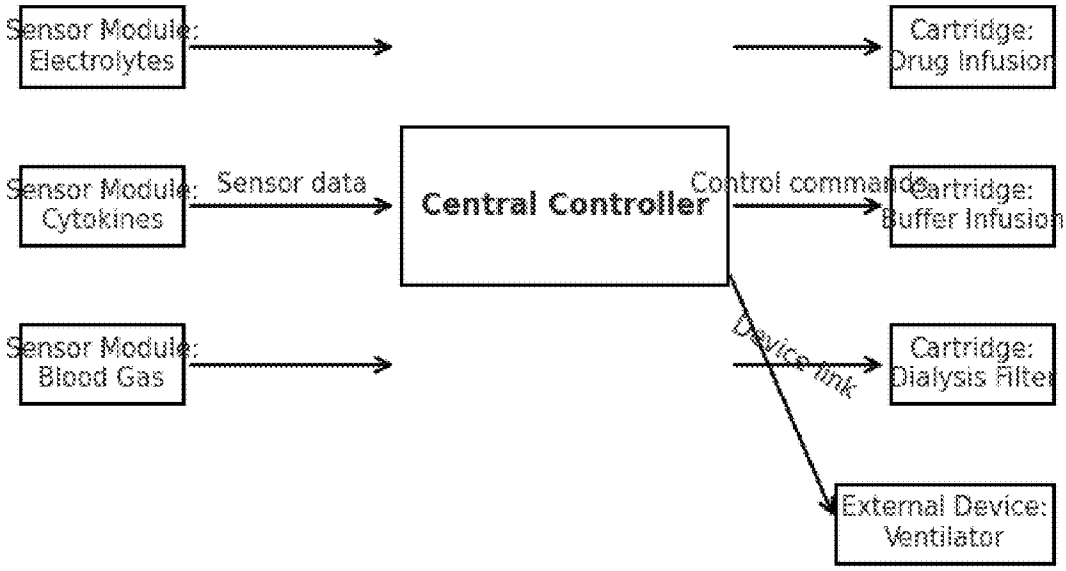

FIG. 1: System Architecture. This block diagram shows multiple **modular sensor modules** (e.g. wearable patches, probes) communicating via a redundant bus to a central **controller**, as well as **actuator cartridges** docked in bays and connected external devices (e.g. a ventilator). Arrows indicate data flow: sensor readings stream into the controller, and the controller sends control commands out to various pumps or actuators. This highlights the modular sensor/cartridge design and how sensor inputs and actuator outputs interface through the controller.

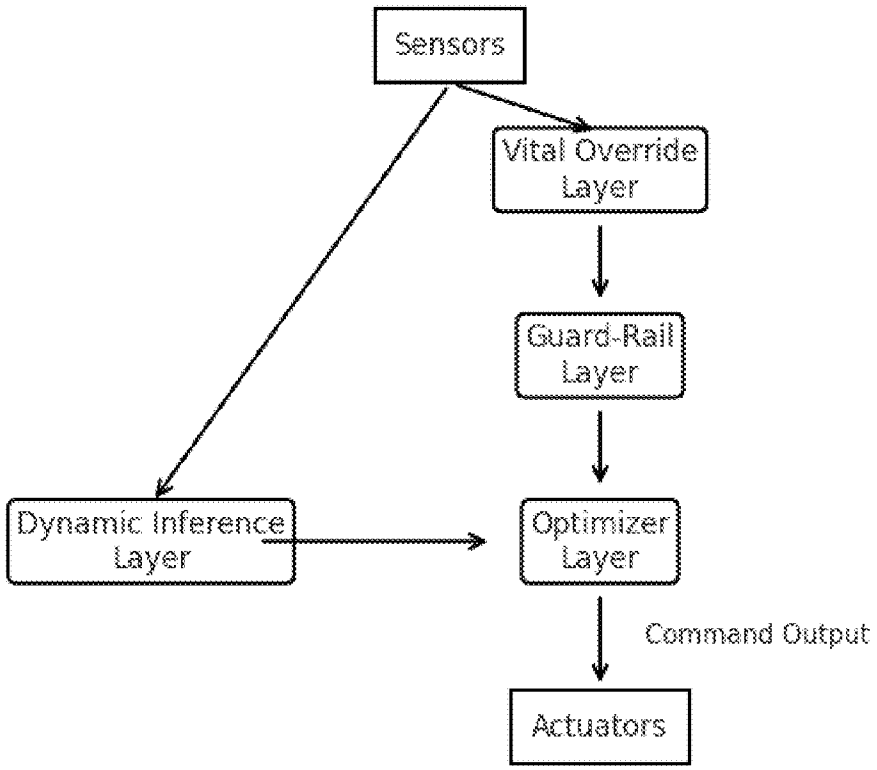

FIG. 2: Hierarchical Control Integration. This schematic illustrates the **multi-layer control logic**. A strict priority "ladder" of control tiers is shown: at the top, a **vital override layer** monitors for any life-threatening sensor readings and can immediately override or shut off outputs; below that, a **guard-rail layer** enforces interlocks and prevents unsafe actuator combinations; next, an **optimizer layer** (e.g. PID feedback controllers) makes routine adjustments; and finally a **dynamic inference (training) layer** learns and adapts parameters. Sensor data feeds into both the deterministic rule-based chain (vital → guard-rail → optimizer) and the adaptive model path. The **dynamic inference layer** interfaces with the optimizer – it receives sensor inputs and system state, and can subtly modulate the optimizer's output based on learned patient-specific relationships. All paths converge to a final command output that drives the actuators, with safety overrides always taking precedence.

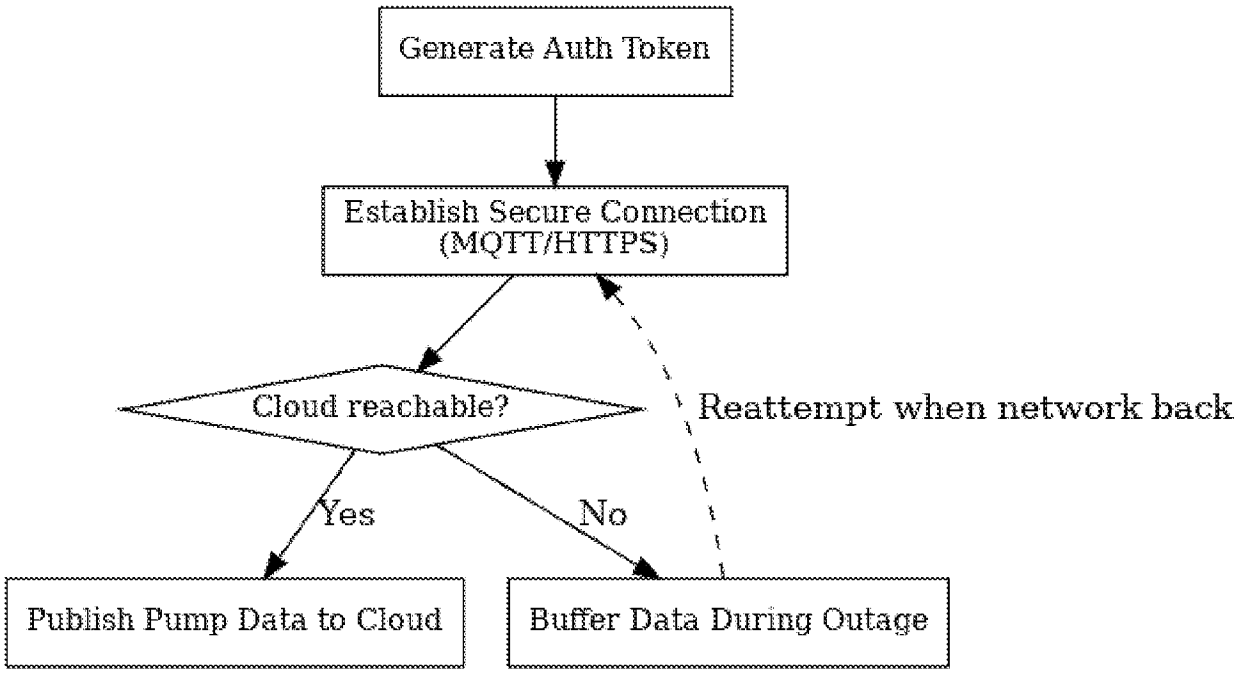

FIG. 2A: Dynamic Internal Model (Relational Graph). Sub-figure 2A depicts the controller's **adaptive neural relational model** as a graph of nodes and connections. Each node (circle) represents a monitored variable (e.g. glucose, CO_2, pH) or an actuator (e.g. insulin pump, ventilator, bicarbonate infusion), and edges indicate learned influence relationships between them. The dynamic inference layer employs a graph-based AI (such as a Graph Neural Network) that infers the strength of each connection – for example, how strongly a change in one actuator affects each sensor variable. (In the diagram, solid lines show primary influence links between actuators and variables, and a dashed line illustrates a discovered cross-coupling between two variables.) This learned graph model evolves over time as the system observes new interactions.

	Insulin	Ventilator	Bicarb
Glucose	-0.8	0.0	0.0
CO_2	0.0	-0.5	-0.1
pH	0.0	+0.3	+0.7

FIG. 2B: Jacobian Sensitivity Matrix. Sub-figure 2B shows an example **Jacobian matrix** representing the sensitivity of each variable to each actuator in the current learned model. Each cell (i,j) in the matrix (rows = variables like *Glucose*, *CO_2*, *pH*; columns = actuators such as *Insulin*, *Ventilator*, *Bicarb*) contains the partial derivative *∂*(variable$_i$)/*∂*(actuator$_j$), i.e. how a small change in a given actuator affects a given sensor reading. These values are continually refined by the dynamic learning layer: for instance, the matrix here shows that increasing the insulin infusion has a strong **negative** effect on glucose (–0.8), while ventilator adjustments mainly affect CO_2 (–0.5 on CO_2 row) and bicarbonate infusion strongly raises pH (+0.7). Such a Jacobian is updated in real time via micro-dose probe trials, gradually converging to the patient-specific sensitivities. **FIG. 2A** and **FIG. 2B** ***FIG. 2B: Jacobian Sensitivity Matrix.*** **Sub-figure 2B shows an example Jacobian matrix representing the sensitivity of each variable to each actuator in the current learned model. Each cell (i,j) in the matrix (rows = variables like *Glucose*, *CO_2*, *pH*; columns = actuators such as *Insulin*, *Ventilator*, *Bicarb*) contains the partial derivative *∂*(variable$_i$)/*∂*(actuator$_j$), i.e. how a small change in a given actuator affects a given sensor reading. These values are continually refined by the dynamic learning layer: for instance, the matrix here shows that increasing the insulin infusion has a strong negative effect on glucose (–0.8), while ventilator adjustments mainly affect CO_2 (–0.5 on CO_2 row) and bicarbonate infusion strongly raises pH (+0.7). Such a Jacobian is updated in real time via micro-dose probe trials, gradually converging to the patient-specific sensitivities. FIG. 2A and FIG. 2B**

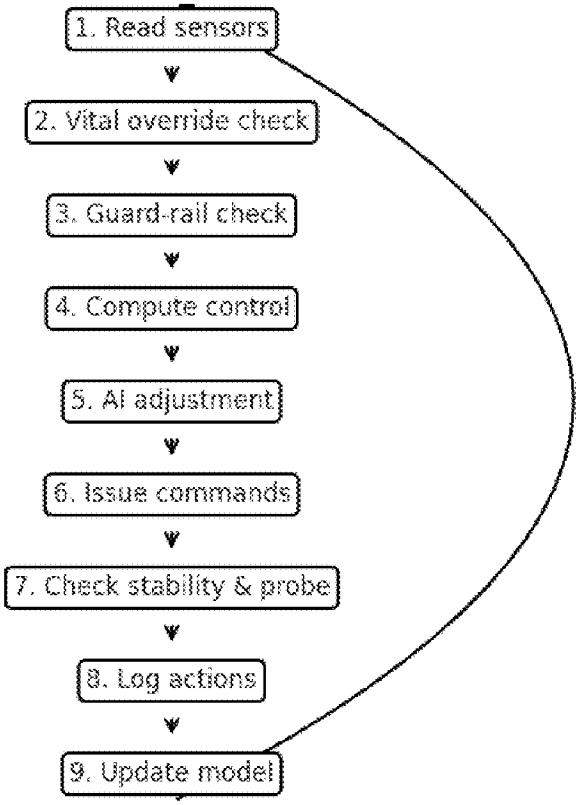

Figure 3 – Real-Time Inference and Control Flow

FIG. 3: Real-Time Control Loop Sequence. This flow diagram illustrates the system's continuous sense-decide-act cycle. **(1)** All sensor inputs are read and the current state is updated (hundreds of data points may be sampled). **(2)** The **vital override** rules are applied – if any reading violates a hard safety limit, the system immediately intervenes (e.g. stops a pump or triggers an alarm) and may skip the rest of the cycle. **(3)** If no override, the **guard-rail** layer checks for any unsafe actuator combinations or dose limits, blocking or adjusting commands that would violate safety constraints. **(4)**The **optimizer** then computes new control outputs for each actuator (e.g. PID adjustments) using the latest sensor data. **(5)** The **adaptive AI** module refines these outputs – it may scale an output down if the model predicts the patient is very sensitive to it, or add a preemptive adjustment if it foresees a trend, as long as this remains within safe bounds. **(6)** The final command set is then issued to all cartridges and external devices (pumps, valves, etc.), effecting the therapeutic actions. **(7)** If conditions are stable (all variables within their target μ-bands), the system injects a tiny **probe** dose on one actuator to test

the response. This controlled perturbation is used to gather learning data. **(8)** All actions and results of the cycle (sensor readings, decisions, overrides, probe outcomes) are **logged** to a secure audit trail. **(9)** Finally, the adaptive model is **updated** with any new data – the measured sensor change from a probe trial (ΔX) is used to refine the corresponding Jacobian parameter, and even normal control actions are fed into the learning process. This loop then repeats continuously (e.g. every 5 seconds), ensuring real-time closed-loop control with ongoing learning.

FIG. 4: Safety Failsafe States. This diagram illustrates the system's layered fallback modes and safety overrides. In normal operation, the controller actively regulates therapy. If a critical threshold is exceeded or a fault is detected, the system escalates to **failsafe modes**: first entering a **Safe-Monitor mode** where active dosing is halted and the system only monitors the patient (maintaining critical functions). If the condition persists or a system fault occurs, the system may transition to a **Maintenance Drip mode**, in which a minimal baseline infusion or bypass is applied to keep the patient stable without dynamic adjustments. At any point, a clinician can activate **Manual Override**, the highest level of control, which returns the system to purely manual operation by the user. Transitions between these states are indicated by arrows in the diagram: e.g. a *"Critical event"* triggers Normal → Safe-Monitor, and a persistent issue triggers Safe-Monitor → Maintenance Drip, while *"Stability restored"* will automatically return the system from Safe or Maintenance back down to Normal once the patient's condition is safe again. The figure also highlights the built-in hardware redundancy: a **watchdog-monitored primary/backup controller** setup. The primary controller's heartbeat is monitored, and on any failure the watchdog triggers an immediate **failover** to the backup controller to ensure continuity. **FIG. 4**

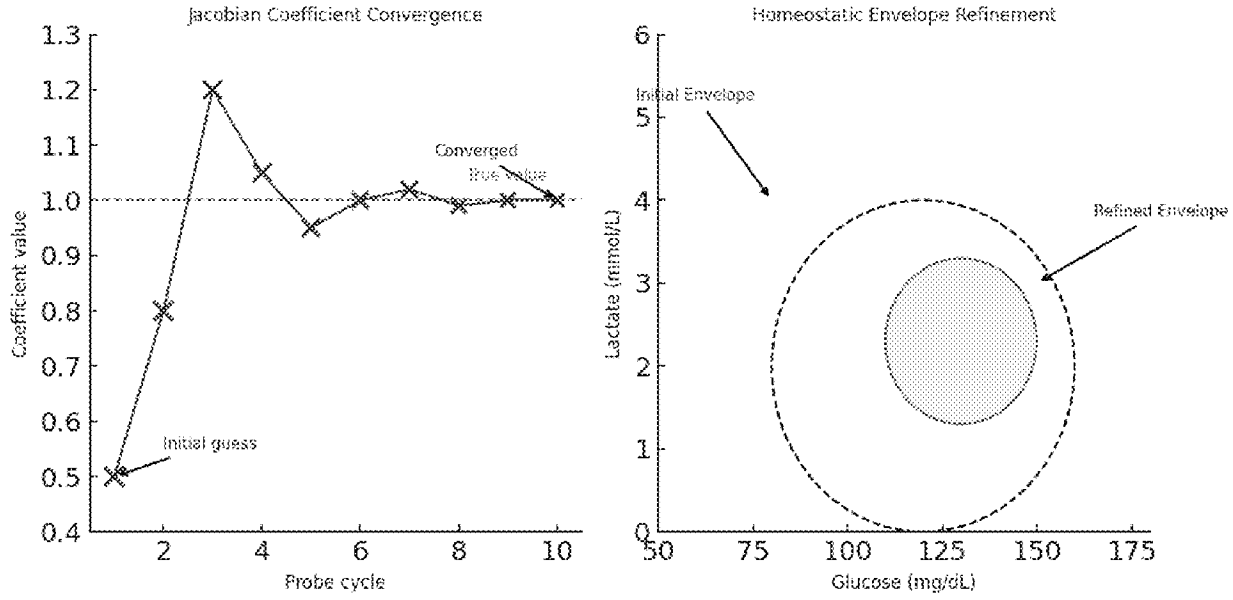

FIG. 5: Adaptive Learning Convergence. This figure illustrates how the adaptive model learns an individualized "homeostatic envelope" over time. **Left plot:** an example **Jacobian coefficient** (sensitivity of a particular variable to an actuator) converging with successive probe trials. The blue curve shows the coefficient estimate gradually approaching the true value (dashed line) as the controller performs iterative micro-dose probes. Initially, the estimate is far off, but with each safe probe and update, it oscillates closer to the true sensitivity, eventually converging within a narrow confidence band. (Probe step sizes decrease as confidence grows – early probes caused larger adjustments, while later probes refine with smaller tweaks, as indicated by the diminishing marker sizes.) **Right plot:** the system's learned **homeostatic envelope** in two dimensions (e.g. glucose vs. lactate levels). The dashed circle represents the initial, broad safe range based on generic population data, whereas the shaded region represents the **refined envelope** tailored to this patient. As the controller gathers data, it discovers that the patient remains stable only within a tighter range of glucose and lactate. The model continually updates this envelope – shrinking and shifting it – to encompass the combinations of variables where the patient's physiology stays in balance. This visualizes how the adaptive layer homes in on the patient-specific stability zone beyond the default settings. **FIG. 5**