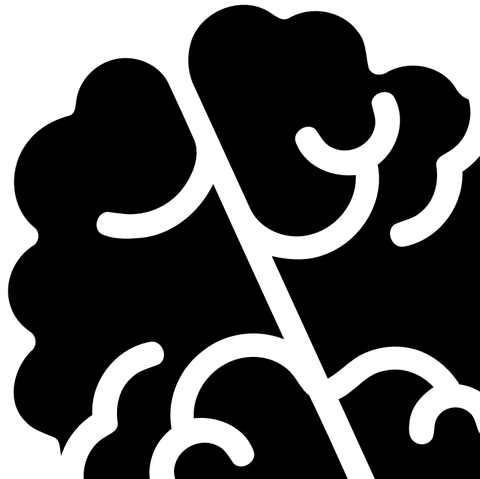
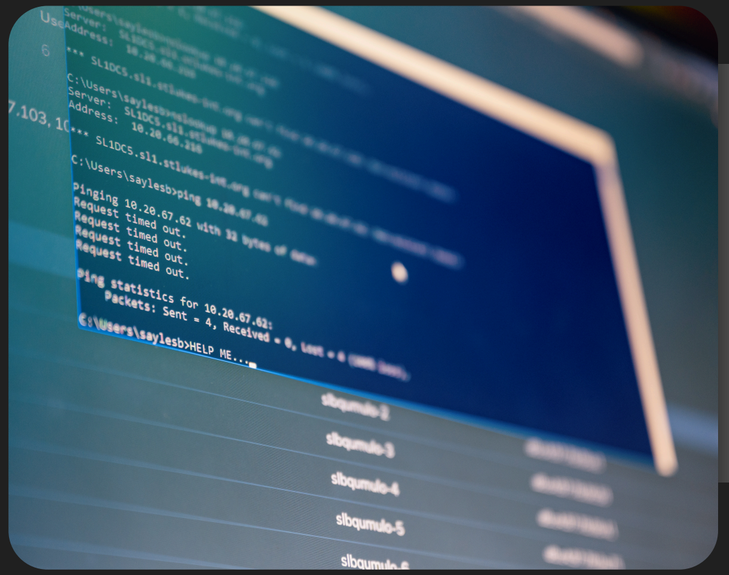
Kaggle competition

Pressure Prediction in Mechanical Ventilators Using Neural Networks

Team 13
Juan Felipe Gonzalez - 20222020200
Joel David Pérez Arroyave - 20242020017
Cristian David Parroquiano Jimenez - 20222020192
Santiago Chavarro - 20231020219



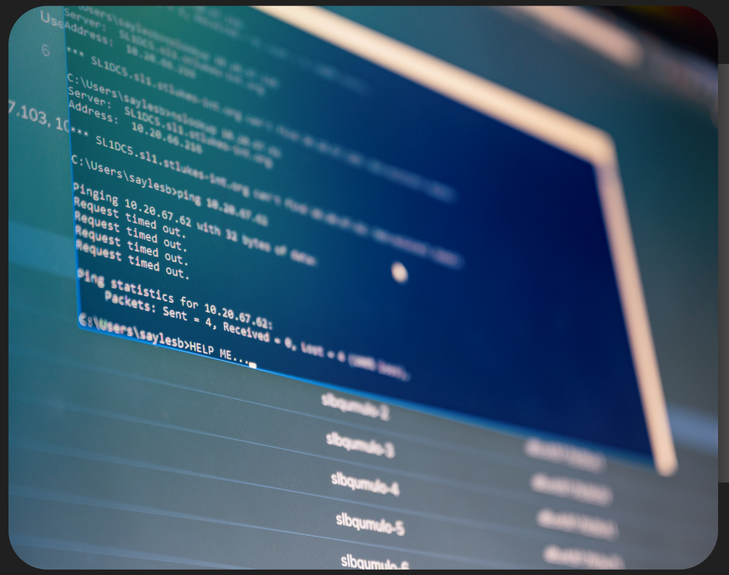
Overview



Layers of the system

- Hybrid system for predicting mechanical ventilator pressure
- Uses LSTM neural networks, physic equiations and data driven simulations
- Processes real time data via sensor through a multi layer architecture
- Ensures safe, robust and physiologically consistent predictions

Goals & Motivation



Main needs and objectives

- Need for accurate pressure control to avoid lung injury
- LSTM learns from temporal breathing patterns, cellular automata models the spatial pressure behaviour
- The main objective is to build a full pipeline including data ingestion, ML prediction and safety control

System Architecture

five core modules

DATA INGESTION

PREPROCESSING

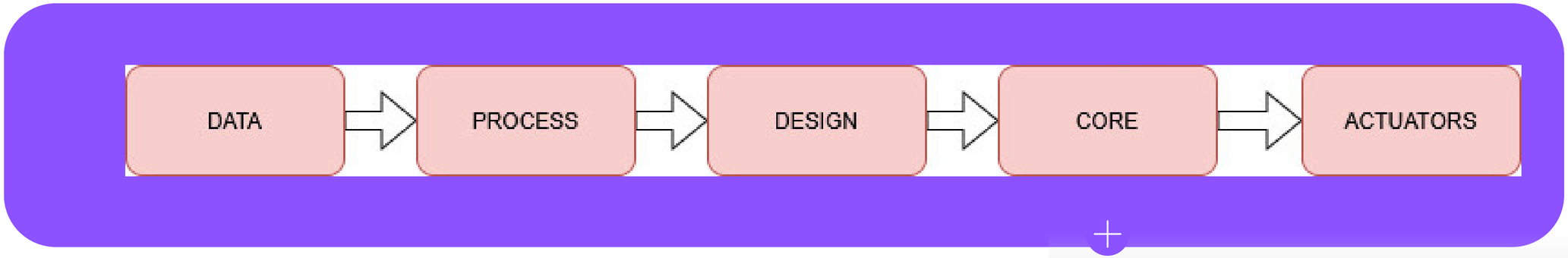
SOCKET COMMUNICATION

LSTM +PHYSICIAL MODEL

SAFETY CONTROLLER

ACTUATION LAYER

MONITORING & VISUALIZATION



Key Components

system relevant parts

ML LAYER

The layer working in the LSTM model prediction, using data driven simulations and analytical equiation for consistency

ACTUATION LAYER

The actuation layer is all the parts that are tasked with the operating of the system, which drives the ventilator valves

PREPROCESSING

it is the phase that its task is to filter, normalize and slide the different events of the system and data that goes into the system

SAFETY CONTROLLER

The safety controller is the phase where the system verifies the predictions before acting and provide the pressure

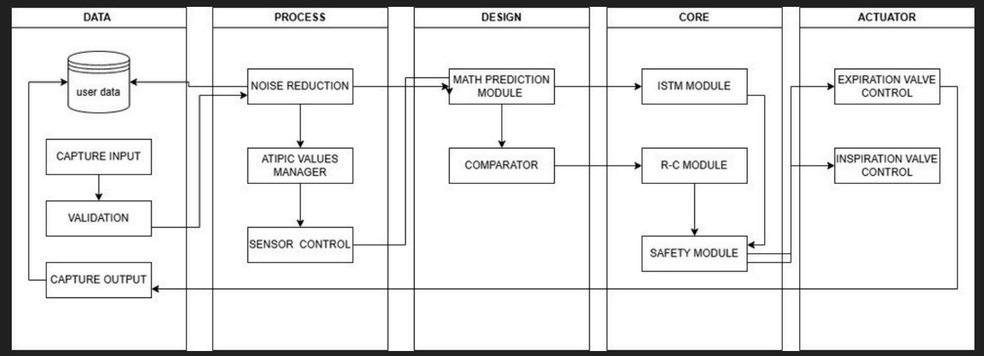
MONITORING

It is the auto regulation phase where the system recolects data and information and transform it into real time metrics and alerts

System components and behavior

components diagram

We decided to use a monolithic architecture to organize the system behavior into 5 coordinated layers achieving, this design objectives through integrated functionality. where the information is cached from the enviroment and user, save on a database and read by LSTM modules,



DATA LAYER:
caputure and validate

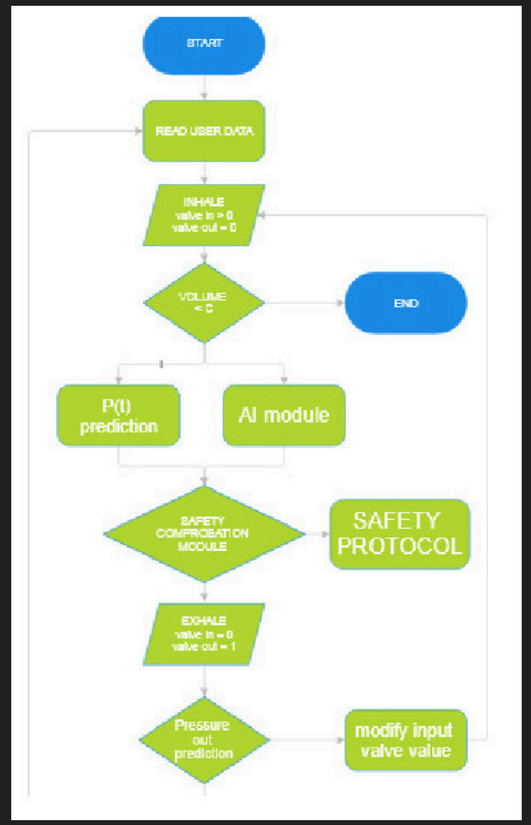
PROCESS LAYER
transform raw signals into trusted information

DESIGN LAYER
feed the prediction module with information and compare it

CORE LAYER
AI modules read the information and takes decisions based on user state

ACTUATOR LAYER
transform information into signals for the hardware, also reads information from user and complete the feedback loop

This flow diagram provides a clear overview of the process. It highlights the key stages and how information moves from one step to the next, helping us understand dependencies and decision points, also define the feedback loop



flux diagram

system sensibility and Chaos

Air and patient attributes are the most sensitive factors → they require real-time validation.

Input	Sensibility	Impact
Air quality	0.9	Patient survival, purity of flow
Physical attributes	0.9	Anatomical variability, personalized requirements
Medical conditions	0.8	Diseases such as COPD/ARDS, response to treatment
Manual operation	0.6	Variabilidad humana, mitigada por automatización
Environmental particles	0.6	Filtrado multi-etapa, bajo riesgo controlado

system sensibility and Chaos

Strategies implemented

- Real-time validation
- Dynamic calibration per patient
- Hybrid LSTM + Physics model
- Independent safety controller

system sensibility and Chaos

Chaotic phenomena identified

- Dependence on initial conditions
- Abrupt phase transitions
- Pulmonary hysteresis
- Limitations of the traditional R-C model

Control mechanisms implemented

- Finite state machines with hysteresis compensation
- Robust initialization
- Bounded output constraints
- LSTM for nonlinear dynamics:

Technologies and implementation

Training Dataset

- 10,000 records - Synthetic breathing cycles
- Main variables: R (resistance), C (compliance), uin (inlet valve), uout (outlet valve)
- 125 breathing cycles for complete training

Simulation: Data-Driven LSTM Model

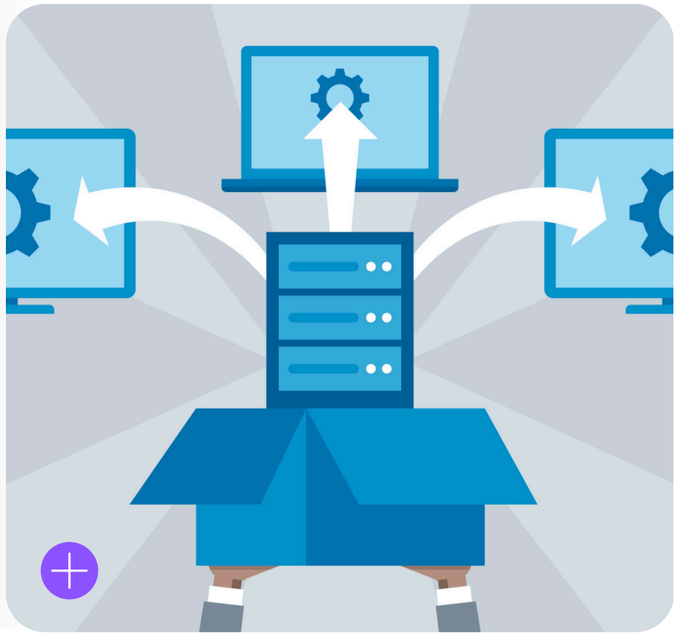
Frontend (JavaScript)

- app.jsx loads dataset
- Calculates theoretical values
- Displays predictions

Backend (Python)

- predict.py con PyTorch
- LSTM entrenado 125 ciclos
- Inferencia en datos nuevos

Production Deployment



- Docker - Component Images
- Kubernetes - Automatic Orchestration
- Non-Disruptive Updates
- Hardware Integration
- ROS2 - Hardware Communication
- CANopen/IEEE 11073 - Protocols

coding highlights

Salford & Co.

Machine learning implementation

model.py

- Loads trained model and scaler
- prepares sequences grouped by breath
- Runs inference and outputs predicted pressures

```python
class VentilatorModel:
    def __init__(self, model_type='fast'):
        """
        model_type: 'fast' (RandomForest) o 'accurate' (GradientBoosting)
        """
        if model_type == 'fast':
            self.model = RandomForestRegressor(
                n_estimators=100,
                max_depth=15,
                random_state=42,
                n_jobs=-1,
                verbose=1
            )
        else:
            self.model = GradientBoostingRegressor(
                n_estimators=100,
                learning_rate=0.1,
                max_depth=5,
                random_state=42,
                verbose=1
            )
```

```javascript
const avgNeighbor = neighborSum / count;
let newPressure = cell;

if (cell === 2 && avgNeighbor < 1.5) {
  newPressure = 1;
} else if (cell === 0 && avgNeighbor > 1.0) {
  newPressure = 1;
} else if (cell === 1) {
  if (avgNeighbor > 1.5) newPressure = 2;
  else if (avgNeighbor < 0.8) newPressure = 0;
}

if (Math.random() < 0.05) {
  newPressure = Math.floor(Math.random() * 3);
}

return newPressure;
```

Cellular automata

- grid with 3 pressure states
- State updates based on neighbor average
- Driven by LSTM prediction to simulate spatial dynamics

coding highlights

Salford & Co.

Visual integration

app.jsx

- Loads csv and computes theoretical pressure
- sends sequences to backend lstm predictor
- plots actual vs predicted pressure curves

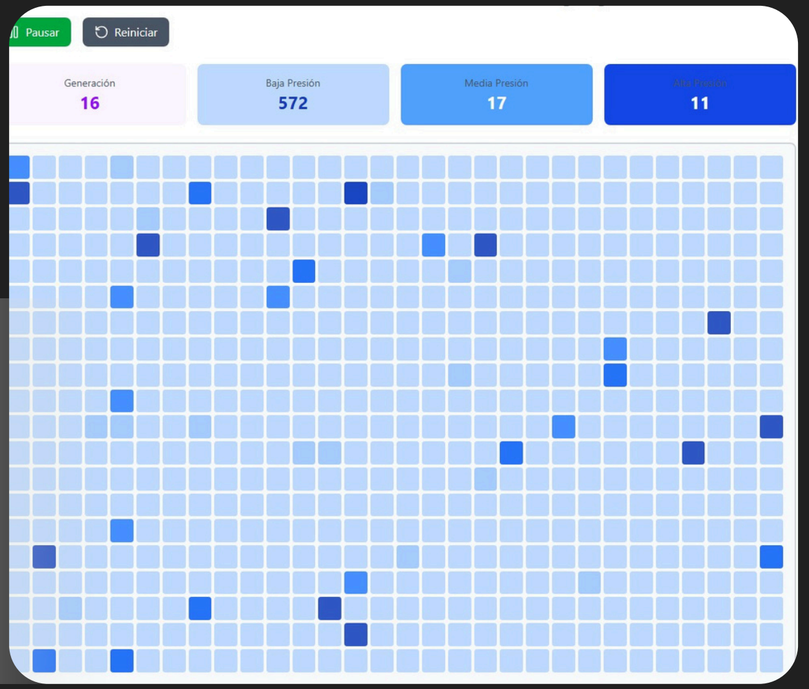
```
// Dataset State
const [datasetLoaded, setDatasetLoaded] = useState(false);
const [datasetInfo, setDatasetInfo] = useState(null);
const [rawData, setRawData] = useState([]);
const [processedData, setProcessedData] = useState([]);
// Test Dataset State
const [testDatasetLoaded, setTestDatasetLoaded] = useState(false);
const [testDatasetInfo, setTestDatasetInfo] = useState(null);
const [testProcessedData, setTestProcessedData] = useState([]);

// ML Simulation State
const [mlRunning, setMlRunning] = useState(false);
const [mlTesting, setMlTesting] = useState(false);
const [mlData, setMlData] = useState([]);
const [mlEpoch, setMlEpoch] = useState(0);
const [mlMetrics, setMlMetrics] = useState({ mae: 0, rmse: 0 });
const [testMetrics, setTestMetrics] = useState({ mae: 0, rmse: 0, samples: 0 });
const [currentBreath, setCurrentBreath] = useState(0);
const [trainedModel, setTrainedModel] = useState(null);
const [testRawData, setTestRawData] = useState([]);
const [predictions, setPredictions] = useState([]);

// Breath parameters
const [breathParams, setBreathParams] = useState({ R: 20, C: 50 });
```

Integration

- Full pipeline from input data to LSTM output to Cellular automata simulation and the graphic visualization
- an xy graph to visualize the behavior of the LSTM prediction



celular Auntomata

Celular Automata graph the emergent behaviors during the training and run the AI module, each cell defines a section from the user lungs

complete usable interface

Finally we have a complete User interface where we can train the AI module and verify the result through a graphic interface, also we can compare the results between the mathematical predictor and the the AI module using a MAC

time plot shows the pressure that the AI module decided to bring to the user (red curve) and the math predictor that trains the AI (blue curve)

Neural Network training

final code and simulation

Thank You