The Problem

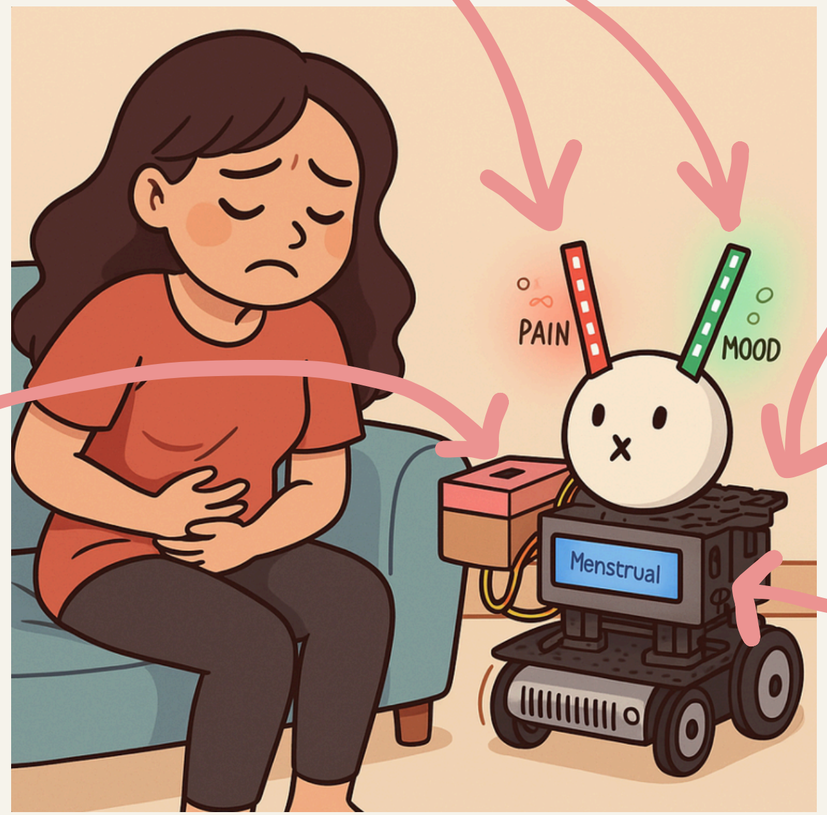
Support during menstruation

- During menstruation, many users experience fatigue, discomfort and pain
- Limited mobility can make routine tasks harder
- Emotional and physical symptoms fluctuate during the cycle
- There are gaps in personalised care and reminders

Overview of Our Solution

LED bars used to display the pain and mood tracking

buttons to input pain and mood levels, and also track menstruation beginning and end



platform where the user can load up items e.g. painkillers, pads etc

display screen that shows the user which part of their cycle they're on (i.e. menstrual, ovulation etc)

System Architecture

Layer	Components
Input Layer	• Physical Buttons • Web Dashboard Interface (Node-RED)
Communication Middleware	• MQTT Message Broker (Topic Routing & Message Distribution)
Control & Processing Layer	• Sensor Control (Grovepi) • Robot Control • Data Processing
Execution & Feedback Layer	• LED Display System (Mood/Pain/Cycle) • TurtleBot Mobile Platform • LCD

System Architecture

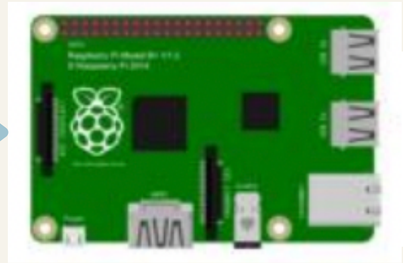
MQTT Client

pain/mood levels or period start/end

publish to topic

MQTT Broker on RPi

implemented by mosquitto

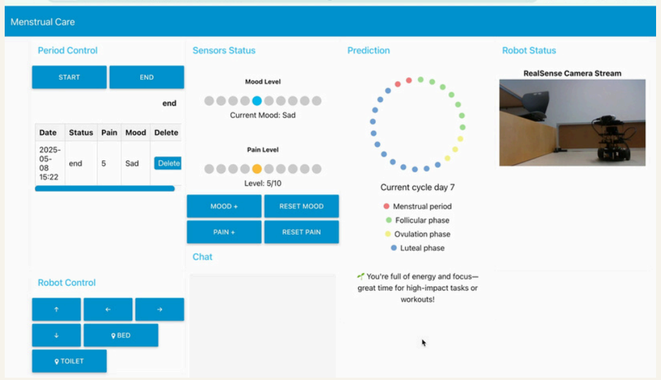


publish to topic

subscribe to topic

MQTT Client

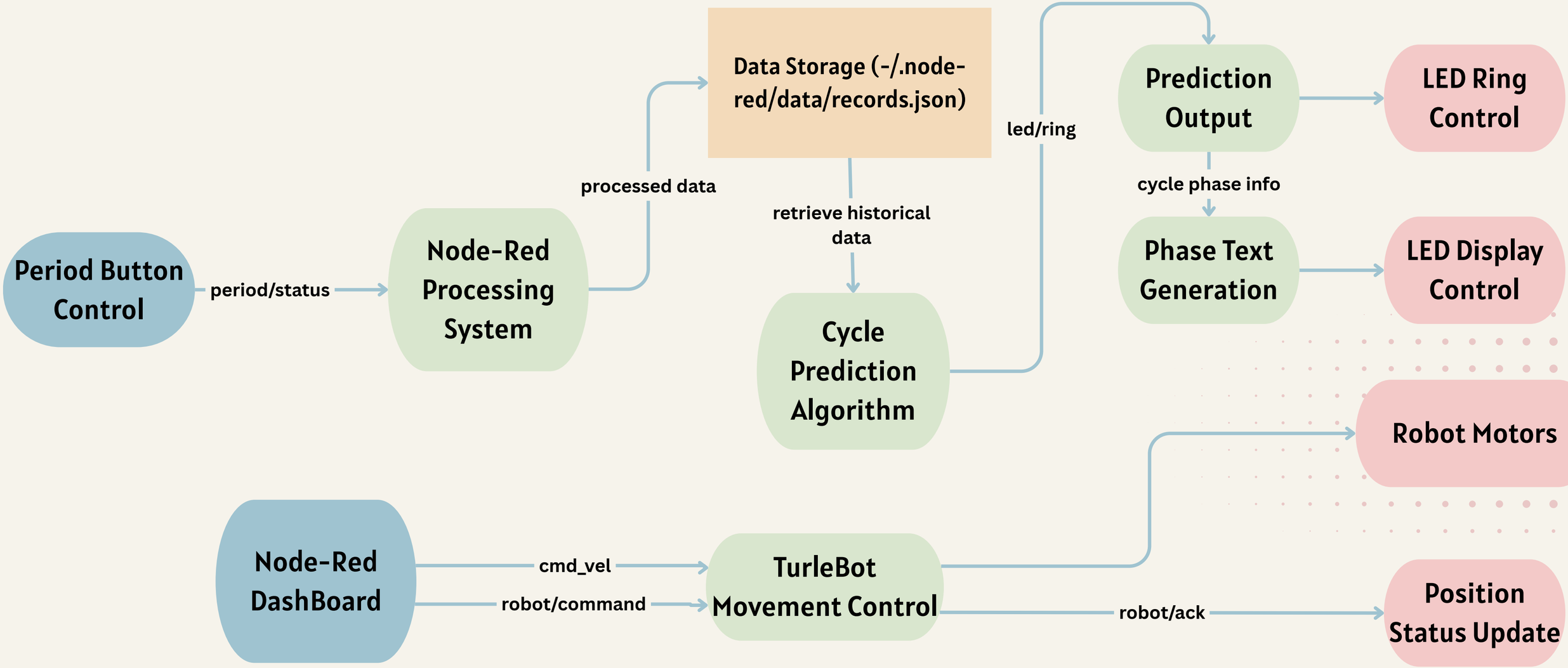
data visualisation dashboard



depending on the button pressed (i.e. period start, period end, mood, pain) , a different MQTT topic will be published/suscribed to

button	topic	button	topic
mood	sensor/mood	**start**	period/status
pain	sensor/pain	**end**	period/status

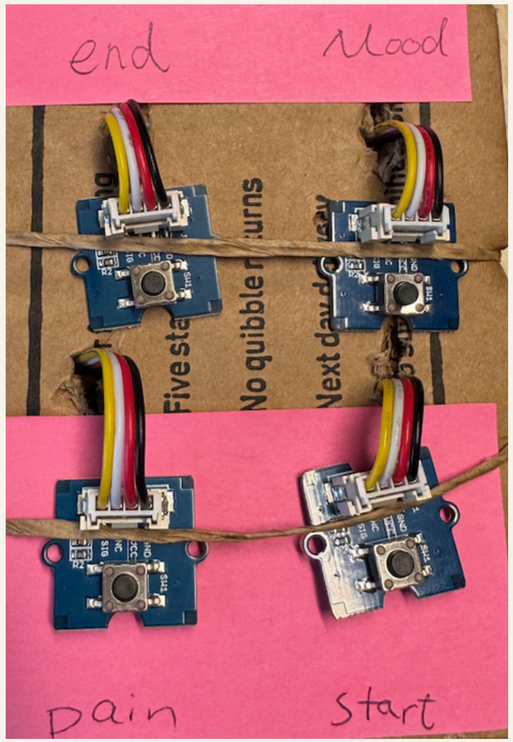
Flow Diagram



SENSORS

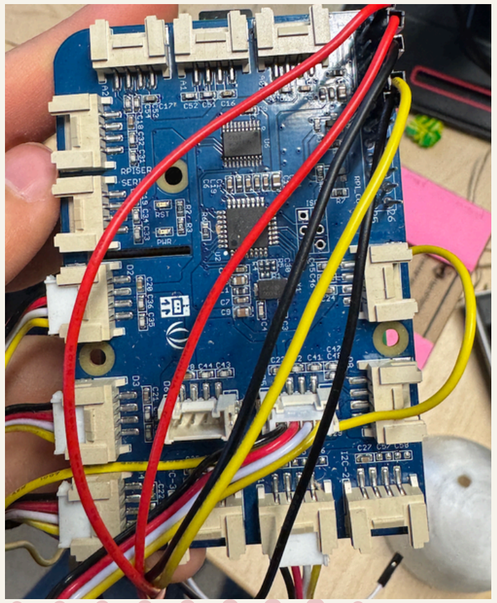
Type of sensor used

Button - we used 4 in total



Connection Ports

- D2/3 for period start and end buttons
- D4/5 for mood and pain buttons



Functions

- Records the start /end date of menstrual period..
- ..or records the mood/pain levels of user

Sensor Implementation

Implementation

- Monitors button state through `button_control.py`
- sends "start"/"end" message to MQTT topic "period/status" when pressed

```
# MQTT Configuration
MQTT_BROKER = "127.0.0.1"
MQTT_PORT = 1883
MQTT_TOPIC = "period/status"
```

```
def publish_button_press(self, button_type):
    """Publish button press event"""
    try:
        # Create message payload
        payload = button_type

        # Publish message
        result = self.client.publish(MQTT_TOPIC, payload)
        if result.rc == mqtt.MQTT_ERR_SUCCESS:
            print(f"\n{datetime.now().strftime('%Y-%m-%d %H:%M:%S')} - {button_type} button pressed")
        else:
            print(f"Failed to publish message, error code: {result.rc}")
    except Exception as e:
        print(f"Error publishing message: {e}")
```

```
def run(self):
    """Main simulation loop"""
    print("\nButton Simulator Running...")
    print("Commands:")
    print("'s' - Simulate Start Button")
    print("'e' - Simulate End Button")
    print("'q' - Quit")
    print("\nEnter command: ")

    try:
        while True:
            command = input().lower().strip()

            if command == 's':
                self.publish_button_press("start")
            elif command == 'e':
                self.publish_button_press("end")
            elif command == 'q':
                print("\nExiting simulator...")
                break
            else:
                print("\nInvalid command. Use 's', 'e', or 'q'")

            print("\nEnter command: ")
```

- The same is done for monitoring pain and mood levels, through s_moodAndPain.py

SENSOR – CHOICE JUSTIFICATION

Why we chose to use buttons

1. Offers a simple and intuitive physical interface, allowing users to easily log period start/end or mood/pain levels

2. Addresses a key issue identified in our questionnaire: most tracking apps involve multiple steps unlocking, navigating, and selecting. This makes consistent use more difficult, especially on rough days

3. One-button operation removes friction from data entry, significantly lowering the barrier to regular, accurate tracking

Sensor Integration

1

Sensor Setup

Sensors were connected via GrovePi+ on the Raspberry Pi, including buttons for period, mood, and pain tracking. Python scripts using the GrovePi library handled data collection.

2

Data Processing & Communication

Node-RED used MQTT to receive sensor data, process it, and visualise it on a dashboard. Data like mood, pain, and period status was stored locally in JSON format.

3

Integration Challenges

LED components had to bypass the GrovePi+ and connect directly to GPIO pins. Limited power from the Raspberry Pi prevented simultaneous use of both the LED strip and ring.

ACTUATORS

We used four different actuators:

- Mood LED Strip
- Pain LED Strip
- Cycle Prediction LED Ring
- Grove LCD RGB Backlight Display

ACTUATOR – LED STRIP

Mood LED Strip

Connection Port: GPIO 18 (PWM)
Number of LEDs: 10
Function:

- Visually displays current mood state level
- 10 LEDs correspond to 10 mood states (Calm, Happy, Excited, Anxious, Sad, Tired, Irritated, Angry, Depressed, Stressed)

Why did we select an LED strip?

- Provides intuitive visual feedback
- Green represents mood, aligning with psychological concepts of green representing balance and harmony

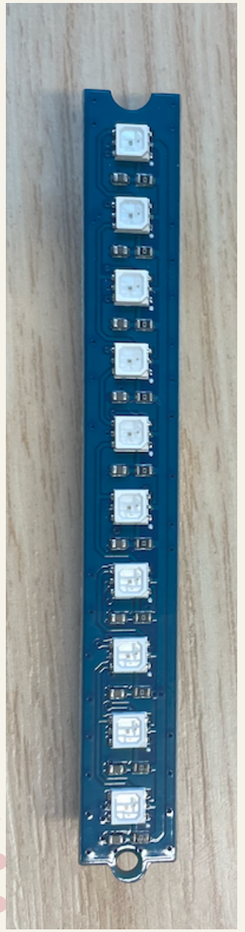
Pain LED Strip

Connection Port: GPIO 19 (PWM)
Number of LEDs: 10
Function:

- Visually displays current pain level
- 10 LEDs correspond to 10 pain levels

Why did we select an LED strip?

- Provides intuitive visual feedback
- Red represents pain, aligning with intuitive understanding
- Addresses the difficulty users have in recalling and accurately describing pain levels, helping users assess pain more objectively through visual scaling



ACTUATOR – LED RING

Cycle Prediction LED Ring

Connection Port: GPIO 21 (PWM)
Model: Grove - RGB LED Ring (20 - WS2813 Mini)
Number of LEDs: 22
Function:

- **Visually displays menstrual cycle prediction and current cycle phase using different colours to represent the different phases**

Color Coding:

- **Red: Menstrual phase, Green: Follicular phase, Yellow: Ovulation phase, Blue: Luteal phase**

Why did we select an LED ring?

- **Ring design symbolises cyclical nature**
- **Multi-color LEDs intuitively distinguish different cycle phases**
- **Solves the problem of traditional apps requiring opening and navigating to prediction pages; users only need to glance at the device to understand their cycle status**

ACTUATOR – BACKLIGHT DISPLAY

Cycle Phase Display

Connection: I2C bus
Display text address: 0x3e
RGB backlight address: 0x30

Function:

- **Displays the current menstrual cycle phase name**

Why did we select a backlight display?

- **Provides clear textual feedback to communicate the current cycle phase without needing an app**
- **Improves accessibility for users who prefer or rely on text over visual indicators like LEDs**
- **Backlight ensures visibility in different lighting conditions, enhancing usability**
- **I2C connection simplifies wiring and reduces GPIO pin usage**
- **Helps users understand their cycle phases through straightforward, readable labels**

ACTUATOR IMPLEMENTATION

LED Strips

- Controlled using the `rpi_ws281x` library
- Displays green/red LEDs to indicate mood/pain levels

Cycle Prediction LED Ring

- Controlled through `predict_ring.py`
- Displays different colors based on cycle prediction data calculated by Node-RED

Cycle Phase Display

- Controlled through `backlight.py`
- Updates display content and backlight color based on cycle phase

Real-Time Monitoring & Decision Making

The system implements a comprehensive real-time monitoring architecture through multiple layers:

1. **Physical Input Layer**
2. **Data Processing Layer**
3. **Feedback Layer**
4. **Multi-Sensor Decision Making**

Real-Time Monitoring & Decision Making

Physical Input Layer:

- Period tracking through start/end buttons (button_control.py)
- Mood levels through 10-state button input (moodAndpain.py)
- Pain levels through 10-state button input (moodAndpain.py)

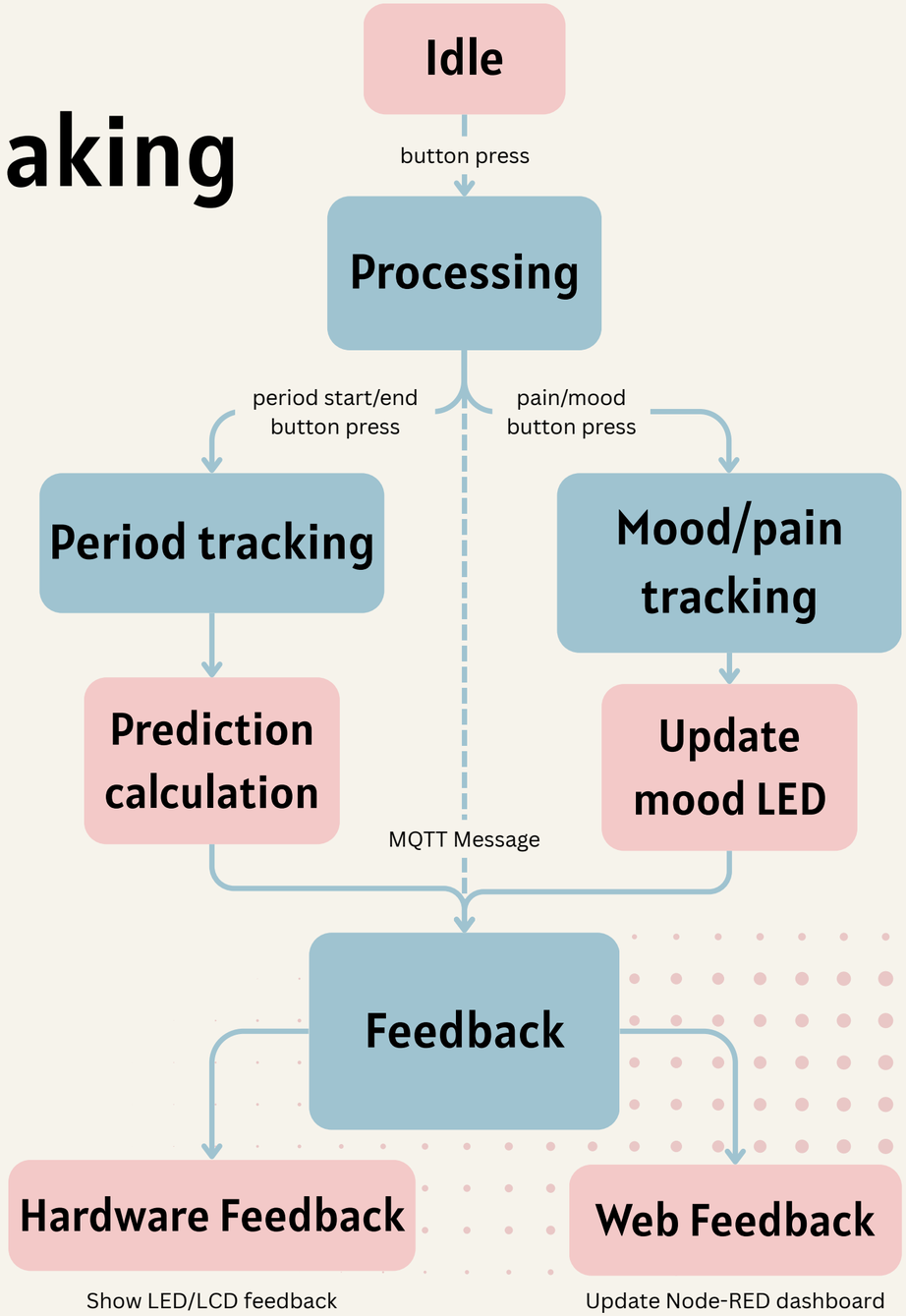
Data Processing Layer:

- MQTT-based real-time message broker for instant data transmission
- Node-RED flows for immediate data processing and state management
- Synchronous LED feedback for user input confirmation

Real-Time Monitoring & Decision Making

Feedback layer

- Provides real-time visual confirmation of inputs using MQTT-driven state machine logic, ensuring fast and reliable user feedback.
- Synchronises outputs across hardware (LEDs, LCD) and the web dashboard (Node-RED) for consistent multi-channel display.
- Coordinates input processing and display updates through a structured state machine, maintaining platform-wide state accuracy.



Real-Time Monitoring & Decision Making

Multi-Sensor Decision Making:

1. Period Phase Determination:

- Uses period start/end button inputs
- Determines cycle phase (Menstrual, Follicular, Ovulation, Luteal)
- Updates LED ring and LCD display accordingly

2. User State Monitoring:

- Combines mood and pain inputs
- Provides visual feedback through multiple LED indicators
- Synchronises state across all display components

3. Predictive System:

- Uses historical period data
- Calculates cycle predictions
- Updates 20-LED ring display with phase predictions
- Colour codes different phases for intuitive understanding

Challenges Faced

Limited Hardware Access

Hardware was only available in the lab, which had restricted opening hours, especially during the Easter break when team members were off campus. This made it harder and slower to test hardware-dependent features.

Integration & Compatibility Challenges

Updating Node-RED and supporting libraries on the Raspberry Pi was difficult, limiting some feature implementations. While we explored integrating a custom Node.js + Express + SQLite platform, deployment complexity led us to use Node-RED's built-in Dashboard for a more stable and maintainable interface.

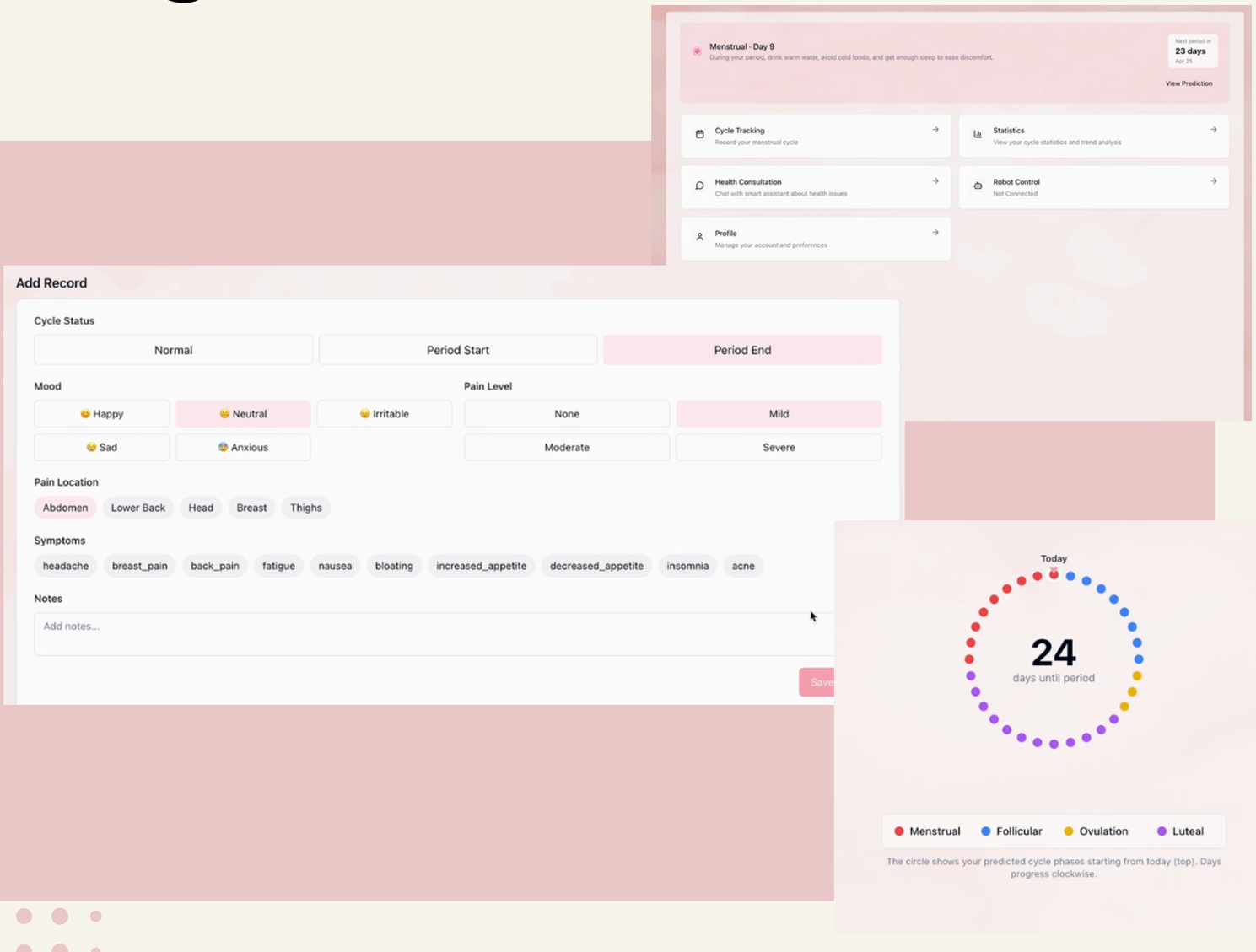
Database & Prediction Constraints

We initially considered a cloud database to support larger datasets and enable ML-based predictions, but compatibility issues and setup complexity made this unfeasible. We used a local database with simple averaging for predictions, and UI limitations restricted visible data to the last 10 entries.

Challenges Faced

Limited Hardware Access

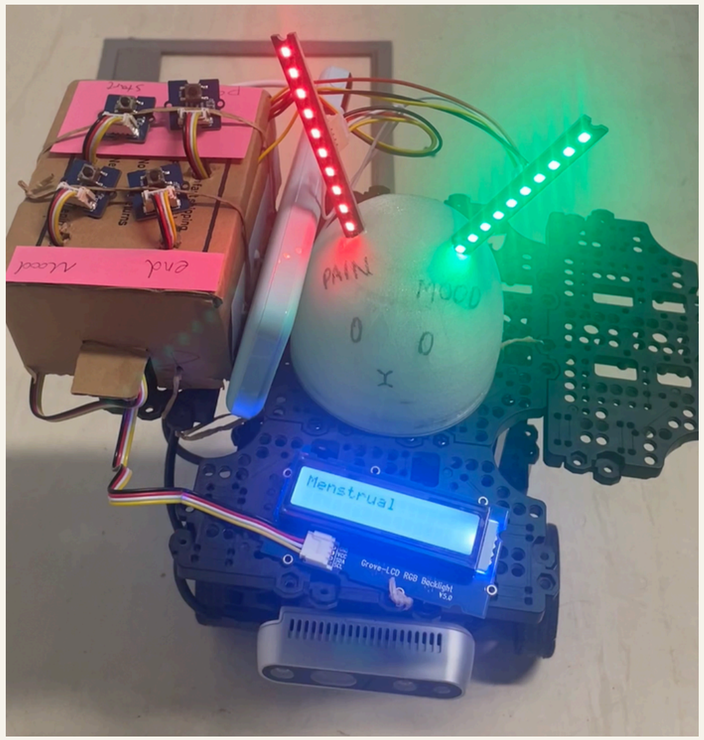
- we initially tried to create our user interface using devbox, however it required a subscription
- the code for the website was also too complicated to integrate with our hardware so we instead created our UI using node red



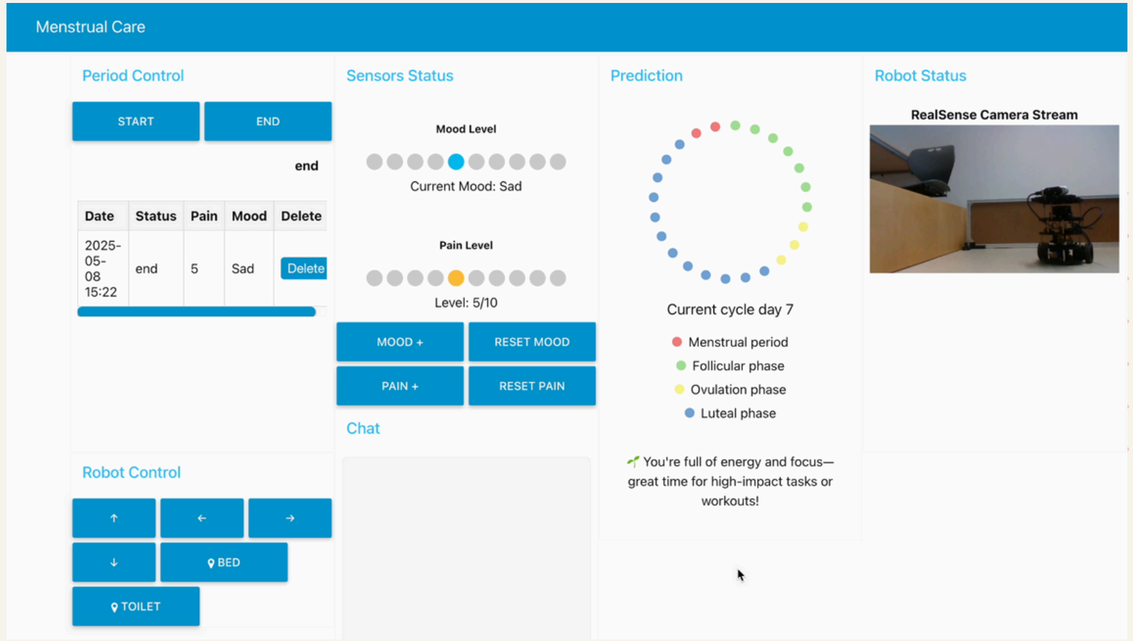
RESULT

Flobot

3d printed bunny cover



User Interface



Key Learnings

User-Centered Development & System Thinking

Throughout this project, we developed both technical and system-level thinking skills. We learned to design with the user in mind, focusing on intuitive interaction and meaningful feedback rather than just technical implementation.

Sensor Integration & Communication Design

Working with multiple inputs required us to calibrate sensors, handle noisy data, and standardise formats to ensure smooth communication across components. Real-time coordination using MQTT taught us how to manage asynchronous data and keep all system parts in sync.

Hardware Constraints & Environmental Awareness

Limited hardware access and real-world constraints like sensor placement and terrain helped us appreciate the physical challenges of robotics and the importance of testing in realistic conditions. Overall, we shifted from a bug-fixing mindset to proactive system design, emphasising reliability, clarity, and user experience.

Future Improvements

Voice Interaction Integration

- Add voice input for natural symptom logging
- Automate data capture from speech to improve ease-of-use

Additional Functions

- Add heating pad module for comfort
- Add internal storage for menstrual products or supplies
- Implement one-touch emergency contact feature
- Create a companion smart wristband to monitor vitals and provide reminders

Conclusion

FloBot integrates real-time tracking, intuitive interaction, and predictive insights to support menstrual health in a meaningful way. By combining multiple sensors with a robust feedback system, we built a user-focused solution that's both functional and accessible.

This project strengthened our skills in system design, user-centered thinking, and hardware-software integration; highlighting how robotics can address everyday health challenges.